Neue Algorithmen zur Bestimmung maximaler gemeinsamer Subgraphen in chemischen Mustern und Räumen

Kumulative Dissertation

zur Erlangung des akademischen Grades

Dr. rer. nat.

an der Fakultät
für Mathematik, Informatik und Naturwissenschaften der
Universität Hamburg

eingereicht beim Fach-Promotionsausschuss Informatik

von

M.Sc.

Robert Julius Schmidt

geboren in Darmstadt

Norderstedt, Dezember 2021

Tag der mündlichen Prüfung: 19.04.2022
Erster Gutachter: Prof. Dr. Matthias Rarey
Zweiter Gutachter: Prof. Dr. Jan Baumback
Dritter Gutachter: Prof. Dr. Christoph Steinbeck

Abstract

In the context of chemistry and cheminformatics, there are different concepts of similarity. Substances can look similar, they can belong to the same class of compounds, they can be made from the same set of source reagents or have similar properties. Focusing on cheminformatics, those similarity measures are preferred that are easy to compute for a computer program. There, three dimensional surfaces, relative positions of functional groups in two and three dimensions, or the existence of certain substructures are used for similarity calculations.

The project of this thesis is focused on similarity measures using maximum common substructures (MCS) which is integrated in two novel application scenarios. These substructures are calculated between a query and a target molecule. Therefore, a novel algorithm is introduced which is capable of solving the MCS problem in a variety of scenarios. The algorithm is designed for a lightweight, efficient and simple structure. Its further properties enable additional control of the computation of connected and disconnected common substructures and the application of additional constraints. As final point of the initial development, the efficiency of the method is shown in comparison to other state of the art algorithms.

Afterwards the developed algorithm is used to scale up the MCS similarity concept from single comparisons on graphs or molecules into combinatorial compound libraries describing several billions of compounds. This requires adaptions to the problem solved and results in an integration of the previously developed algorithm into a larger workflow. This enables a focused enumeration of the most similar compounds in combinatorial compound libraries. Using the correspondingly developed computer program, the efficiency of the concept and algorithm is demonstrated. Furthermore, its application shows the ease of using an MCS similarity search in commercial make-on-demand compound libraries for drug development endeavors.

Finally the adaptions to the MCS method are used to transform the molecular similarity measure into a similarity concept for molecular patterns. There are a few concepts for molecular patterns in cheminformatics which all have in common that they represent molecular substructures. The introduction of a novel concept of generic molecular patterns enables comparisons of molecular patterns independent of their textual representation for the first time. These comparisons are based on an MCS calculation and cover determination of identical patterns, subset relations and similarity on those patterns in general. The applications derived from the developed algorithm cover single pattern comparisons and searching whole pattern collections. They are integrated into an easy to use webserver.

Kurzfassung

Im Anwendungsgebiet der Chemie und auch in der Chemieinformatik gibt es verschiedene Konzepte von Ähnlichkeit. Substanzen können ein ähnliches Aussehen haben, sie können aus den gleichen Ausgangsstoffen bestehen, zu einer Stoffgruppe gehören oder ähnliche physikalische und sonstige Eigenschaften aufweisen. Mit Blick auf die Chemieinformatik werden Ähnlichkeitsmaße bevorzugt, die es einem Computerprogramm ermöglichen, einfach einen Ähnlichkeitswert zu berechnen. In diesem Kontext werden dreidimensionale Strukturen und Oberflächen, Anordnungen funktioneller Gruppen im Zwei- und Dreidimensionalen oder das Vorhandensein von Substrukturen für die Ähnlichkeitsberechnung verwendet.

Im Promotionsprojekt dieser Arbeit wird der Fokus auf Ähnlichkeitsmaße gelegt, die sich aus maximalen gemeinsamen Teilstrukturen (MCS) zwischen einer Anfrage- und einem Zielmolekül errechnen. Es wird ein neuer Algorithmus vorgestellt, der das Problem generisch und in diversen Anwendungsszenarien lösen kann. Zusätzlich werden zwei neue Anwendungsszenarien für die MCS Methodik erschlossen. Der Fokus liegt dabei auf einer schlanken und einfachen Struktur, die es zusätzlich ermöglicht, zusammenhängende und nicht zusammenhängende Teilstrukturen gezielt und mit weiteren Einschränkungen zu berechnen. Die initiale Algorithmusentwicklung schließt mit einer Demonstration der Effizienz im Vergleich zu anderen Methoden ab.

Daraufhin wird der vorgestellte Algorithmus genutzt, um mit dem MCS-Ähnlichkeitsmaß nicht nur einzelne Vergleiche durchzuführen, sondern kombinatorische Substanzbibliotheken, die mehrere Milliarden Moleküle repräsentieren, gezielt zu durchsuchen. Dazu wird das Problem an die Struktur der Räume angepasst und der entwickelte MCS-Algorithmus in einem mehrstufigen Ablauf integriert. Das ermöglicht es, ähnlichste Moleküle zielgerichtet zu enumerieren. Mit dem zum Algorithmus gehörenden Computerprogramm wird sowohl die Effizienz der Methode gezeigt, als auch die Einfachheit demonstriert, den das Durchsuchen kommerzieller Fragmenträume für Wirkstoffentwicklung hat.

Abschließend werden die methodischen Adaptionen am MCS genutzt, um aus dem Ähnlichkeitsmaß für Moleküle ein Ähnlichkeitsmaß auf molekularen Mustern zu entwickeln. In der Chemieinformatik gibt es verschiedene Konzepte, molekulare Muster über Substrukturen darzustellen. Über ein neuartiges Konzept allgemeiner chemischer Muster wird es ermöglicht, mit dem MCS molekulare Muster unabhängig von der textuellen Repräsentation zu vergleichen. Diese Vergleiche umfassen das Feststellen von Identität, Teilmengenrelationen und allgemeiner Ähnlichkeit. Aus dem Algorithmus entstehen mehrere Anwendungen rund um den Vergleich einzelner Muster und ganzer Sammlungen chemischer Muster. Sie können ohne weiteres Expertenwissen über einen Webserver benutzt werden.

Inhaltsverzeichnis

Anhang

1 Einleitung

In der Chemie und auch im Feld der Chemieinformatik gibt es verschiedene Konzepte molekularer Ähnlichkeit.[Haw07, Rar98, Rog10, She02, Sta05, Wil98, Wol05] Je nachdem in welchem Teilgebiet Anwender arbeiten, haben sie auch eine andere Intuition von ähnlichen Substanzen. Für Laien ist die am einfachsten zu erfassende Ähnlichkeit eine gleich aussehende Abbildung von Molekülen oder eine identische Summenformel ohne jegliche Visualisierung. Chemiker hingegen können am Reaktionsverhalten, an physikalischen Eigenschaften oder an Stoffgruppenzugehörigkeit als Ähnlichkeitsbegriff interessiert sein. In der Chemieinformatik kommen weitere Ähnlichkeitsmaße vor. Dort werden Moleküle in der Regel als (zweidimensionale) Graphen oder als dreidimensionale Strukturen modelliert. Zweidimensionale Abbildungen, die von Anwendungen wie sie zum Beispiel Mona[Hil13, Hil15] erzeugt werden, stellen eine abstrakte Form der Ähnlichkeit über die visuelle Darstellung dar. Diese kann weiter unterstützt werden, indem die Darstellung in eine kanonische Form gebracht wird oder ähnlich zu einer ausgewählten Ankerstruktur erzeugt wird. Visuelle Darstellungen bieten ebenfalls den Vorteil, abstrakte theoretischere Ähnlichkeitskonzepte darstellen zu können. Das kann zum Beispiel das Hervorheben gemeinsamer Substrukturen sein. Für dreidimensionale Darstellungen gibt es ebenfalls Ähnlichkeitsmaße, die einfache Visualisierungen unterstützen. 3D-Oberflächenapproximationen sind intuitiv visualisierbar und können effizient überlagert werden, so dass ein Ähnlichkeitswert bestimmt werden kann.[Haw07] Eine ebenfalls dreidimensionale abstraktere Darstellung sind Pharmakophore.[Wol05] Sie stellen Moleküle als reduzierte Graphen dar, wobei die Knoten nur funktionelle Gruppen und Ringsysteme darstellen, die für Interaktionen wichtig sind. Vom Molekülgerüst wird ansonsten vollständig abstrahiert. Die Knoten behalten jedoch ihre Koordinaten im dreidimensionalen Raum.

Wie an den eingehenden Beispielen gesehen, basieren chemieinformatische Darstellungen und Ähnlichkeitsmaße fast immer auf Graphstrukturen oder werden von ihnen abgeleitet. Dazu gehören insbesondere auch maximale gemeinsame Teilstrukturen, die sowohl auf Graphen, als auch auf Abstraktionen von den Molekülgraphen angewendet werden können. Ein anderes Beispiel ist die Graph-Editierdistanz.[Say20] Diese beschreibt die Anzahl von atomaren Änderungsoperationen, um einen Graph in einen anderen zu überführen. Als letztes Beispiel sollen hier noch diverse Arten von molekularen Fingerabdrücken als Ähnlichkeitsmaße dienen.[Bel19, Rin13, Rog10, Xue03] Molekulare Fingerabdrücke werden je nach Anwendung als dichte oder dünne Bitvektoren dargestellt. Die einzelnen Bits stehen jeweils für Eigenschaften der beschriebenen Moleküle. Ursprünglich wurden mit den Bits im Vorfeld definierte Eigenschaften, wie z.B. das Vorhandensein von Teilstrukturen oder funktionellen Gruppen, abgebildet. Durchgesetzt haben sich jedoch Fingerabdrücke, die für jede im Graph auftretende Substruktur Bits erzeugen. Insbesondere das rasante Wachstum von verfügbaren Substanzbibliotheken hat dazu beigetragen, dass molekulare Ähnlichkeit in den meisten Anwendungsfällen über Fingerabdrücke bestimmt wird, da die Ähnlichkeitsberechnung auf gefalteten dichten Bitvektoren mit definierter Länge eine konstante und sehr geringe Laufzeit hat.

Dieser Punkt ist bei graphbasierten Ähnlichkeitssuchen anders. Einerseits gibt es festgelegte Regeln zur Abstraktion, so dass ganze Teilstrukturen auf ihre chemische Funktionalität reduziert werden. Andererseits können Elemente nach Hauptgruppenzugehörigkeit klassifiziert werden, wenn diese sich im betrachteten Kontext

ähnlich verhalten. Nach Reduktion des Graphen wird auf diesem eine Ähnlichkeit bestimmt, die je nach Anwendung und Algorithmus verschiedene Knotengewichte beinhaltet.

Abhängig von der Abstraktion, die dazu führen kann, dass Moleküle grundsätzlich zu Baumstrukturen reduziert werden, kann die Ähnlichkeit effizient bestimmt werden. Im allgemeinen Fall, für den das nicht gilt, wird die Ähnlichkeit mit Algorithmen für das maximale gemeinsame Teilstruktur Problem (MCS) bestimmt.

In der Anwendung haben sich Methoden, die auf Fingerabdrücken arbeiten durchgesetzt. Der Geschwindigkeitsvorteil beim Vergleich von Molekülen durch Fingerabdrücke gegenüber anderen Verfahren, die die Graphstruktur ggf. auch abstrahiert berücksichtigen, basiert auf der starken Abstraktion durch die Bits.

1.1 Maximale gemeinsame Teilstrukturen

Das Problem der maximalen gemeinsamen Teilstruktur (MCS) ist seit langen bekannt und zählt seit den 1970er Jahren zu den NP-vollständigen Optimierungsproblemen der theoretischen Informatik.[Bar76] In der Anfangszeit wurden maximale gemeinsame induzierte Teilstrukturen meist durch Reduktion auf das Cliquen-Problem[Lev73, Bro73] gelöst. Ab den achtziger Jahren wurde neben dem Problem der induzierten maximalen gemeinsamen Teilstruktur, auch das Problem der maximalen gemeinsamen kantenbasierten Teilstruktur auf molekularen Graphen betrachtet.[Nic87] Seitdem wurden diverse Ansätze zur Lösung des allgemeinen Problems als auch Anwendungsfälle für dieses Problem in der Chemieinformatik veröffentlicht. Die Anwendungsmöglichkeiten für MCS-Algorithmen gehen von allgemeiner Ähnlichkeitssuche auf molekularen Graphen [Ray02a] über die Pharmakophorsuche[Cha09] bis hin zu Überlagerungen.[Han98] Die Bandbreite der Anwendungsmöglichkeiten spiegelt sich auch in den bevorzugten Algorithmen zur Lösung der Probleme wider. Neben der allgemeinen Distinktion einen induzierten (MCIS) oder kantenbasierten (MCES) maximalen gemeinsamen Teilgraph zu berechnen, kann dieser sowohl zusammenhängend als auch nicht-zusammenhängend sein. Mit dieser Unterscheidung ergeben sich die vier Varianten des MCS-Problems, die zusammenhängenden, cMCIS und cMCES und die nicht-zusammenhängenden dMCIS und dMCES. Unabhängig vom betrachtetem Problem unterscheiden sich die Algorithmen dadurch, dass diese entweder ein exaktes Ergebnis liefern oder auf Basis von Heuristiken versuchen, ein exaktes Ergebnis anzunähern.[Ehr11, Ray02b, Due16]

Die zu lösenden Problemstellungen implizieren in den meisten Fällen auch die zu verwendende Methodik. Bei kleinen bis mittelgroßen Molekülen mit bis zu 50 Atomen sind exakte Methoden noch performant. Darüber hinaus empfehlen sich heuristische Verfahren aufgrund viel besserer Programmlaufzeiten. Bei der Betrachtung von nicht-zusammenhängenden Teilstrukturen ist der Laufzeitvorteil heuristischer Verfahren bei viel kleineren Atomzahlen signifikant. Als Kompromiss gibt es verschiedene Einschränkungen des zu lösenden Problems, wie zum Beispiel topologische Distanzen,[Kaw11, Mar07, She98] die neben verbesserten Laufzeiten auch die chemische Relevanz der Ergebnisse verbessern.

1.2 Kombinatorische Fragmenträume

Kombinatorische Fragmenträume beschreiben einen graphentheoretischen Ansatz zur Beschreibung und Erzeugung einer Menge von Graphen, ausgehend von Fragmenten und Verbindungsregeln. Die grundlegende Idee hinter Fragmenträumen ist die, dass es einfacher und effizienter ist, eine Menge von Graphen anhand einer Menge von Subgraphen zu beschreiben, die gewissen Regeln folgend, zusammengesetzt werden können. In

der Chemie ist dieses Konzept z.B. durch Markush Strukturen[Det91] anhand von zweidimensionalen Darstellungen bekannt. Markush Strukturen erlauben es, an einer molekularen Struktur Substituenten zu definieren. Sie stellen eine Beschreibung von Molekülmengen dar, die sowohl graphische als auch textuelle Bestandteile verbindet. So wird in der Regel eine Substruktur eines Moleküls graphisch dargestellt und es werden Restgruppen zur Erweiterung der Substruktur angegeben. Die Reste können explizit gegeben oder textuell beschrieben sein. Bei Fragmenträumen ist der Ausgangspunkt eine Menge von Molekülen, sogenannten Fragmenten. Diese sollen jedoch nicht an beliebigen Atomen miteinander verbunden werden können. Um das zu realisieren, werden Verbindungsatome, sogenannte *Linker*, eingeführt. Das Vorhandensein von Linkern macht in diesem Kontext aus Molekülen Fragmente, da durch die Existenz solcher Platzhalteratome ein Molekül als unvollständig angesehen werden muss, weil Linker Verbindungsstellen zwischen Fragmenten darstellen. Die Linker beschreiben nicht nur Ansatzpunkte für die Erweiterung der Moleküle, sie haben auch einen eigenen *Linktyp*, der über die Kompatibilität von Fragmenten entscheidet. Zur Verbindung zweier Fragmente werden die zu den Linkern adjazenten Atome mit der für die kompatiblen Linktypen definierten Bindung verbunden und die beiden Linker jeweils aus dem Molekül entfernt.

1.3 Chemische Muster

Chemische Muster sind ein allgemeines Konzept in der Chemie. Jede Stoffgruppenbezeichnung kann als ein Art allgemeines chemisches Muster betrachtet werden. Eine relevante Anwendung für chemische Muster ist zum Beispiel die Beschreibung von Substanzen in Patenten, die in der Regel durch Markush Strukturen erfolgt.[Det91] Die Kombination textueller und graphischer Elemente erschwert die automatische Suche nach Markush Strukturen. Für die Chemieinformatik sind rein textuelle maschinenlesbare Beschreibungen von Molekülen und chemischen Mustern einfacher und effizienter umzusetzen. Zur Beschreibung chemischer Muster gibt es verschiedene Systeme, die im Wesentlichen über die Beschreibung von Substrukturen funktionieren. Dazu gehören SLN,[Ash97] eine einzeilige Notation für molekulare Muster, die auch Markush Strukturen unterstützt oder MQL[Pro07] eine in Java implementierte Anfragesprache für chemische Muster und auch SMARTS.[Day, Say97] In dieser Dissertation wird der Fokus auf die SMARTS-Sprache gelegt. SMARTS ist eine für chemische Muster abgeleitete Variante von SMILES,[Wei88, Dav89] einer Beschreibungssprache für Moleküle. Die grundlegenden Elemente von SMILES und SMARTS sind identisch, mit dem Unterschied, dass in SMILES Atome und Bindungen beschrieben werden, während es in SMARTS Knoten und Kanten sind. Die Knoten und Kanten in SMARTS beschreiben verschiedene Eigenschaften von Atomen und Bindungen, auf die sie abgebildet werden können. Die folgende Beschreibung konzentriert sich auf SMILES.
In SMILES werden Moleküle als lineare Zeichenketten beschrieben. Atome werden innerhalb von eckigen Klammern [] beschrieben. Innerhalb der Beschreibung steht das Element, die Anzahl an Wasserstoffen und die Ladung, zum Beispiel [NH4+]. Eigenschaften, die definierten Standardwerten des verwendeten Chemiemodells entsprechen, können auch ausgelassen werden. Bindungen werden jeweils zwischen benachbarten Atomen beschrieben. Für Einfach-, Doppel- und Dreifachbindungen werden die Symbole „-“, „=“ und „#“ verwendet. Diese Beschreibung entspricht einem linearen Graphen. Verzweigungen werden in SMILES und SMARTS durch Klammern () ausgedrückt, so dass auch Baumstrukturen möglich sind. Über eine Beschreibung offener Bindungen durch numerische Platzhalter werden letztendlich Ringschlüsse beschrieben, so dass jeder beliebige Graph dargestellt werden kann.

SMARTS verwendet ebenfalls diese Konzepte und erweitert die Beschreibung in den Knoten und Kanten einerseits durch zusätzliche Eigenschaften, als auch durch logische Operatoren, die die Konjunktion oder Disjunktion einfacher Eigenschaften der Knoten und Bindungen ausdrücken. Darüber hinaus ist es in SMARTS

möglich, die Umgebung um einen Knoten rekursiv zu spezifizieren. Das Besondere an einer solchen rekursiven Beschreibung „$(<SMARTS>)“ eines Knotens ist, dass die sogenannte *SMARTS Rekursion* unabhängig vom restlichen Muster das Molekül treffen muss. Alle Atome, die in einer vollständigen Abbildung des rekursiven SMARTS Ausdrucks auf dessen ersten Knoten abgebildet werden können, können auch auf den Knoten mit der SMARTS Rekursion abgebildet werden.

Wenn ein SMARTS in einem Molekül gesucht wird, dann bedeutet dies, dass eine injektive Abbildung der Knoten und Kanten des SMARTS auf die Atome und Bindungen eines Moleküls gesucht wird, die alle Knoten und Kanten des SMARTS beinhaltet. Dieser Vorgang wird im Kontext dieser Arbeit auch *SMARTS matching* bezeichnet.

1.4 Aufbau der Arbeit

Diese Arbeit bezieht sich auf drei zusammenhängende Themenkomplexe, die während der Promotion bearbeitet wurden. Das allgemeine Thema dieser Arbeit ist die Berechnung von Ähnlichkeit in verschiedenen Bereichen der Chemieinformatik. In den drei betrachteten Fällen wird jeweils eine über die maximale gemeinsame Teilstruktur (MCS) berechnete Ähnlichkeit verwendet.

Im ersten Kapitel wird ein effizienter exakter Algorithmus für die Lösung des MCS-Problems auf kleinen chemischen Graphen entwickelt und evaluiert. Dieser hat den Namen RIMACS (**R**ecursive **I**ncremental **MA**ximum **C**ommon **S**ubstructure). RIMACS zeichnet sich durch seine Anpassungsfähigkeit an unterschiedliche Graphtypen aus. Er ist nicht auf ein bestimmtes Graphmodell festgelegt, sondern definiert eine Schnittstelle, über die auf Knoten, Kanten, Nachbarschaften und Inzidenzeigenschaften zugegriffen wird. Die Evaluation des Algorithmus fokussiert sich neben einer allgemeinen Auswertung des Laufzeitverhaltens und einem Vergleich zu heuristischen Verfahren aus der Literatur, auf das eingeschränkte nicht-zusammenhängende MCS-Problem.

Nach der Entwicklung eines Algorithmus für das MCS-Problem folgen zwei neuartige Anwendungsszenarien, die im Kern auf dem MCS basieren, darüber hinaus aber weitere Modellierungen und Einschränkungen benötigen.

Die erste Anwendung ist die Skalierung der MCS-basierten Ähnlichkeitssuche von kleinen enumerierten Molekülmengen hin zu kombinatorischen reaktionsbasierten Fragmenträumen, die mehrere Milliarden Moleküle beschreiben können. Die Modellierung ermöglicht es, die Fragmentstruktur des kombinatorischen Raums auszunutzen, so dass MCS-Berechnungen auf den Fragmenten hinreichend sind, um Aussagen über alle Substanzen im beschriebenen Raum abzuleiten. Die Grundlage für diese erwünschten Implikationen stellt die Einschränkung des MCS auf grundlegende Eigenschaften von Fragmenträumen dar. Der wichtigste Aspekt ist der, dass Teile des Anfragemoleküls mehrfach mit allen Fragmenten des Raums verglichen werden und die Teilergebnisse anschließend zielgerichtet enumeriert werden können. Neben der Entwicklung werden auch Anwendungsbeispiele für die Teilstruktursuche in Fragmenträumen gegeben und das Potential in Verbindung mit kommerziellen Räumen evaluiert.

Für die zweite Anwendung wird der Ähnlichkeitsbegriff von Molekülen zu einem Ähnlichkeitsmaß auf chemischen Mustern weiterentwickelt. Chemische Muster können als Verallgemeinerung von Molekülstrukturen betrachtet werden. Im Vergleich zu Molekülen ist es schwerer, eine einheitliche Darstellung solcher Muster zu definieren, da diese keine reale Grundlage mehr haben. Inspiriert von verschiedenen Sprachen für chemische Muster wird ein theoretisches Konstrukt eines allgemeinen chemischen Musters eingeführt, das erstmalig die

Möglichkeit schafft, Muster sprachunabhängig zu vergleichen. Dieses Konzept wird innerhalb eines Chemiemodells für die Mustersprache SMARTS umgesetzt und ausführlich evaluiert.

Final folgt eine Zusammenfassung der während der Promotion erbrachten Forschungsleistung und ein Ausblick auf mögliche Folgearbeiten und Themen.

2 Berechnung maximaler gemeinsamer Teilstrukturen

2.1 Anforderungen und Motivation

Wie in Kapitel 1.1 beschrieben, gibt es viele verschiedene Ansätze zur Lösung des MCS-Problems.[Ehr11, Ray02b, Due16] In vielen Fällen werden dabei die Algorithmen im Zusammenhang mit einer Anwendung oder einer chemischen Bibliothek vorgestellt.[Dal13, Eng15, Lar16, Kaw11, Ray02a] Auch wenn der Quellcode verfügbar ist, ist dieser häufig an ein gegebenes Chemiemodell angepasst und auf dessen Struktur optimiert. Insbesondere bietet es sich auch aus Gründen der Laufzeiteffizienz an, Algorithmus und Modell zu verweben. Der Nachteil davon ist eindeutig die Wiederverwendbarkeit. Das erste Ziel dieser Dissertation ist nicht nur der Entwurf und die Implementation eines neuen Algorithmus. Dieser soll zudem mit einer einfachen und generischen Schnittstelle versehen werden, so dass unterschiedliche Chemiemodelle und Implementationen von generischen Graphen unterstützt werden. Letzteres soll eine einfache Adaption an verschiedene Graphen zum Beispiel innerhalb des NAOMI[Urb11] Chemiemodells erlauben, wo der Algorithmus sowohl für Moleküle als auch für chemische Muster in der SMARTS Sprache eingesetzt werden soll. Zu den weiteren Anforderungen des Algorithmus gehören die exakte Berechnung induzierter und nicht-induzierter maximaler gemeinsamer Teilstrukturen. Da die Variante des nicht-induzierten Problems durch die Nutzung von Liniengraphen in ein induziertes Problem überführt werden kann,[Nic87, Whi32] besteht die Anforderung, diese effizient und einfach für den Anwender umzusetzen. Die finale Anforderung an den zu entwickelnden Algorithmus ist der zusammenhängende und nicht-zusammenhängende MCS (dMCS). Beide Varianten können von vielen Algorithmen berechnet werden.[Due18, Eng15, Kaw11, Ray02a] Deshalb soll nicht nur der dMCS berechnet werden. Dieser soll vom Nutzer auch sinnvoll beschränkt werden können, so dass das Ergebnis einen guten Kompromiss aus akzeptabler Laufzeit und gesteigerter chemischer Relevanz der Ergebnisse ermöglicht. Dazu soll die Anzahl der Zusammenhangskomponenten und deren minimale Größe parametrisierbar sein. Beide Bedingungen werden von existierenden Algorithmen unterstützt,[Cao08, Har11] wurden aber bisher noch nicht ausführlich evaluiert. Abschließend soll der Algorithmus für eine optimale Wiederverwendbarkeit der Bibliothek „freie Software“ sein und der Quellcode entsprechend veröffentlicht werden.

Beim Design von Algorithmen müssen neben den Anforderungen weitere richtungsweisende Entscheidungen getroffen werden. Es gibt verschiedene Ansätze das Problem zu lösen und in jeder Publikation werden die Vorteile der eigenen Implementation hervorgehoben. Ohne unabhängige Vergleichsstudien ist es schwer, wirkliche Vorteile unterschiedlicher Modellierungen zu erkennen. Grundsätzlich besteht bei MCS-Algorithmen immer die Möglichkeit, das Problem auf das Cliquenproblem zu reduzieren. Im Vergleich zu anderen heuristischen Verfahren haben sich cliquenbasierte Algorithmen, die eine lokale Iteration benutzen, durchgesetzt.[Due18, Eng15, Gro08] Bei den exakten Methoden ist die Situation weniger eindeutig. Das größte Problem der Reduktion auf das Cliquenproblem beim Vergleich chemischer Molekülgraphen ist die Tatsache, dass Molekülgraphen dünne Graphen sind, die Anzahl an Kanten also linear von der Knotenzahl abhängt.[Eng15] Daraus folgt, dass der erzeugte Kompatibilitätsgraph, in dem die Clique gesucht wird, ein dichter Graph ist. Das ist ein Nachteil, da die Clique im Vergleich zum Graphen immer klein ist. Der Fokus auf chemische Graphen hat im Allgemeinen den Vorteil, dass alle Kanten und Knoten Element- und Bindungstypen haben, die die Kompatibilität

einschränken. In der graphentheoretischen Betrachtung werden diese Typen Farben genannt.[San14] Ein anderer Ansatz, die Entwicklung eines substrukturbasierten Algorithmus[Cao08] hat nicht den genannten Nachteil der Cliquenmodellierung. Dazu kommt, dass sich substrukturbasierte Algorithmen[Cor04, May19] für die Substruktursuche, ein vermeintlich einfacheres aber auch ein NP-vollständiges Problem, durchgesetzt haben.

2.2 Stand der Forschung

Wie schon in der Motivation bemerkt, ist die Cliquenmodellierung[Caz05, Koc01, Ray02a, Mcc16, She98] vor allem für heuristische Verfahren interessant.[Eng15, Gro08, Kaw11] Im Allgemeinen sind die Laufzeiten zur Berechnung eines zusammenhängenden MCS (cMCS) deutlich geringer als die des dMCS.[Due18] Damit ein cMCS in einer Clique gefunden werden kann, muss die Modellierung angepasst werden.[Caz05, Koc01] Ansonsten wird eine im Normalfall nicht zusammenhängende Abbildung berechnet. Eine beliebte Methode zur Beschleunigung des dMCS sind topologische Distanzen.[Kaw11, Mar07, She98] Die Idee dahinter ist es, dass die einzelnen abgebildeten Zusammenhangskomponenten ähnliche Distanzen zueinander haben und das auch für die jeweiligen Knotenabbildungspaare gilt. In Bezug auf das Cliquenproblem hat diese Modellierung den Vorteil, dass der Kompatibilitätsgraph ausgedünnt wird. Zudem steigt die chemische Relevanz der Ergebnisse, da große Zusammenhangskomponenten nur dann abgebildet werden können, wenn die relative Anordnung zueinander gleich ist. Diese wird, wie gesehen, nur implizit betrachtet. In einem dMCS kann die relative Anordnung aber auch einen Teil der Gewichtung für das Gesamtresultat ausmachen. Stahl *et al.* [Sta05] berechnen einen dMCS und bewerten die Anordnung der größten drei Zusammenhangskomponenten mit mindestens vier Atomen. In Abhängigkeit der Anordnung und der einzelnen Größen der Zusammenhangskomponenten wird der initial berechnete Ähnlichkeitswert um bis zu 0.3 abgewertet oder 0.1 aufgewertet.

Eine ganz andere Art das Problem zu lösen ist die Substrukturenumeration.[Dal13, Tak87, Var79] Ausgehend von einer „kleinsten“ Struktur werden Substrukturen enumeriert und in einer Menge von Graphen gesucht. Diejenige mit maximaler Knotenanzahl entspricht dem cMCS. Diese Art das Problem zu lösen hat vor allem Vorteile, wenn der MCS zwischen mehr als zwei Graphen gesucht wird. In einem solchen Fall werden die Substrukturen einfach in einem weiteren Graph gesucht. Die Skalierung bezüglich der Anzahl an Graphen ist als linear einzuordnen. Alternativ wurde das Problem mit einem Cliquenalgorithmus und der Enumeration von maximalen gemeinsamen Teilstrukturen gelöst, von denen maximale Schnitte bestimmt wurden.[Har11] Die Anzahl an MCS Berechnungen wächst hier ebenfalls linear mit der Anzahl an Graphen.

Als letzter häufig genutzter Ansatz seien substrukturbasierte Algorithmen genannt.[Cao08, Lar16, McG82, Van13] Diese haben die Eigenschaft, dass die Abbildung zwischen den Graphen direkt modelliert wird. Das ermöglicht schlanke und einfache Strukturen für effiziente Algorithmen. Im Vergleich zur häufig verwendeten Cliquenmodellierung sind triviale und akkurate Abschätzungen der erreichbaren Ergebnisgüte möglich, indem z.B. die Anzahl der verbleibenden abbildbaren Knoten verwendet wird. Exakte Algorithmen eignen sich insbesondere für kleine Moleküle und Graphen[Cao08] und bei großen (biologischen) Netzwerken ist ein heuristischer Ansatz unabdingbar.[Lar16] Dieser ist durch einen „Simulated Annealing“-Ansatz in Verbindung mit einer lokalen Suche in der Lage, einen heuristischen MCS auf einer Vielzahl von Graphen zu berechnen.

2.3 Der RIMACS Algorithmus

Dieses Kapitel fasst den RIMACS (Englisch für: **R**ecursive **I**ncremental **MA**ximum **C**ommon **S**ubstructure) Algorithmus zusammen, der in [D1] entwickelt wurde. RIMACS ist ein substrukturbasierter generischer MCS-Algorithmus, der für die in Kapitel 2.1 beschriebenen Anforderungen entwickelt wurde. Die Publikation [D1] betrachtet nicht nur die Entwicklung und insbesondere interne Heuristiken zur Beschleunigung unter Bewahrung der exakten Ergebnisse, sondern auch den Laufzeitvergleich mit anderen Methoden und die Evaluation des eingeschränkt nicht-zusammenhängenden MCS-Problems. Die RIMACS Methode selbst ist alleinig für den induzierten MCS (MCIS) entworfen, bei dem eine Knotenabbildung maximiert wird. Der nicht-induzierte MCS (MCES), bei dem eine Kantenabbildung maximiert wird, wird über die Modellierung als Liniengraph gelöst. Dabei entspricht die Anzahl an Kanten der Anzahl Knoten im Liniengraphen. Somit gelten Beschreibungen der folgenden Kapitel für beide Varianten.

2.3.1 Vergleich der Einschränkungen für den Nichtzusammenhang

Der RIMACS Algorithmus ist als effizienter generischer MCS-Algorithmus mit einem Fokus auf kleine und mittlere Knotenzahlen von molekularen Graphen ausgelegt. Für eine optimale Nutzung als Ähnlichkeitsmaß auf Molekülen muss sowohl der cMCS als auch der dMCS berechnet werden können. Da letzterer ohne ein heuristisches Verfahren in der Regel nicht effizient gelöst werden kann, setzt RIMACS auf aus der Literatur bekannte Einschränkungen:

1 Kontrolle über die Anzahl der Zusammenhangskomponenten, aus denen die finale Abbildung besteht

2 Mindestanzahl an Knoten, beziehungsweise Kanten in einer Zusammenhangskomponente, sofern es mindestens zwei gibt.

Der Vorteil beider genannter Bedingungen ist, dass sie sich gut in einen Algorithmus integrieren lassen, der zusammenhängende Zwischenergebnisse bis zum Erreichen eines lokalen Maximum expandiert. Damit lässt sich der Algorithmus in Phasen der Expansion von Zusammenhangskomponenten unterteilen. Zwischen den Phasen werden die genannten Einschränkungen geprüft, die sich ebenfalls in interne Datenstrukturen und die Abschätzung der erreichbaren Ergebnisgüte integrieren lassen.

Die Modellierung topologischer Distanzen bevorzugt Adjazenzmatrizen für die Graphen, da alle paarweisen Distanzen von Interesse sind. Diese wären eine Voraussetzung für eine effiziente Integration in substrukturbasierte Algorithmen. Die für eine Cliquenmodellierung üblichen Kompatibilitätsgraphen bieten hingegen ein Werkzeug, in das sich die Distanzen sinnvoll integrieren lassen. Daher wurden diese genauso wie in [Cao08] nicht in den Algorithmus integriert.

Die für den nicht-zusammenhängenden MCS interessanten zusätzlichen Bewertung der Anordnung der Zusammenhangskomponenten, wie in [Ray02a] beschrieben, wurde ebenfalls evaluiert und aufgrund sinkender Laufzeiteffizienz verworfen. Eine Überprüfung der Topologie der Zusammenhangskomponenten kann immer nur als ein nachgelagerter Schritt in einen Algorithmus integriert werden. Bei deren Evaluation kann dann die Abschätzung der noch erreichbaren Ergebnisgüte dramatisch sinken, was eventuelle Vorteile schnell ausgleicht. Aus genannten Gründen wird dieser Anwendungsfall in RIMACS nicht unterstützt.

2.3.2 Korrektheit

Jeder Algorithmus, der auf ein Problem angewendet wird, sollte dieses auch korrekt lösen. Damit das sichergestellt ist, muss die Korrektheit formal bewiesen werden. Handelt es sich hingegen um einen approximativen Ansatz ein Problem zu lösen, kann in der Regel nicht bewiesen werden, dass jede Probleminstanz korrekt gelöst wird. In solchen Fällen hingegen ist eine Evaluation der Ergebnisse und eine Diskussion von Fehlern und Fehlerquellen notwendig. In der chemieinformatischen Literatur zu MCS Algorithmen gibt es nur wenige Beweise dafür, dass eine Methode auch korrekt ist. Alle cliquenbasierten Verfahren basieren auf dem Beweis der Reduktion des MCS Problems auf Instanzen des Cliquenproblems.[Lev73] Instanzen der nicht induzierten MCS Varianten nutzen die Beweise von Whitney und Nicholsen,[Nic87, Whi32] die besagen, dass ein nicht induzierter MCS mit wenigen Ausnahmen einem induzierten MCS im Liniengraphen entspricht. Einen zusammenhängenden MCS mittels Cliquen zu berechnen erfordert einige Modifikationen im Cliquenalgorithmus, da die bestehende Clique nicht mehr mit allen möglichen benachbarten Knoten erweitert werden darf, da dies den Zusammenhang des dargestellten MCS verletzt. Eine solche Anpassung des Problems und der Algorithmen bedarf eines Beweises, wie von Koch[Koc01] erbracht und von Cazals korrigiert.[Caz05] Die meisten exakten Algorithmen sind sogenannte Backtracking Algorithmen. Für die Berechnung der Cliquen ist die prinzipielle Vorgehensweise ähnlich. Die Unterschiede in den Verfahren und insbesondere auch die Vor- und Nachteile im Laufzeitverhalten folgen aus der Möglichkeit obere und untere Schranken für das bestmögliche Ergebnis, sowie des aktuell erreichbaren Ergebnis zu berechnen. Daraus ergeben sich Möglichkeiten Bereiche im Zustandsraum zu ignorieren, die das gesuchte maximale Ergebnis nicht beinhalten. In der Chemieinformatik angewendete Methoden, wie z.B. der RASCAL Algorithmus[Ray02a] nutzen eine Vielzahl aus der mathematischen Literatur bekannten Optimierungsstrategien. Die Korrektheit der RASCAL Methode im Gesamten wird aber nicht noch einmal bewiesen. Das MCS Problem lässt sich wie in den bisherigen Kapiteln beschrieben, nicht nur über die Reduktion auf das Cliquenproblem lösen. Cao[Cao08] entwickelte einen substrukturbasierten Algorithmus, dessen Ideen von Van Berlo[Van13] aufgegriffen und als heuristisches Verfahren beschleunigt wurde. Ein Korrektheitsbeweis wurde für diesen Ansatz noch nicht erbracht. Die RIMACS Methode[D1] basiert auf dem von Cao[Cao08] präsentierten Algorithmus und liefert einen Korrektheitsbeweis. Neben der Korrektheit der grundlegenden Methode werden auch alle verwendeten Approximationen betrachtet, für die jeweils festgestellt wird, dass sie keinen Einfluss auf die Güte der Berechnung haben. Eine gute Evaluation von heuristischen Verfahren[Eng15] betrachtet neben der Laufzeit auch immer die Güte der erreichbaren Ergebnisse. Das kann im Vergleich zu anderen Methoden oder zur exakten Lösung erfolgen,[Kaw11] sofern diese mit akzeptablen Aufwand berechenbar ist.

2.3.3 Die RIMACS Vergleichsstudien

In [D1] wurde RIMACS in zwei verschiedenen Szenarien evaluiert. Einerseits wurde der Einfluss auf die Laufzeit und Ergebnisgüte der Beschränkung des dMCS, andererseits die Laufzeiten im Vergleich zu anderen Methoden bezüglich des cMCS evaluiert. In beiden Fällen wurden publizierte Datensätze von Englert[Eng15] und zwei neu erstellte verwendet.[D1] Letzterer der beiden, soll eine reproduzierbare zufällige Auswahl an Molekülen darstellen. Eine zufällige Auswahl von Testfällen ist ein häufig gewähltes Mittel in Evaluationsstudien.[Bel20, Cao08, Due18, Lar16, Les19] Sofern die Datensätze nicht mit veröffentlicht werden, sind die erzielten Ergebnisse aber nicht reproduzierbar. Auch wenn solche Selektionen meistens tatsächlich zufällig ablaufen, ist es nicht möglich festzustellen, ob ein weiteres (unbekanntes) Kriterium die Auswahl mit beeinflusst hat. Im Fall des NCI-Key genannten Datensatzes in [D1] wurde eine dreistellige Zahl im Suffix der automatisiert erstellten Molekülnamen zum Auswahlkriterium. Der Grund für die Auswahl der beiden Ziffernfolgen „425“ und „545“

hängt mit den Zielen zusammen, eine zufällige Auswahl zu simulieren und den Datensatz nicht veröffentlichen zu müssen. Das NAOMI[Urb11, Urb14] Chemiemodell ist streng und teilweise selektiv beim Einlesen organischer Moleküle, wenn diese Halbmetalle oder Metalle enthalten. Die beiden als Suffix gewählten Ziffernfolgen entsprechen der Selektion aller Datensätze eines dreistelligen Namenssuffix, die vollständig mit NAOMI einlesbar sind. Solche Entscheidungskriterien werden in der Regel nicht mitveröffentlicht. Ebensowenig erfahren Leser, ob eine zufällige Auswahl von Strukturen wiederholt wurde oder überhaupt nicht zufällig war.

2.3.3.1 Evaluation des eingeschränkten nicht-zusammenhängenden MCS

Der Zugewinn abbildbarer Knoten des dMCS im Vergleich zum cMCS wurde bisher noch nicht evaluiert. In [D1] wird dies auf zwei verschiedenen Datensätzen ausgewertet. In erster Linie geht es um einen Laufzeitvergleich und die farbliche Visualisierung der Skalierung für die einzelnen Konfigurationen des dMCS. Darüber hinaus wird für jede der Konfiguration der durchschnittliche Zugewinn an abgebildeten Knoten gegenüber dem cMCS abgebildet. Diese Analyse ist in den Abbildungen 2.1 und 2.2 für den in [D1] EvalSet genannten Datensatz durchgeführt. EvalSet wurde aus aktiven und inaktiven Molekülen des DUD-Datensatzes[Mys12] erstellt. Abbildung 2.1 verdeutlicht die exponentielle Natur hinter dem dMCS und auch, dass das Erzwingen großer Zusammenhangskomponenten vergleichbare Laufzeiten zum cMCS ermöglicht. Unter Hinzuziehen der Ergebnisse aus Abbildung 2.2 folgt aber auch, dass große Zusammenhangskomponenten nur selten auftreten und einen signifikanten Zugewinn für die Abbildung ermöglichen. In [D1] wird geschlussfolgert, dass drei bis fünf Zusammenhangskomponenten mit einer Mindestgröße von drei bis fünf Atomen einen guten Kompromiss aus akzeptabler Laufzeit und Ergebnisverbesserung darstellen.

Die Ansätze aus der Literatur wurden vergleichsweise evaluiert. Insbesondere der Fokus auf den Zugewinn an abbildbaren Knoten oder Kanten wurde bisher nicht explizit erwähnt. Kawabata[Kaw11] hat eine vergleichbare Studie für die Evaluation topologischer Distanzen durchgeführt. Diese wurde jedoch für den heuristischen Algorithmus durchgeführt. So wurden sowohl die Laufzeiten und Ergebnisse im Vergleich zu einem exakten cliquenbasierten Algorithmus betrachtet. In der Studie von Englert und Kovács[Eng15] hingegen wurde kein exaktes Verfahren zum Vergleich herangezogen, sondern eine obere Schranke der exakten Lösung.

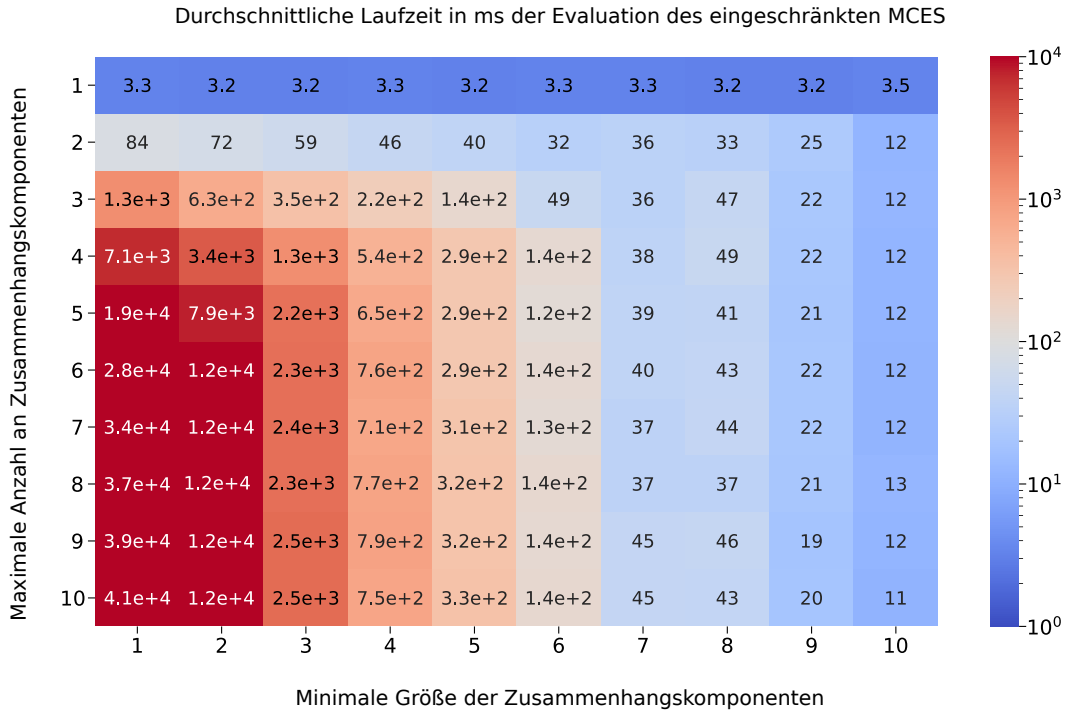


Abbildung 2.1: Darstellung der durchschnittlichen Laufzeiten aller paarweisen MCES Berechnungen der Moleküle des EvalSet genannten Datensatzes. Diese Abbildung wurde aus [D1] übernommen und ins Deutsche übersetzt.

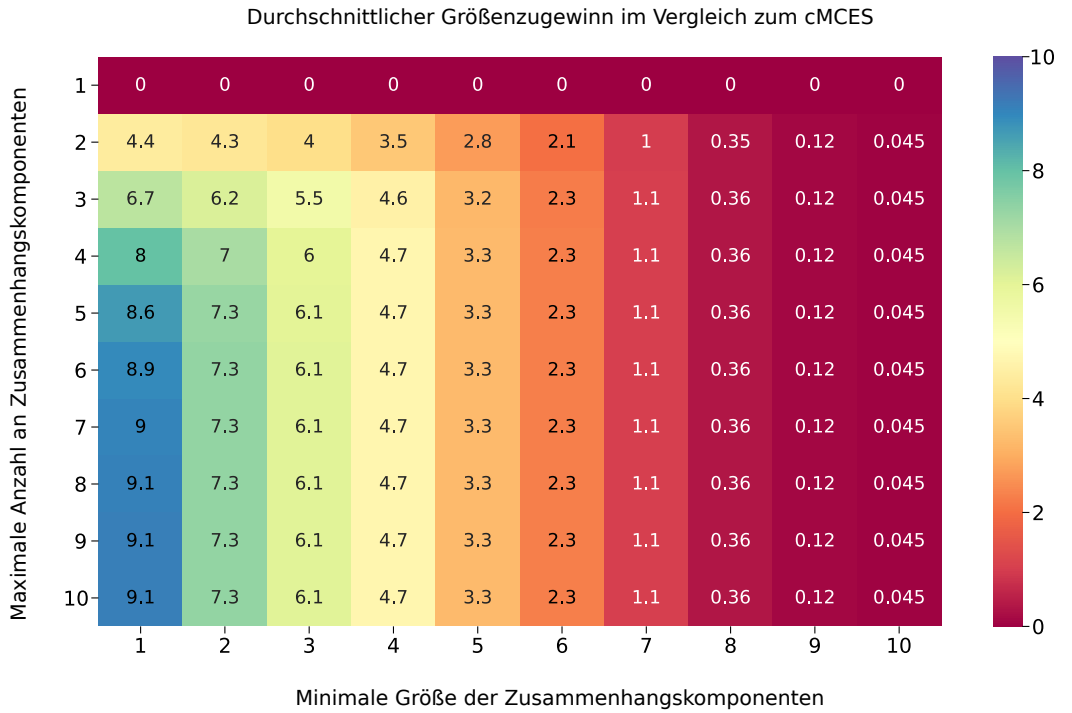


Abbildung 2.2: Darstellung des durchschnittlichen Zugewinns an abgebildeten Atomen aller paarweisen MCES Berechnungen im Vergleich zum cMCES der Moleküles des EvalSet Datensatzes. Diese Abbildung wurde aus [D1] übernommen und ins Deutsche übersetzt.

2.4 Vergleich vom MCS-Algorithmen

Genauso, wie in anderen Teildisziplinen der Chemieinformatik ist es auch für das exakt lösbare MCS Problem schwer, gute Vergleichsstudien zu finden.[Due18, Fri17] Sofern die Softwarepakete der Vergleichskandidaten nicht als Open-Source Software zur Verfügung stehen, ist es häufig schwierig, Lizenzen für die Anwendungen innerhalb einer Vergleichsstudie zu erhalten. Insbesondere kommerzielle Anbieter haben kein Interesse daran, in einer Vergleichsstudie schlecht abzuschneiden.[Ope, Sch].

Unabhängig von genannten organisatorischen Schwierigkeiten für ausführliche Vergleichsstudien ist häufig das Ziel die Vorstellung und Etablierung einer neuen Methode. Frei verfügbare Datensätze, auf denen das Laufzeitverhalten der Methoden evaluiert wurden, sind ein Kompromiss, um beide Ziele zu erreichen, der bisher aber zu selten für MCS-Algorithmen angewendet wird. Neue Datensätze bieten im Vergleich zur Nutzung bestehender immer die Möglichkeit, die Stärken des entwickelten Algorithmus besser hervorzuheben, aber die Schwächen des Ansatzes zu umgehen. Eine gute Evaluation wird sich immer daran messen lassen müssen, bestehende und gegebenenfalls als schwer zu lösen bekannte Datensätze zu inkludieren. Genau das wurde mit der Wahl der Datensätze von Englert[Eng15] in [D1] angestrebt.

2.4.1 Vergleichsstudien in der Literatur

Es gibt verschiedene Motivationen bestehende Algorithmen zu vergleichen. Der Vergleich an sich kann das Ziel sein, um bei einer Menge konkurrierender Ansätze für ein Teilgebiet, eine Methode der Wahl zu bestimmen. Anderenfalls ist für Projekte, die auf einer Berechnung aufbauen, diese aber nur benutzen, eine Evaluation ebenfalls wichtig, da die Wahl die später erzielbaren Laufzeiten stark beeinflussen kann. Und abschließend sollten für jeden neu vorgestellten Algorithmus die Laufzeiten evaluiert und verglichen werden.

So kann in aufbauenden Projekten eine kleine Evaluation verschiedener Ansätze durchgeführt werden, wenn die beste Technologie zur Integration in ein komplexeres Problem gesucht wird. Genau das haben unter anderem Stahl *et al.* durchgeführt.[Sta05] Andererseits haben Englert und Kovács[Eng15] einen externen Algorithmus im Vergleich zu den verschiedenen Varianten ihrer Implementationen betrachtet. Der in [D1] durchgeführte Vergleich orientiert sich stark an dem von Englert. In beiden mit Beispielen untermalten Fällen liegt der Fokus des Vergleichs auf dem spezifischen Anwendungsgebiet der Algorithmen.

Insgesamt erweist sich der Vergleich verschiedener MCS-Algorithmen als schwierig. Auch wenn immer wieder neue Algorithmen vorgestellt werden, gibt es in den wenigsten Fällen ausreichend Vergleichsstudien, die neue Algorithmen von bestehenden sinnvoll abgrenzen und unterscheidbar machen. Das liegt an verschiedenen Gründen. Viele Algorithmen werden innerhalb von nicht-öffentlichen Produkten oder Bibliotheken implementiert, so dass der Quellcode oder eine Vergleichsanwendung nicht zur Verfügung stehen. Für den Fall, dass die Algorithmen in z.B. einer frei verfügbaren Bibliothek implementiert sind, ergeben sich zwei Möglichkeiten für einen Vergleich: Eine Anbindung an die eigene Bibliothek, bei der die internen Strukturen angepasst werden müssen oder der Vergleich über verschiedene Chemiemodelle. Das letzteres schwierig ist, zeigt die Unterscheidung der RIMACS-Versionen in Tabelle 6 von [D1].

Die genannten und weitere Probleme lassen sich im Vergleich von MCS Algorithmen beobachten, wie von Duesbury[Due18] durchgeführt. In der Publikation werden 11 verschiedene Algorithmen aus unterschiedlichen Quellen miteinander verglichen. Für ein geeignetes Maß an Vergleichbarkeit wurden alle diese Methoden zudem in Java reimplementiert. Ziel der Reimplementation war es, unabhängig von der Sprache einen besten

Algorithmus und Ansatz zu finden. Dieser löst somit das Problem der Chemiemodelle, bringt aber das Problem mit sich, dass Laufzeitoptimierungen der Originalimplementationen verloren gehen können.

Zusammenfassend bleibt festzustellen, dass es schwierig ist, einzelne Algorithmen zu vergleichen, die normalerweise in Anwendungen und chemischen Modellen eingebunden sind. Die Implementation dieser, in unterschiedlichen Programmiersprachen, ist dabei ein zusätzliches Problem. Wenn ganze Anwendungen und Programmfolgen verglichen werden können, löst das einige der hiesigen Probleme, bringt aber andere mit sich, die sich aus den Möglichkeiten der Parametrisierung ergeben.[Fri17]

2.5 Zusammenfassung und Ausblick

Die Entwicklung und der systematische Vergleich des RIMACS Algorithmus haben die Schwierigkeiten bei der Entwicklung effizienter MCS-Berechnungen aufgezeigt. Auf kleinen Molekülen ist die Berechnung eines zusammenhängenden exakten MCS vergleichbar zur Laufzeit von zusammenhängenden heuristischen MCS Ergebnissen. Sofern der Zusammenhang nicht mehr erforderlich ist, sind heuristische Methoden die logische Wahl. Durch Einschränkungen des betrachteten Problems kann insbesondere die Laufzeit von exakten Methoden verbessert werden, und im Optimalfall verbessern eingeführte Randbedingungen die Relevanz der Ergebnisse, indem die Intuition von Ähnlichkeit im Anwendungsbereich besser abgebildet wird. Besonders eindrucksvoll lässt sich dieser Effekt bei der Verwendung topologischer Distanzen in cliquenbasierten MCS-Algorithmen beobachten. Andere Einschränkungen des MCS-Problems und Einschränkungen auf den zu durchsuchenden Graphen können ebenfalls das Potenzial bieten, MCS-Berechnungen soweit zu beschleunigen, dass ähnlichste Strukturen in großen Datensätzen gefunden werden können. In Molekülgraphen stehen die Anzahl an Knoten und Kanten immer in linearen Verhältnissen zueinander. Im Vergleich zu allgemeinen Graphen lohnt sich auf ihnen die Unterscheidung, ob Knoten und Kanten zyklisch, oder azyklisch sind. Diese Beobachtung kann genutzt werden, um das MCS-Problem einzuschränken und die Berechnung auf unter entsprechenden Randbedingungen erstellten großen Datenmengen soweit zu beschleunigen, dass die lineare Suche mit bestehenden effizienten Methoden im Vergleich weniger effizient wird.

RIMACS ist einer von vielen publizierten Algorithmen zur Lösung des MCS Problems. Damit er beachtet wird, sollte er in eine Anwendung integriert werden, die auf der Basis des Algorithmus entstanden ist. Neben der Effizienz ist die generische Schnittstelle des RIMACS Algorithmus ein zentrales Element, dass es ermöglicht, ihn in verschiedenen Problemfeldern anzuwenden. RIMACS soll außer für den Vergleich von Molekülen auch für andere Graphen eingesetzt werden können. Insbesondere molekulare Muster haben die gleiche Graphstruktur wie Moleküle. RIMACS eignet sich exzellent um mittels eines MCS solche Graphen zu vergleichen und tiefgreifende Erkenntnisse in einem neuen Feld zu erlangen.

In anderen Zusammenhängen ist es auch möglich, RIMACS in komplexe algorithmische Anwendungsabfolgen effizient zu integrieren, um einen MCS in kombinatorischen Graphbibliotheken zu berechnen, der sich aus mehreren MCS Teilergebnissen berechnet. Für solche Einsatzzwecke bietet RIMACS verschiedene Schnittstellen, die dazu genutzt werden können, die Kompatibilität von Knoten über Hinweise an den Algorithmus einzuschränken oder die Bewertung eines gewichteten Ergebnis nachträglich zu beeinflussen.

Abschließend ermöglicht die Veröffentlichung des RIMACS Quellcodes auf der Plattform GitHub viele weitere Einsatzmöglichkeiten und Adaptionen des Algorithmus.

3 Ähnlichkeitssuche in kombinatorischen Fragmenträumen

3.1 Was sind Fragmenträume?

Wie in Kapitel 1.2 beschrieben sind chemische Fragmenträume ein Konzept zur Repräsentation kombinatorischer Räume und im Speziellen auch von Molekülmengen. Sie bestehen aus Fragmenten und dazugehörigen Verbindungsregeln. Die Räume sind so definiert, dass Fragmente immer über Baumstrukturen zu den Molekülen des Raums verbunden werden. Zur besseren Unterscheidung von anderen Fragmentraumdarstellungen werden sie im Folgenden *klassische Fragmenträume* genannt.

Das theoretische Konzept der Fragmente und Linker erlaubt es, dass Fragmente mit jeweils zwei und mehr Linkern auch zu Zyklen zusammengesetzt werden könnten. Für die Anwendung des MCS in Fragmenträumen wird dies allerdings verboten. Insbesondere sind alle Bindungen, die zwei Fragmente eines Moleküls in klassischen Fragmenträumen verbinden azyklisch. Diese Einschränkung resultiert aus der Entwicklung der Fragmenträume für die Feature-Tree[Rar98] und FTrees-FS[Rar01] Algorithmen, die auf Baumstrukturen definiert sind. Diese Einschränkung hat sich ebenfalls als bedeutsam erwiesen, um Fragmenträume systematisch zu enumerieren.[Lau16] In alternativen Darstellungen von Fragmenträumen, wie sie dem SpaceLight[Bel20] Algorithmus zugrunde liegen, werden Ringschlüsse über Fragmentgrenzen hinweg in sogenannten *topologischen Fragmenträumen* modelliert.

3.1.1 Erstellung von Fragmenträumen

In der Anfangszeit wurden klassische Fragmenträume anhand von Retrosyntheseregeln, sogenanntem Shredding,[Deg08, Lew98] erstellt. Basierend auf einer Menge von Molekülen wurden diverse Bindungen geschnitten und Fragmente gewonnen. An den Schnittstellen wurden Linker angefügt, wobei die Gesamtzahl an Linktypen insgesamt gering ist. Auf diese Art gewonnene Fragmente hatten häufig zwei oder mehr Linker. Um ein Molekül, das sich aus mehr Fragmenten zusammensetzt, zu terminieren, ist es zwingend notwendig, dass es Fragmente mit genau einem Linker gibt. Diese Eigenschaft haben sogenannte Terminatoren übernommen, spezielle Fragmente, die an Fragmentkombinationen angebaut werden können, um Linker mit validen molekularen Substrukturen zu ersetzen. Terminatoren haben in der Regel ein Minimum an Schwertatomen. Das heißt, dass Linker mit Einfachbindungen meistens mit einen Wasserstoff terminiert wurden. Doppel- und Dreifachbindungen wurden oft mit minimalen Alkenen oder Alkinen terminiert.

Die aktuelle Entwicklung von klassischen Fragmenträumen verfolgt eine andere Strategie, seitdem sich gezeigt hat, dass Substanzen aus retrosynthetischen Räumen häufig chemisch schwer zugänglich sind.[Hof19] Aktuell erstellte Fragmenträume basieren auf Syntheseregeln, die auf Moleküle vorhandener Datenbanken angewendet werden. Kommerzielle Anbieter[Ena, OTA, WuX] solcher Fragmenträume versprechen 80% beliebiger Moleküle aus ihren kommerziellen Fragmenträumen innerhalb weniger Wochen synthetisieren und liefern zu können. Tabelle 3.1 gibt eine Übersicht über die Anzahl an Fragmenten, Verbindungsregeln und Molekülen in kommerziellen und öffentlichen Fragmenträumen. Für die Effizienz von Fragmentraumalgorithmen ist aber insbesondere die letzte Spalte, der Exponent, der das Verhältnis der Anzahl an Fragmenten zur

Anzahl der beschriebenen Moleküle beschreibt wichtig. An diesem wird die Effizienz im Vergleich zu einer linearen Prozessierung aller Substanzen gemessen. Die Tatsache, dass die Exponenten kommerzieller Räume häufig oberhalb von zwei liegen, wie aus Tabelle 3.1 hervorgeht, spricht für eine effiziente Darstellung und zeugt davon, dass viele Moleküle aus drei und mehr Fragmenten zusammengesetzt werden.

Tabelle 3.1: Eine Übersicht über die Anzahl an Fragmenten, Verbindungsregeln und dargestellten Moleküle von kommerziellen und öffentlichen Fragmenträumen. Darüber hinaus ist der Exponent angegeben, der das Verhältnis der Anzahl Fragmente zur Anzahl Moleküle beschreibt.

Fragmentraum	Anz. Fragmente	Anz. Verbindungsregeln	Anz. Moleküle	Exponent
CHEMriya[OTA21]	~ 50.000	~ 10.000	$\sim 1{,}1 \cdot 10^{10}$	2,13
GalaXi[WuX20]	~ 22.000	~ 190.000	$\sim 2{,}1 \cdot 10^{9}$	2,15
REAL Space[Ena21]	~ 890.000	~ 8.500.000	$\sim 1{,}9 \cdot 10^{10}$	1,73
KnowledgeSpace[Det10, Bio19]	~ 318.000	~ 4.800.000	$\sim 2{,}9 \cdot 10^{14}$	2,63

3.2 Motivation

In Wirkstoffentwicklungsprojekten werden Substanzbibliotheken nach Struktureigenschaften und häufig auch nach molekularer Ähnlichkeit zu einer vorhandene Leitstruktur durchsucht. Dabei gilt es das Problem zu lösen, den nächsten, besseren Wirkstoffkandidaten im verfügbaren chemischen Raum zu finden. Andererseits ist es ebenfalls eine Anforderung an die durchsuchten Substanzbibliotheken, dass diese so groß und divers wie möglich sind. Diese Anforderungen begünstigten letztendlich das Aufkommen von Anbietern chemischer Substanzen, die diese nicht vorrätig haben, sondern erst bei Bestellung synthetisieren. Die von solchen Anbietern vertriebenen Substanzen werden nachfolgend als *make-on-demand Compounds* bezeichnet. Der Erfolg und die stetig wachsende Verbreitung von kommerziellen und proprietären Fragmenträumen von Pharmafirmen macht Fragmenträume im Allgemeinen zu einem interessanten und zukunftsträchtigen Forschungsgebiet. Die stetig wachsende Anzahl an make-on-demand Compounds zu enumerieren, wird immer aufwendiger und erfordert Ressourcen im Wert von mehreren zehntausend bis zu hunderttausenden Euro.[Irw20] Die Beschreibung solcher Molekülmengen über Fragmenträume ermöglicht es jedoch, diese viel kompakter darzustellen und verfügbar zu machen. Fragmenträume sind die Antwort auf stetig wachsende virtuelle Substanzbibliotheken von make-on-demand Compounds.

3.2.1 Notwendigkeit einer effizienten Moleküldarstellung

Die Anzahl verfügbarer make-on-demand Compounds hat in den letzten Jahren stetig zugenommen und liegt mittlerweile bei über 33 Milliarden.[OTA21, WuX20, Ena21] Große Pharmafirmen verfügen ebenfalls über auf Reaktionswissen basierende Räume, die diverse Zehnerpotenzen größer sein können als die kommerziellen Räume.[Hof19] In Abbildung 3.1 werden die Größen verschiedener öffentlicher, kommerzieller und proprietärer Datenbanken und Fragmenträume verglichen. Dabei fällt auf, dass die proprietären Fragmenträume mit teilweise mehr als 10^{20} Molekülen mehr als acht Zehnerpotenzen größer sind als jede Datenbank oder kommerzieller Fragmentraum. Allein die Enumeration eines solchen Raumes muss bei heutiger Technik als unmöglich angesehen werden. Der Hersteller Seagate behauptet in seinem Firmenblog der erste zu sein, der in seiner Firmengeschichte insgesamt drei Zettabyte[Pau21] ($3 \cdot 10^{21}$) an Festplattenkapazität ausgeliefert hat. Im Vergleich zur Enumeration des größten proprietären Fragmentraums, dem GSK XXL (siehe Abbildung 3.1), entspricht das weit weniger als einem einzigen Bit Speicherplatz, das pro Molekül zur Verfügung steht.

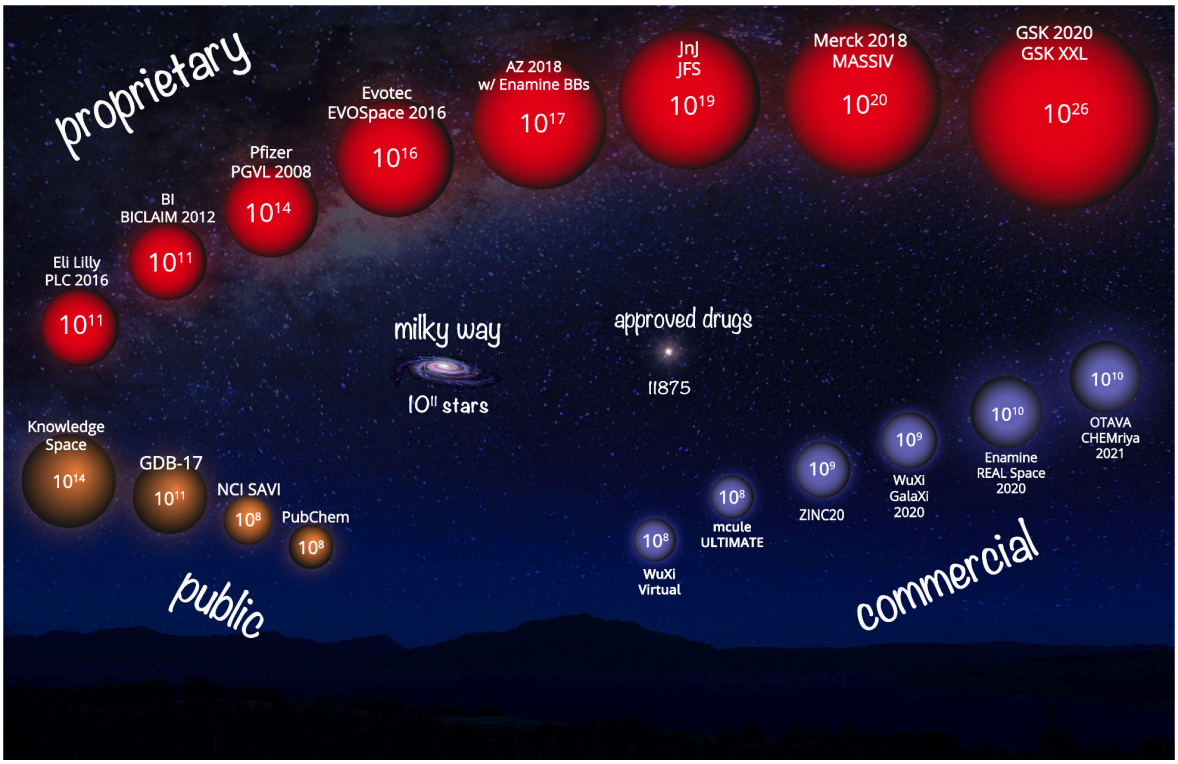


Abbildung 3.1: Einer Übersicht über die Größe kommerzieller, öffentlicher und proprietärer Fragmenträume und Datenbanken. Die Graphik ist eine aktualisierte Version der von Hoffmann und Gastreich[Hof19] publizierten Originalgraphik.

Bis 2012 wurde mit den generierten Datenbanken des chemischen Universums (GDB11,[Fin07] GDB13[Blu09] und GDB17[Rud12]) der Versuch unternommen, alle Moleküle mit bis zu 17 Schweratomen aufzuzählen. Insbesondere letztere, die GDB17 mit über 166 Milliarden Molekülen ist in ihrer Größe in den meisten Anwendungsszenarien unbenutzbar und nicht durchsuchbar und nur durch die Anwendung diverser Filterkriterien nicht noch mehrere Zehnerpotenzen größer. Die Analyse der enumerierten Moleküle hat zudem verdeutlicht, dass diese im Vergleich zu bestehenden Substanzbibliotheken viele neuartige Molekülgerüste[Bem96] enthält und vor allem deutlich mehr kugelförmige Molekülgerüste, die in existierenden Substanzbibliotheken unterrepräsentiert sind. Diese Aussage wurde mit der Vorstellung der ZINC20[Irw20] nochmals bestätigt. Der direkte Nutzen der Datenbanken des chemischen Universums für die Wirkstoffentwicklung ist jedoch gering, da die meisten Moleküle nicht chemisch verfügbar sind. Um den Zugewinn an Diversität von Molekülgerüsten der Datenbanken ausnutzen zu können, wurden Moleküle zunächst zufällig ausgewählt und im folgenden nach Diversität gefiltert.[Awa17]

Mit dem Aufkommen des sogenannten „Cloud-Computing“ bei dem kommerzielle Anbieter freie Rechenkapazitäten in eigens dafür geschaffenen Rechenzentren, vermieten, begann sich ein neuer Trend abzuzeichnen. Das stetige Wachstum der Substanzbibliotheken wurde nicht mehr als Problem gesehen. Anstatt diese auf eigenen Rechnern zu durchsuchen, wurden diese in der „Cloud“ oder auf sonstigen Clustern durchsucht.[Gre20, Lyu19] Insbesondere Lyu *et al.* [Lyu19] stellen in ihrer Studie fest, dass es wichtig gewesen sei, alle Moleküle der betrachteten Substanzbibliothek zu testen. Die Auswahl der zu testenden Substanzen wurde sowohl von Experten als auch automatisch durchgeführt. Darüber hinaus stellen Lyu *et al.* fest, dass die Wahrscheinlichkeit eine aktive Substanz zu finden in beiden Fällen ähnlich gering ist. Unabhängig davon wird die Möglichkeit, große Molekülmengen via Docking zu prozessieren, auch kommerziell angeboten.[Gre20] Dabei können allein die Kosten für das Docken in der Cloud von ~1.5 Milliarden Moleküle 10.000$ und mehr betragen.[McG20] Der

Trend, größtmögliche Datensätze linear zu durchsuchen, erinnert an das Aufkommen des sogenannten „high-throughput-screening“.[Cla20] Jedoch meint Clark[Cla20] es wäre eine Stärke der medizinischen Chemie, Kandidaten für Medikamente in virtuellen Substanzbibliotheken zu finden, die nicht nur nach der Wirksamkeit im Protein, sondern auch nach den sogenannten ADME-T Kriterien optimiert werden. Der Schritt dorthin zurück wäre beim jetzigen Wachstum virtuell verfügbarer Substanzen unausweichlich. Unabhängig von aller Euphorie, dass sich Berechnungen auf die aktuell größten enumerierten Bibliotheken skalieren lassen, sind und bleiben die Computermethoden, die die Wirksamkeit einer Substanz bewerten die größte Fehlerquelle.[Lyu19, Stu20]

Im Vergleich zu den Docking-Anwendungen ist die auf der Graph-Editierungsdistanz aufbauende Ähnlichkeitssuche, wie sie für die ZINC20[Irw20] angeboten wird die nachhaltigste Technologie, die auf Enumeration basiert. Diese Suche mit Namen SmallWorld[Irw20, Say20] benutzt eine 42TB große Datenbank von anonymen Graphen für das Traversieren und das Auffinden von ähnlichen Strukturen. Die Ergebnisdarstellung im Internetbrowser unterstützt die Sortierung nach diversen verschiedenen Fingerabdrücken, gibt aber keine Garantie auf Vollständigkeit. Im Vergleich zu bestehenden kommerziellen Fragmenträumen mag das noch auf Jahre hinreichend sein, im Vergleich zu den größten Fragmenträumen der Pharmafirmen ist der Ansatz jedoch nicht anwendbar, wie Abbildung 3.1 zeigt.

Zusammenfassend lässt sich feststellen, dass ein System benötigt wird, das auf virtueller Synthese erstellter Substanzbibliotheken ohne Enumeration beschreiben kann. Klassische und topologische Fragmenträume sind dafür zwei Beispiele.

3.2.2 Anwendungen von Fragmentraumsuchen

Fragmentraumsuchen unter Verwendung der maximalen gemeinsamen Teilstruktur finden Anwendung in diversen Fragestellungen der medizinischen Chemie. Um die Relevanz zu verdeutlichen kann ein beispielhaftes Anwendungsszenario in die Publikation einer neuen Methode inkludiert werden. Potentielle Anwendungsbeispiele können z.B. die Suchen nach ähnlichen Substanzen zu bekannten bioaktiven Molekülen sein. Durch die Verfügbarkeit virtueller Moleküle bei kommerziellen Anbieter kann noch eine weitere Anwendung beworben werden. Die Suche nach erwerbbaren make-on-demand Compounds für Anwendungen des computerunterstützten Wirkstoffentwurfs in kommerziellen Räumen bietet den Vorteil, dass gefundene Wirkstoffkandidaten nicht von den Anwendern synthetisiert werden müssen. Diese können ihre finanziellen und personellen Ressourcen somit auf das Testen fokussieren.

3.3 Bestehende Suchverfahren in kombinatorischen Fragmenträumen

Abseits der Vorteile einer möglichst kompakten Darstellung großer Molekülmengen haben Fragmenträume den Nachteil, dass die vorhanden Moleküle nicht trivial zugänglich sind. Wie soeben beschrieben, ist die Enumeration und Suche nicht mehr effizient durchführbar. Um dennoch Substanzen oder ähnliche Moleküle zu einer angefragten Substanz zu finden, ist es notwendig, spezielle Suchverfahren anzuwenden, die auf die Struktur der Fragmenträume optimiert sind. In den folgenden Unterkapiteln werden drei Verfahren aus der Literatur beschrieben:

3.3.1 FTrees-FS

Die sogenannten Feature Trees,[Rar98] repräsentieren molekulare Eigenschaften in einer Baumstruktur, die einem reduzierten Graphen des Moleküls entspricht. Diese molekularen Eigenschaftsbäume werden im Folgenden *FTrees* genannt. Die einzelnen Knoten in den FTrees stellen Ringsysteme der Moleküle, funktionelle Gruppen mit Pharmakophoreigenschaften, wie zum Beispiel Wasserstoffbrücken Donoren und Akzeptoren, und Ketten, dar. FTrees wurden als Deskriptor für die Ähnlichkeitssuche auf Molekülen entwickelt und ermöglichen es, diese mit ihrem eigenen Ähnlichkeitsmaß zu vergleichen. Sie sind dafür bekannt, dass auch Moleküle mit unterschiedlichen Gerüsten hohe Ähnlichkeiten erreichen können. Das kann insbesondere bei ligandbasierten Methoden im Wirkstoffentwurf vorteilhaft sein. Die Fähigkeit Moleküle mit unterschiedlichen Gerüsten in einer Ähnlichkeitssuche zu finden wird in der Literatur und im folgenden als *Scaffold Hopping* bezeichnet. Die Beschreibung durch reduzierte Graphen abstrahiert stark vom tatsächlichen Gerüst und ist für die Fähigkeit zum Scaffold Hopping bekannt. Dieser Vorteil ist häufig auch der größte Nachteil einer Ähnlichkeitssuche mit FTrees.

Der einzelne Vergleich zweier FTrees ist durch Ausnutzen der Baumstruktur des Deskriptor mit einem Verfahren der dynamischen Programmierung effizient lösbar.

Genau diese Baumstruktur des Deskriptors lässt sich ebenfalls ausnutzen, um die Ähnlichkeit in Fragmenträumen zu betrachten. Klassische Fragmenträume sind so aufgebaut, dass Fragmente unter Einhaltung der Verbindungsregeln zu Bäumen zusammengesetzt werden. Diese repräsentieren im Folgenden alle Moleküle des Fragmentraums. Durch die Darstellung der Fragmente als FTrees ermöglicht ein weiteres Schema dynamischer Programmierung eine effiziente Ähnlichkeitssuche in Fragmenträumen.[Rar01] Der Algorithmus ist unter dem Namen FTrees-FS[Rar01] bekannt. Während der Suche wird jeder FTree-Deskriptor der Fragmente eine von der Anfrage abhängige Anzahl oft betrachtet. Anfragemoleküle sind in der Regel jedoch klein. Daher kann dieser Faktor für die Laufzeitabschätzung als konstant angesehen werden. Gleiches gilt für die einzelne Ähnlichkeitsberechnung von zwei Deskriptoren. Als nächstes fließen die Anzahl der gesuchten Ergebnisse und die Anzahl an Fragmenten des Raums in die Laufzeitbetrachtung mit ein. Die Anzahl der durch den Fragmentraum repräsentierten Moleküle hingegen hat keinen Einfluss. Die wichtigste Eigenschaft dieser und anderer Fragmentraummethoden ist die unabhängige Betrachtung von Teilergebnissen auf anderen Fragmenten. Hier bedeutet es, dass die Anzahl, wie häufig ein Deskriptor eines FTrees während einer Suche betrachtet wird, nicht von der Struktur des durchsuchten Raums abhängt. Genannte Anzahl darf hingegen von der Anfrage abhängen.

3.3.2 SMARTS-Fs

Die SMARTS-Fs[Ehr12, Ehr13] Methode ist ein spezifischer Algorithmus zur Suche molekularer Muster in klassischen Fragmenträumen. Die Idee hinter SMARTS-Fs ist die Unterteilung des gesuchten Musters in alle möglichen Substrukturkombinationen, die die Fragmentaufteilung eines getroffenen Moleküls darstellen können. An den geschnittenen Bindungen in der Anfrage werden, vergleichbar zur Generierung von Fragmenten aus Molekülen, Verbindungsknoten eingefügt. Diese werden im Folgenden *SMARTS-Linker* genannt. Damit die Suche erfolgreich sein kann, müssen die SMARTS-Linker auf die Linker in den Fragmenten abgebildet werden. Die einzelnen Ergebnisse der Treffer werden über Baumstrukturen repräsentiert. Eine solche Baumstruktur stellt jeweils eine mögliche Unterteilung der Anfrage in ihre Substrukturen dar. In ihren Knoten enthält die Baumstruktur jeweils Listen der getroffenen Fragmente. Um SMARTS Rekursion zu unterstützen werden die Baumstrukturen der getroffenen Moleküle verwendet, um einen abgeleiteten Fragmentraum zu erzeugen. In diesem abgeleiteten Raum wird nun die Substruktursuche der rekursiven Umgebung fortgesetzt. Das geschieht

für alle rekursiven SMARTS Teilausdrücke des initialen Musters. Anschließend werden die Ergebnisbäume des SMARTS Ausdruck ohne Rekursion, mit den Ergebnisbäumen der rekursiven Ausdrucksteilen überlagert und unifiziert. Abschließend werden daraus Fragmente eines neuen Ergebnisfragmentraums erzeugt. Dieser kann bei Bedarf enumeriert werden, um alle Moleküle zu erzeugen, die vom gesuchten SMARTS Ausdruck getroffen werden.

3.3.3 SpaceLight

Der SpaceLight[Bel20] Ansatz ist eine Methode, die fingerabdruckbasierte Ähnlichkeitsberechnung in kombinatorischen Räumen ermöglicht. Die Methode benötigt allerdings im Vergleich zu FTrees-FS und SMARTS-Fs eine andere Darstellung der Fragmenträume. Für SpaceLight wurden sogenannte *topologische Fragmenträume* eingeführt.[Bel20] Topologische Fragmenträume bestehen aus Fragmenten und einer Vielzahl von Topologien. Eine Topologie stellt jeweils eine Anordnung dar, wie Fragmente zu Molekülen des topologischen Fragmentraums verbunden werden. Um Ringschlüsse zu ermöglichen sind Topologiegraphen Multigraphen, bei denen zwei Knoten über mehrere Kanten miteinander verbunden werden können. Die Kanten im Topologiegraphen repräsentieren die Verbindungsregeln klassischer Fragmenträume. Für die Fragmente ist es somit relevant, in welchen Topologieknoten sie in diesen vorkommen können. Alle Fragmente eines Topologieknotens haben dieselbe Anzahl an Linkern und Linktypen, die Verbindungen zu anderen Topologieknoten repräsentieren. Dieser Aufbau ermöglicht nicht nur die Abbildung von Ringschlüssen und Ringschlussreaktionen, sondern generell die Behandlung von Makrozyklen. Beides ist aufgrund der Baumstruktur klassischer Fragmenträume nicht möglich. Dafür ist in klassischen Fragmenträumen die Anzahl an Fragmenten, aus denen die Moleküle bestehen, theoretisch unbeschränkt. Beide Arten von Fragmenträumen werden auf Basis von Reaktionswissen und virtueller Synthese erstellt. In der Praxis ergeben sich dadurch ähnliche Beschränkungen für die Anzahl an Fragmenten, aus denen die Moleküle der Räume bestehen.

Eine Ähnlichkeitssuche in einem topologischen Fragmentraum basiert auf den folgenden Schritten:[Bel20] Das Anfragemolekül wird entsprechend aller möglichen Topologien des topologischen Fragmentraums partitioniert. Dieser Schritt wird in [Bel20] als eine Unterteilung des Moleküls in eine Menge zusammenhängender disjunkter molekularer Substrukturen beschrieben, deren Vereinigung wiederum das Molekül ergibt. Im Partitionierungsschritt werden alle Partitionierungen erzeugt, die in den Atomzahlen der Substrukturen und deren Verbindungen untereinander kompatibel zu mindestens einer Topologie des Raumes sind. Die Anzahl der erzeugten Partitionen hängt somit von der Größe des Moleküls als auch von den Topologiegraphen des topologischen Fragmentraums ab. Insbesondere bei großen Ringsystemen kann die Erzeugung aller Partitionierungen einen großen Laufzeitanteil ausmachen. Im Vergleich zu FTrees-FS und SMARTS-Fs werden bei der Partitionierung auch Ringbindungen geschnitten. Für den Fingerabdruckvergleich mit den Fragmenten werden die Partitionierungen einzelnen Topologien zugeordnet. Anschließend können die Ergebnisse innerhalb der Topologien unabhängig voneinander zu den ähnlichsten Molekülen des Raumes kombiniert werden.

Im Vergleich der existierenden Ansätze bleibt festzustellen, dass sie auf die jeweiligen Fragmenträume, auf denen sie angewendet werden, optimiert sind. Darüber hinaus unterscheiden sich auch die laufzeitintensiven Schritte in den Algorithmen. Bei FTrees-FS und SMARTS-FS ist die Berechnung der Ähnlichkeit beziehungsweise der Kompatibilität eines Teils der Anfrage mit den Fragmenten relativ gesehen teuer. Das gilt insbesondere im Vergleich zu SpaceLight, wo die Berechnung der Kompatibilität über die Fingerabdruckähnlichkeit von Partitionen der Anfrage und Fragmenten relativ günstig ist. Dafür wird bei SpaceLight die Anfrage in deutlich mehr Teile unterteilt, für die eine Ähnlichkeit zu den Fragmenten bestimmt werden muss, als bei FTrees-FS und SpaceMACS. FTrees-FS und SpaceMACS erzeugen für jede azyklische Bindung maximal zwei

Anfrageunterteilungen. SpaceLight hingegen berechnet alle möglichen kompatiblen Partitionierungen für alle Topologiegraphen des durchsuchten Raums. Die genaue Anzahl an betrachteten Partitionierungen wurde noch nicht näherliegend untersucht.

3.4 Suche maximaler gemeinsamer Teilstrukturen in Fragmenträumen

Die beschriebenen Methoden zur Suche in Fragmenträumen ermöglichen eine Ähnlichkeitssuche, bei der es schwierig ist, Struktureigenschaften der Moleküle in die Ähnlichkeitssuche mit einzubeziehen. SMARTS-Fs bildet zwar Struktureigenschaften auf atomarer Ebene ab, ist aber eine Substruktursuche, die keine Ähnlichkeitssuche im eigentlichen Sinn unterstützt. Die verwendeten Fingerabdrücke der SpaceLight Methode stellen mit ihren Bits auch Substrukturen dar, allerdings ist die Suche nach speziellen Substrukturen nicht möglich. Dieses Problem wurde mit der Entwicklung der SpaceMACS Methode erstmals angegangen, so dass eine Ähnlichkeitssuche auf Basis zusammenhängender gemeinsamer Teilstrukturen realisiert wird. Dieses Kapitel behandelt die Suche nach maximalen gemeinsamen zusammenhängenden induzierten Teilstrukturen, wie sie in [D2] beschrieben wird.

Fragmenträume zeichnen sich dadurch aus, dass sie große Mengen unterschiedlicher Moleküle anhand ihrer Fragmentbausteine beschreiben können. Die Effizienz der Beschreibung steigt, je mehr Fragmente miteinander kombiniert werden, um die Produkte zu bauen. Für Algorithmen, die auf Fragmenträumen arbeiten, hat das jedoch zur Folge, dass die Berechnungen über Fragmentgrenzen unabhängig voneinander sein müssen. Die Erkennung von Fragmentgrenzen kann *dynamisch* erfolgen, so dass beim Auftreffen von Linkern während der Berechnung auf bereits berechnete Ergebnisse zurückgegriffen wird oder *statisch* durch Unterteilung in die Anfrage modelliert werden. Die FTrees-FS Methode betrachtet Fragmentgrenzen dynamisch. Das wird durch die Verwendung der dynamischen Programmierung ersichtlich, bei der die Anfrage auf die Feature-Tree Deskriptoren der Fragmente abgebildet wird und beim Auftreffen eines Knotens, der einen Linker repräsentiert, auf die besten nachfolgenden Teilergebnisse zugegriffen wird. SpaceLight und SMARTS-FS hingegen modellieren die Fragmentgrenzen statisch. Bei SpaceLight kann das aus der Partitionierung der Anfrage und die Bestimmung der Passung auf die Topologien des topologischen Fragmentraums gefolgert werden. Im SMARTS-Fs werden in die Unterteilung der Anfrage explizite SMARTS-Linker eingebaut, die nur zu Linkern der Fragmente kompatibel sind. Hier ist die statische Modellierung der Fragmentgrenzen noch deutlicher zu erkennen als bei SpaceLight.

Für die Suche maximaler gemeinsamer Teilstrukturen in Fragmenträumen ist ein Ziel, Fragmentgrenzen dynamisch zu erkennen, da die einzelne MCS Berechnung als ein laufzeitintensiver und geschwindigkeitsbestimmender Schritt angesehen wird. Um das zu ermöglichen, wird das zu lösende MCS Problem angepasst und auf die Anwendung in klassischen Fragmenträumen optimiert.

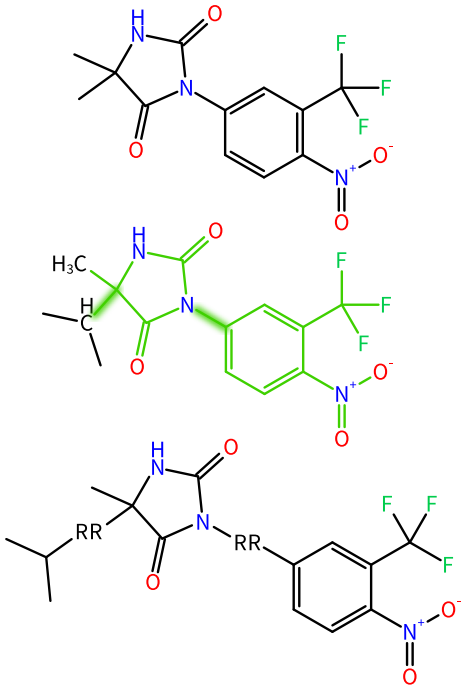


Abbildung 3.2: Nilutamide[Drua] (Oben) und eines der ähnlichsten Moleküle (Mitte) aus dem REAL Space[Ena21]. Die gemeinsame Teilstruktur und die Fragmentgrenzen sind in der mittleren Struktur grün hervorgehoben. Unten sind die Fragmente dargestellt, aus denen sich die mittlere Struktur zusammensetzt. Die Graphik wurde aus [D2] unverändert übernommen.

Abbildung 3.2 lässt am Beispiel von Nilutamide[Drua] einige Eigenschaften der entwickelten MCS Suche erkennen. Fragmentgrenzen laufen immer über azyklische Bindungen, weshalb das Problem dahingehend eingeschränkt wird, dass azyklische Bindungen nur auf azyklische Bindungen abgebildet werden. Abbildungen zyklischer Bindungen werden analog dazu eingeschränkt. Diese Einschränkung ist einer der Schlüssel für die Effizienz des Ansatzes. Außerdem ergibt sie sich aus der Zielsetzung einer dynamischen Fragmentgrenzenerkennung. Eine korrekte Handhabung des uneingeschränkt zusammenhängenden MCS in Fragmenten bedingt eine statische Detektion von Fragmentgrenzen, sofern eine zyklische Bindung der Anfrage über eine Fragmentgrenze verlaufen können soll.

Eine Anfrage an einen Fragmentraum mit dem SpaceMACS Algorithmus besteht aus vier aufeinander folgenden Schritten.

1 Unterteilung der Anfrage in Teilstrukturen und die Berechnung einer Halbordnung als Prozessierungsreihenfolge.

2 Berechnung aller MCS-Abbildungen zwischen den molekularen Substrukturen und den Fragmenten des Fragmentraums.

3 Enumeration von Ergebnissen mit den Teilergebnissen aus Schritt 2.

4 Berechnen von MCS-Abbildungen der gesamten Anfrage mit allen Fragmenten, wobei Linker nicht abgebildet werden.

Die ersten befassen sich damit, Strukturen des Raums zu identifizieren, bei denen sich die gemeinsame Teilstruktur über mindestens zwei Fragmente erschließt.

Der SpaceMACS Algorithmus unterteilt die Anfrage in molekulare Substrukturen, bei der jede azyklische Bindung individuell geschnitten wird. Anschließend werden die Substrukturen gemäß einer Halbordnung auf die Fragmente des Raumes abgebildet. Die Halbordnung wird benötigt, da Zwischenergebnisse von Substrukturen auf jene von kleineren Strukturen angewiesen sein können. Zudem ermöglicht die Realisierung als Halbordnung einen Begriff von Unabhängigkeit der die Parallelisierung vereinfacht.

Die gewählte Modellierung garantiert, dass im Abbildungsschritt die Struktur aus dem kombinatorischen Raum gefunden werden kann, bei der die gemeinsame Teilstruktur die größte Anzahl an Atomen umfasst. Damit die in der Praxis relevantere Frage nach den nicht optimalen Ergebnissen so weit wie möglich korrekt vom Algorithmus beantwortet werden kann, wird das tatsächlich berechnete MCS Problem auf eine stark gefilterte Variante des „alle maximalen gewichteten gemeinsamen Teilstrukturen“ abgeändert. Zur der effizienten Umsetzung dieses Problems hilft die Möglichkeit der nachträglichen Anpassung der Gewichtung von MCS Ergebnissen, wie es der RIMACS[D1] Algorithmus ermöglicht.

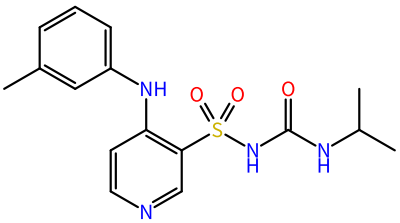
Die Enumeration der Ergebnisse stellt das Herzstück des SpaceMACS Algorithmus dar. Die Teilergebnisse werden gruppiert, sortiert und enumeriert. Während der Enumeration werden ausgehend von einem „besten“ Kandidaten immer weiter die besten nachfolgenden Ergebnisse in absteigender Sortierung hinzugefügt. Die Sortierung garantiert dabei, dass kleine Anzahlen von Ergebnissen innerhalb kürzester Zeit berechnet werden können, die Laufzeit aber durch die Enumeration auch von der Anzahl der Ergebnisse abhängt.

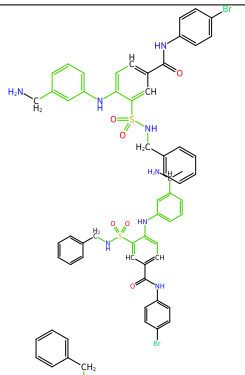
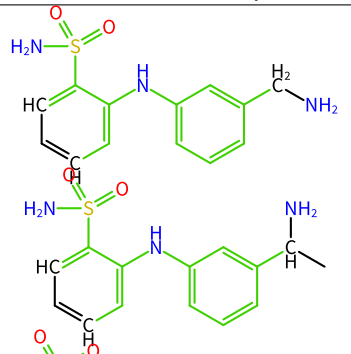
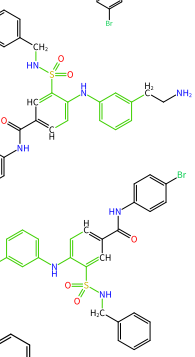
Die abschließende Berechnung der Strukturen bei denen der MCS vollständig in einem Fragment liegt, wird als Sonderfall nachgelagert betrachtet. Insbesondere gelten hier triviale Schranken der erreichbaren Güte der Ergebnisse, die sich aus der Atomanzahl des Fragments ableiten lässt.

Ein wichtiger Aspekt der Methode offenbart sich in den unterstützen Ähnlichkeitsmaßen für den MCS. Vom Algorithmus nativ unterstützt ist die Maximierung der MCS Abbildung. Dieses Maß wird im Folgenden *MCS-Size* genannt. Da das aber zu als unähnlich empfundenen ähnlichsten Kandidaten führen kann, wird ein weiteres Ähnlichkeitsmaß mit dem Namen *MCS-Similarity* unterstützt. Dabei werden auch die Größen der Anfrage und des Ergebnismoleküls für den Ähnlichkeitswert betrachtet. Das kann auch in der Gegenüberstellung der besten fünf Ergebnisse aus dem GalaXi[WuX20] in Tabelle 3.2 betrachtet werden. Die gemeinsame Teilstruktur zur Anfrage Torasemid[Drub] ist jeweils in grün hervorgehoben.

Tabelle 3.2: Vergleich der ähnlichsten Moleküle zur Anfrage Torasemid[Drub] auf dem GalaXi[WuX20]. Links sind die ähnlichsten Moleküle nach MCS-Size, bei der die Sortierung nur durch die Anzahl abgebildeter Knoten beeinflusst wird. Rechts werden die Atomzahlen von Anfrage und Resultatmolekül mit einbezogen (MCS-Similarity). Die einzelnen Moleküldarstellungen sind aus [D2] übernommen und neu arrangiert.

Anfrage:



Rang	MCS-Size	MCS-Similarity
1		
2		
3		
4		
5		

Neben der Güte der gefundenen Ergebnisse ist die benötigte Rechenzeit für eine Anfrage ein wichtiges Maß. Der SpaceMACS Algorithmus skaliert gut bis zu ca. 16 Threads. Das Potenzial des Algorithmus wird in Abbildung 3.3 dargestellt. Dabei wurden 100 von Lessel und Lemmen[Les19] ausgewählte Moleküle als Anfrage auf dem REAL Space[Ena21] verwendet. Für jeden der drei wichtigen Schritte des Algorithmus ist die durchschnittliche Laufzeit dargestellt. Dabei fällt auf, dass die Laufzeit für die Berechnung der MCS Ergebnisse unabhängig von der Anzahl der angefragten Ergebnisse ist. Allerdings hat diese Anzahl einen bedeutsamen Einfluss auf die Laufzeit der Ergebnisenumeration und der Berechnung der Ergebnisse, bei denen der MCS in einem einzigen Fragment enthalten ist. Im günstigsten Fall sind auf diese Art durchschnittliche Anfragedauern von unter 10s möglich. Um 10.000 Ergebnisse anzufragen, reichen immer noch ca. 17,5s im Durchschnitt. Damit liegen die Laufzeiten im selben Bereich, der für Anfragen auf dem von der ZINC angebotenen SmallWorld[Irw20] Algorithmus benötigt wird. Für SmallWorld sind die Anforderungen an die Hardware aber signifikant größer. Von den Anforderungen vergleichbar ist da der SpaceLight Algorithmus, der bei Verwendung von drei Threads

auf einer älteren Version des REAL Space weniger als 10s im Durchschnitt benötigt. Die Anzahl berechneter Ergebnisse ist dabei leider nicht genannt.

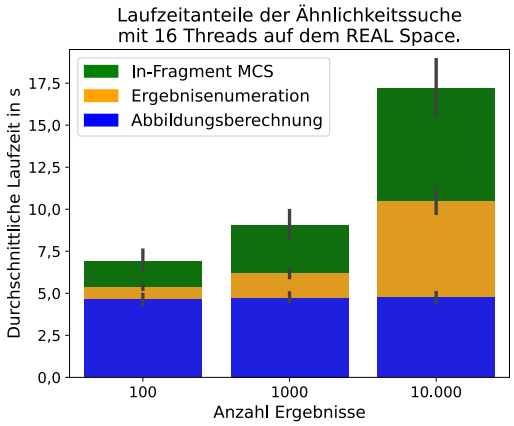


Abbildung 3.3: Darstellung der durchschnittlichen Laufzeiten der einzelnen Algorithmusschritte auf dem REAL Space[Ena21] bei der Nutzung von 16 Threads. Als Anfragen wurden die 100 von Lessel und Lemmen[Les19] ausgewählten Substanzen verwendet. Die Abbildung wurde aus [D2] übernommen und ins Deutsche übersetzt.

3.4.1 Validierung von Fragmentraummethoden

Für die Validierung der SpaceMACS Methode wurden drei Experimente durchgeführt, die bestätigen, dass die gelieferten Ergebnisse in den meisten Fällen mit denen einer exakten Suche übereinstimmen. Dazu wurden jeweils die ähnlichsten 5000 Moleküle, die aus einem kleinen Fragmentraum stammen, mit den Ergebnisses aus der enumerierten Molekülmenge verglichen. Weiterhin wurde geprüft, ob alle Substanzen aus den enumerierten Räumen auch gefunden werden können und die Konsistenz der Ergebnisse evaluiert, wenn verschiedene Anzahlen an Ergebnissen angefragt werden oder Parameter des Algorithmus variieren. Diese Validierung folgt denen der existierenden Fragmentraummethoden.

Der SpaceLight Ansatz ist die am besten validierte Methode der bestehenden Technologien. Für die Validierung wurden drei kleine Fragmenträume mit bis zu 22.000 dargestellten Molekülen verwendet. Dabei wurde eine Korrelation der vollständigen Ergebnisreihenfolge bestimmt, wenn alle Moleküle mit der SpaceLight Methode oder einem klassischen Ansatz nach absteigender Fingerabdruckähnlichkeit zu einer Anfrage sortiert werden. Für diese Analyse wurden jeweils 500 Anfragen verwendet, die aus dem Raum selbst stammen und 500 externe Anfragen.

Die Räume und internen Anfragen an diese liegen der Publikation bei, die externen Anfragen sind nicht verfügbar. Das ist ähnlich zur Validierung von SpaceMACS, bei der die enumerierten Räume und der KnowledgeSpace[Bio19] verfügbar sind. Die kommerziellen Räume sind intellektuelles Eigentum der make-on-demand Compound Anbieter und können nicht der Öffentlichkeit zur Verfügung gestellt werden. Im Vergleich dazu wurde für SMARTS-FS und FTrees-FS nur gezeigt, dass die Methoden plausible Ergebnisse bei ausgewählten Anfragen liefern kann ohne eine tiefer greifende Validierungsstrategie zu verfolgen.

FTrees-FS ist die älteste und bislang relevanteste Fragmentraumtechnologie. Das leitet sich daraus ab, dass sie seit Jahren vermarktet wird.[Bio] SpaceLight und SpaceMACS werden außerhalb von Demonstrationsanwendungen noch nicht vermarktet, sollen aber zusammen mit FTrees kommerziell angeboten werden. Davon weit entfernt ist der SMARTS-FS Ansatz. Zu den Publikationen wurden Ergebnisse von internen Anwendungen genannt, die aber nicht mit publiziert wurde, so dass sie nur wenig Beachtung finden kann.

3.5 Vergleich der Fragmentraummethoden

3.5.1 Ergebnisvergleich der Fragmentraummethoden

Der SpaceMACS Algorithmus zeichnet sich dadurch aus, dass dieser atomare Eigenschaften für die Berechnung des MCS berücksichtigt. FTrees-FS und SpaceLight sind komplementäre Verfahren mit entsprechenden Ähnlichkeitsmaßen. In [D2] wurden jeweils 50.000, insgesamt 150.000, ähnlichste Moleküle zu einer Anfrage aus den drei kommerziellen Fragmenträumen [OTA21, WuX20, Ena21] extrahiert. Die Gemeinsamkeiten der methodenspezifischen ähnlichsten Molekülmengen sind in Abbildung 3.4 dargestellt. Sie machen deutlich, dass jede Methode einen anderen Aspekt molekularer Ähnlichkeit abdeckt und die Räume so groß und divers sind, dass es nur zu geringfügigen Überschneidungen kommt. Auch in Fragmenträumen wird gelten, dass das Ähnlichkeitsmaß der Wahl vom Anwendungsszenario abhängt, wie es schon für klassische Ähnlichkeitsmaße auf Molekülmengen gilt.[She02]

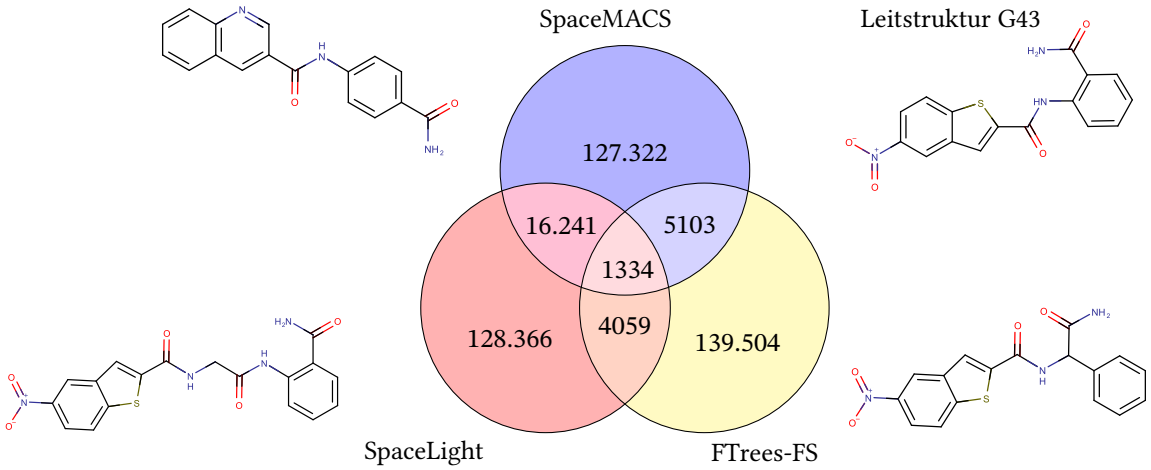


Abbildung 3.4: Überschneidung der 150.000 ähnlichsten Moleküle zur Leitstruktur G43[D2, Nij21] in den 3 kommerziellen Fragmenträumen, die mit den drei Methoden, FTrees-FS, SpaceLight und SpaceMACS extrahiert wurden. Die SpaceMACS Ergebnisse weisen große identische Atomanordnungen in den Kernregionen zu der Anfrage auf (blau). Die SpaceLight Ergebnisse sind ebenfalls strukturell ähnlich, können aber auch duplizierte Regionen enthalten (rot). Abschließend FTrees-FS Resultate können die strukturell größten Abweichungen tragen, da für die eine ähnliche Anordnung der funktionellen Gruppen wichtig ist (gelb). Diese Abbildung ist aus [D2] übernommen.

Insgesamt verdeutlicht Abbildung 3.4, dass SpaceMACS die Anfragemöglichkeiten an kombinatorische Fragmenträume erweitert und bisher nicht unterstützte Anfragearten ermöglicht und somit einen Mehrwert zu dem bisherigen Portfolio an Fragmentraumsuchen bildet.

3.5.2 Methodischer Vergleich der Fragmentraummethoden

Der SpaceMACS Algorithmus hat das Ziel, exakte Ergebnisse zu liefern, die eine lineare Suche auf den gegebenen Fragmenträumen liefern würde. Der kombinatorische Aufbau der Fragmenträume und die unterstützten Ähnlichkeitsmaße lassen genau das nicht immer zu, approximieren das Verhalten jedoch hinreichend.Der SpaceMACS Algorithmus unterstützt ähnlichkeitsbasierte Resultatfilter innerhalb der Enumeration der Ergebnisse. Darüber hinaus besteht für den Nutzer aber keine Möglichkeit einen Ähnlichkeitszielwert anzugeben. In Bezug auf die Exaktheit ist das auch nicht sinnvoll, da der MCS grundsätzlich eine Maximierung erfordert.

Sollten suboptimale Ergebnisse mit einem Ähnlichkeitszielwert gewünscht sein, bestünde die Gefahr, dass SpaceMACS falsche MCS Ergebnisse liefert, da diese grundsätzlich aus den Teilergebnissen übernommen werden. Alternative, größere Abbildungen mit den selben Fragmenten, aber einer anderen Aufteilung der Abbildung könnten oberhalb des Grenzwerts liegen, so dass suboptimale und damit falsche Ergebnisse für Resultatmoleküle berechnet würden. Deshalb unterstützt SpaceMACS im Vergleich zu FTrees-FS keinen benutzerdefinierten Ähnlichkeitszielwert. Der von SpaceMACS berechnete MCS ist zusammenhängend, so dass die Ähnlichkeiten in einzelnen Fragmenten hohe Ähnlichkeitsspannen aufweisen können. FTrees-FS nutzt bei dem Ähnlichkeitszielwert die Heuristik,[Rar01] dass die Ähnlichkeit über alle Fragmente innerhalb eines gegebenen Intervalls der Zielähnlichkeit nächstmöglich sein sollte. Das ist insbesondere für einen zusammenhängenden MCS nicht anwendbar.

Die Nutzung von Fingerabdrücken auf Partitionierungen des Eingabemoleküls, wie von SpaceLight verwendet, ermöglicht ebenfalls eine heuristische Anwendung von Ähnlichkeitszielwerten. Alle drei Verfahren haben das Problem, dass bei der Berechnung suboptimaler Ergebnisse Moleküle aufgrund ungünstiger Abbildungen schlechtere, also falsche Ähnlichkeitswerte erhalten können. Die abstrakten Ähnlichkeitsmaße von FTrees-FS und SpaceLight sind für Anwender in der Regel nicht einfach nachvollziehbar. Ein MCS-Ähnlichkeitswert, der zudem eine Anzahl gemeinsamer Atome nennt, ist deutlich einfacher als falsch zu erkennen. SpaceMACS ist die erste ähnlichkeitsbasierte Suchmethode in kombinatorischen Fragmenträumen, die über molekulare Anfragen hinausgeht. Im Vergleich publizierter Methoden zur Exploration kombinatorischer Fragmenträume weist SpaceMACS die Größten Ähnlichkeiten zum SMARTS-Fs[Ehr12, Ehr13] Algorithmus auf. SMARTS-Fs unterstützt im Vergleich zu SpaceMACS aber kein Ähnlichkeitsmaß und bietet alleinig eine Substruktursuche an. Diese unterstützt auch SMARTS Rekursionen und ist nicht dahingehend beschränkt, dass Bindungen explizit in zyklisch und azyklisch unterteilt werden müssen. Diese Unterschiede spiegeln sich auch in den verwendeten Ansätzen und der dynamischen beziehungsweise statischen Erkennung der Fragmentgrenzen wider. In dem Punkt ist SMARTS-Fs ein algorithmisches Bindeglied von SpaceLight und SpaceMACS. Das Ergebnis einer SMARTS-Fs Suchanfrage ist ein dedizierter Fragmentraum, in dem jedes Molekül vom angefragten SMARTS getroffen wird. Aus dem Raum können Moleküle enumeriert werden, oder dieser kann mit weiteren Methoden untersucht werden. Sofern es sich um einmalige Anfragen an die Fragmenträume handelt und die Beschränkung der Zykleneigenschaft auf Bindungen und das Verbot von SMARTS Rekursion keine Einschränkung darstellen, ist es vorteilhaft, diese mit SpaceMACS im Substrukturmodus durchzuführen. Unabhängig davon ist kein Programm, dass den SMARTS-Fs Algorithmus beinhaltet Teil der NAOMI ChemBioSuite(https://uhh.de/naomi). SpaceLight und SpaceMACS sind als Anwendung für Windows, Linux oder MAC innerhalb der NAOMI ChemBioSuite verfügbar.

3.6 Zusammenfassung und Ausblick

Die SpaceMACS Methode beschränkt sich darauf, dass die gefundene Substruktur einerseits zusammenhängend ist und andererseits die Ringeigenschaft aller Bindungen berücksichtigt. Gegenüber anderen Ähnlichkeitsmaßen können diese beiden Restriktionen ein Nachteil sein. Sie sind jedoch notwendig, da diese aus dem Modell stammen und SpaceMACS die Möglichkeit geben, als exakte Methode angesehen zu werden. In der Validierung wurden kleinere Probleme mit der Ergebniskonsistenz über verschiedene Parameterwerte gefunden. Diese beziehen sich insbesondere auf das MCS-Similarity Ähnlichkeitsmaß, das nicht nativ vom Modell unterstützt wird und im geringen Maß auch auf das nativ unterstützte MCS-Size Ähnlichkeitsmaß. Im letzteren Fall basieren die Fehler auf den durch eine zwischenzeitliche Enumeration von Duplikaten nicht expandierten Teilergebnissen. Dieser Effekt kann bei MCS-Similarity noch verstärkt auftreten.

Perspektivisch kann die strenge Auslegung des Modells innerhalb von SpaceMACS reduziert werden: Azyklische Bindungen der Anfrage könnten zu allen Bindungen der Fragmente kompatibel werden oder es wird nur von Bindungen zu Linkern gefordert, dass diese auf azyklische Bindungen der Anfrage abgebildet werden. Dadurch würde das Ergebnis heuristisch, könnte aber den Wünschen von Anwendern besser entsprechen. Weitere Möglichkeiten den Algorithmus anzupassen bestehen darin, auch topologisch eingeschränkte nichtzusammenhängende MCS-Resultate zu erlauben. Eine topologische Einschränkung ist notwendig, da die Lokalität von Abbildungen über Fragmentgrenzen hinweg nicht verletzt werden darf. Für die Qualität der Ergebnisse ergibt sich der Vorteil, dass einzelne Unterschiede an zentralen Stellen übersprungen werden können und sich deutlich bessere Ähnlichkeitswerte ergeben können, die sich in den Anforderungen der Anwender widerspiegeln.

Bezüglich der Anwendung bietet es sich an, weitere Filtermöglichkeiten in die Enumeration der Ergebnisse zu integrieren. Bisher können Ergebnisse anhand simpler Größeneigenschaften der Abbildung, des Ergebnis und des Resultatmoleküls gefiltert werden. Die Betrachtung weiterer additiver molekulare Deskriptoren, wie z.B. Molekulargewicht oder die Anzahl von Wasserstoffbrückendonoren und Akzeptoren, ist möglich.

Eine weitere Möglichkeit zukünftiger Änderungen am Algorithmus ist die Fokussierung auf Molekülgerüste. Die SpaceMACS Suche könnte darauf eingeschränkt werden, dass alle Ergebnisse ein vorgegebenes Murckow-Gerüst[Bem96] haben müssen. Eine solche Einschränkung ließe sich auch in die Aufbereitung der Fragmenträume und die Ergebnisenumeration integrieren und wurde in einem Projekt von den Studierenden Christoph Noack und Katharina Kaufmann demonstriert.

Bezüglich der Anwendung gibt es zwei primäre Ansatzpunkte die Effizienz zu verbessern. Es kann sowohl versucht versucht werden die Parallelisierung des Algorithmus oder den Speicherverbrauch der Molekülдarstellung zu optimieren. Ein viel größeres Potenzial zur Effizienzsteigerung hat allerdings eine verbesserte Beschreibung zuverlässiger Mehrkomponentenreaktionen in Fragmenträumen.

4 Vergleich chemischer Muster

4.1 Motivation

Wie in den bisherigen Kapiteln diskutiert, werden in der Chemieinformatik im Allgemeinen und im Speziellen vor allem im Bereich der Wirkstoffentwicklung regelmäßig große Molekülenmengen auf Wirksamkeit gegen Zielproteine getestet.[Bro09] Für einen Wirkstoffkandidaten eines oral verfügbaren Medikaments gelten strenge Richtlinien, damit dieser überhaupt zugelassen werden kann. Ein Teil dieser Kriterien wird durch die ADME-T Kriterien (Aufnahme (A), Verteilung (D), Metabolismus (M), Ausscheidung (E), Toxizität (T)) zusammengefasst. Sie beschreiben den Werdegang eines Medikaments von der Aufnahme in den Körper, zum Beispiel durch Einnahme der Tablette, bis zur Ausscheidung nach Abbau durch z.B. die Leber. Für Wirkstoffkandidaten gibt es verschiedene einfach anwendbare Regelwerke, wie z.B. Lipinskis Regel der Fünf,[Lip97] die das Ziel haben, die große Anzahl an möglichen Molekülen auf sinnvolle und vielversprechende Kandidaten zu reduzieren. Über Regeln, die auf einfachen Eigenschaften basieren, gibt es weitere auf Erfahrung basierende Regelwerke, die unerwünschte Interaktionen eines Wirkstoffkandidaten vermeiden sollen. Große Popularität erlangten diese Regelwerke erstmalig unter dem Namen PAINS.[Bae10] Zusammen mit der Publikation wurden über 450 chemische Substrukturmuster veröffentlicht, die Moleküle und funktionelle Gruppen beschreiben, die in diversen Versuchsreihen unerwünschte Interaktionen oder Wirkungen gezeigt hatten. Das Spektrum unerwünschter Interaktionen in den Regelwerken reicht von falsch angezeigter Aktivität im chemischen Assay über allgemeine Reaktivität bis hin zu gefährlichen Nebenwirkungen am z.B. hERG-Kaliumkanal im menschlichen Herzen.[Bae10, Bla05, Bre08b, Bru12, Gau17, Han99, Pea06, Sus12]

Häufig sind Substruktureigenschaften für die aufgezählten Wirkungen verantwortlich, die wiederum in Form von chemischen Mustern einfach angewendet werden können. Dabei hat sich die SMARTS-Sprache[Day] als Quasi-Standard gegenüber anderen Sprachen, wie SLN[Ash97] oder MQL[Pro07] durchgesetzt. Die Verfügbarkeit des gesammelten Wissens kommt mit einer Einschränkung: Chemische Assays sind untereinander schwer zu vergleichen, was insbesondere jene chemischen Muster beeinflusst, die eine unspezifische Interaktion mit dem Assay anzeigen sollen. Weitere zentrale Punkte sind die Zielsetzungen, die bei der Erstellung der jeweiligen Regelwerke verfolgt wurden und auch die Tatsache, dass einige auf anderen aufbauen. Die Komplexität aller Sprachen für chemische Muster hat zur Folge, dass ein algorithmischer Vergleich und auch eine kanonische Form bisher nicht vorhanden sind. Für eine abschließende Bewertung der einzelnen Regelwerke und einen sinnvollen Vergleich der enthaltenden Muster sowie der Intentionen der Autoren ist ein einfach zugänglicher Vergleich chemischer Muster unabdingbar.

Analog zu Molekülen werden chemische Muster als Graphen dargestellt. Daher ist es naheliegend für den Vergleich von chemischen Mustern und einer möglichen Ähnlichkeitsbewertung einen Algorithmus zu entwickeln, der eine maximale gemeinsame Teilstruktur auf den Graphen der internen Repräsentation der Muster berechnet.

4.1.1 Chemische Muster in der SMARTS-Sprache

SMARTS ist eine Beschreibungssprache für chemische Muster. Grundsätzlich basiert sie auf SMILES,[Wei88, Dav89] einer Sprache, die zur Beschreibung von Molekülen eingeführt wurde. SMILES und SMARTS haben einen ähnlichen Aufbau der Sprache. Der primäre Unterschied von SMILES und SMARTS ist neben dem Einsatzzweck die Tatsache, dass SMARTS eine Erweiterung der SMILES-Sprache ist und viel mehr Eigenschaften an den Knoten und Bindungen erlaubt. Diese können mit logischen Operatoren verknüpft oder durch allgemeine Platzhalter ersetzt werden. Darüber hinaus gibt es die Möglichkeit, die Umgebung um einen Knoten rekursiv zu spezifizieren. Diese sogenannte SMARTS-Rekursion ist die mächtigste Eigenschaft der SMARTS Sprache und erfordert in allen Aspekten besondere Betrachtung, sei es beim Schreiben solcher Muster, als auch bei deren Verarbeitung. Des Weiteren gab es auch schon Ideen, die zu einer Kanonisierung führen könnten, aber nicht weiter verfolgt wurden.[Say97]

Generell sind textuelle Beschreibungen von Molekülen und chemischen Mustern für Menschen nur schwer lesbar und begreifbar. In Bezug auf Moleküle gibt es seit jeher visuelle zwei und dreidimensionale Repräsentationen. Eine intuitive visuelle Beschreibung, die bekannten zweidimensionalen Molekülvisualisierungen nachempfunden ist, wurde im Vergleich zu den Beschreibungssprachen der Muster erst viel später eingeführt.[Sch10] Die Vorteile der Visualisierung sind offensichtlich, da sie es auch Laien bezüglich der SMARTS Syntax erlaubt, einfache strukturelle Fehler in den Mustern zu identifizieren. Unabhängig davon profitieren auch Experten von der Visualisierung, da die Struktur aus der textuellen Beschreibung generell nur mühsam abgeleitet werden kann. Der in den SMARTS.plus Webserver[Cen, D5, Sch10] integrierte SMARTSviewer[Sch10] folgt bei der Visualisierung der Elemente der SMARTS Sprache so weit wie möglich den IUPAC[Bre08a] Vorgaben zur Visualisierung von Molekülen. Die Symbolisierungen der SMARTS Eigenschaften ermöglichen dadurch ein intuitives Verständnis von chemischen Mustern.

Aufbauend auf der Visualisierung bestehender Muster ist es ein logischer Schritt, auch die Erstellung solcher mit graphischen Anwendungen zu unterstützen. Der SMARTSeditor[Sch13] löst genau dieses Problem. Dieser ermöglicht es, einen SMARTS Ausdruck mit einer graphischen Anwendung interaktiv zu erstellen, ohne der Syntax von SMARTS mächtig sein zu müssen.

Abseits der Visualisierung wurde ein weiterer Algorithmus entwickelt, der die Konzepte der Substrukturenumeration benutzt,[Han07] um SMARTS Muster zur Unterscheidung von Molekülmengen zu generieren.[Bie15] Dabei wird eine Menge als positiv, die andere als negativ definiert. Der Algorithmus generiert SMARTS Ausdrücke, so genannte Kontrastmuster, die Moleküle der positiven Menge von denen der negativen Menge unterscheiden.

All diese Probleme gab es für die Molekülbeschreibungen in SMILES[Wei88] nicht. Für Moleküle gab es seit jeher Visualisierungskonzepte. Zudem ist die Handhabung von Substrukturen und Restgruppen in der Chemie allgegenwärtig. Nach der Einführung von SMILES[Wei88] war die Uneindeutigkeit, der aus Molekülen generierten SMILES Ausdrücke, das größte Problem. Die sogenannten USMILES[Dav89] lösen das Problem relativ zuverlässig. Im Vergleich zum InChI[Hel14] sind die erzeugten USMILES häufig nur für Anwendungen einer chemischen Bibliothek eindeutig. Anstrengungen, SMILES auch über solche Bibliotheken und Anwendungen eindeutig zu gestalten wurden verfolgt[OBo12], haben sich aber nicht durchgesetzt.

Chemische Muster sind bekanntermaßen deutlich komplexer und schwieriger zu handhaben als Moleküle. Die häufige Nutzung von Mustern macht es aber wünschenswert, dass es auch für diese eine kanonische Form geben sollte. Zur Beschleunigung von Anwendungen wie dem SMARTS matching ist es üblich, selten vorkommende Elemente in SMARTS Mustern im ersten Knoten zu beschreiben. Über solche Empfehlungen für eine

effizientere Nutzung der Sprache hinaus, gibt es aber kein Konzept einer eindeutigen Darstellung von SMARTS. Es ist nicht einmal möglich, verschiedene SMARTS Ausdrücke unabhängig von ihrer Repräsentation algorithmisch zu vergleichen. Auf einem Datensatz können selbstverständlich SMARTS im Experiment als spezifischer oder als allgemeiner im Vergleich zu anderen klassifiziert werden. Das ist analog zur Berechnung der Kontrastmuster immer möglich. Die Gültigkeit solcher Aussagen bezieht sich aber immer nur auf den Datensatz, da ein zusätzliches Molekül, dass nicht Teil des Experiments war, diese Aussage widerlegen könnte. Zudem muss so ein Ansatz als rechenaufwendig und ungenau eingestuft werden. Bevor es möglich ist, chemische Muster im Allgemeinen und SMARTS im Speziellen zu kanonisieren, ist es notwendig, diese überhaupt vergleichen zu können. Eine Kanonisierung soll für identische Muster eine identische textuelle Repräsentation erzeugen. Daraus folgt, dass solange keine Möglichkeit existiert, chemische Muster überhaupt auf Identität zu vergleichen, eine Kanonisierung ebenfalls nicht möglich sein kann. Das folgende Kapitel widmet sich der ersten publizierten Methodik chemische Muster zu vergleichen die in [D3, D4] und [D5] umgesetzt wurden. Abschließend wird ein Ausblick gegeben, der die Möglichkeit der Kanonisierung in Folgeprojekten thematisiert.

4.2 Die SMARTScompare Methode

Sprachen chemischer Muster, zu denen auch die SMARTS Sprache gehört, erlauben logische Ausdrücke, mit denen Atome und Bindungen von getroffenen Molekülen beschrieben werden. Durch Äquivalenzumformungen und Regeln zur Reihenfolge der einzelnen atomaren Aussagen über Atome und Bindungen wäre es prinzipiell möglich, die Ausdrücke in Knoten und Kanten der Muster zu kanonisieren. Das Problem dabei ist, dass einerseits die atomaren Aussagen über Atome nicht unabhängig von einander sind und andererseits diese implizit durch die Graphstruktur gegeben sein können.

Der Vergleich molekularer Muster basiert auf der Umsetzung eines neuen theoretischen Konzepts *allgemeiner chemischer Muster*, dass so angelegt ist, dass es aus einer beliebigen Mustersprache generiert werden kann. Die Idee hinter allgemeinen chemischen Mustern ist es, in den Knoten und Kanten der Graphstruktur eines Musters alle Atom-, beziehungsweise Bindungszustände abzubilden, die ein Atom oder eine Bindung, auf die der Knoten oder die Kante abgebildet wird, annehmen kann und auch annimmt. Der Vorteil daran ist, das dadurch alle atomaren logischen Aussagen unabhängig voneinander sind. Das erlaubt einen einfachen Vergleich von Kanten und Knoten unabhängig von der gewählten Repräsentation. Zur Umsetzung des Konzepts werden auf Basis des NAOMI[Urb11, Urb14] Chemiemodells alle möglichen Atom und Bindungszustände enumeriert und als Fingerabdrücke an Knoten und Kanten angehängt. Die in die Definition der allgemeinen chemischen Muster integrierte Eindeutigkeit der Repräsentation wird mit einer mehrstufigen Bereinigungsprozedur der Fingerabdrücke umgesetzt.

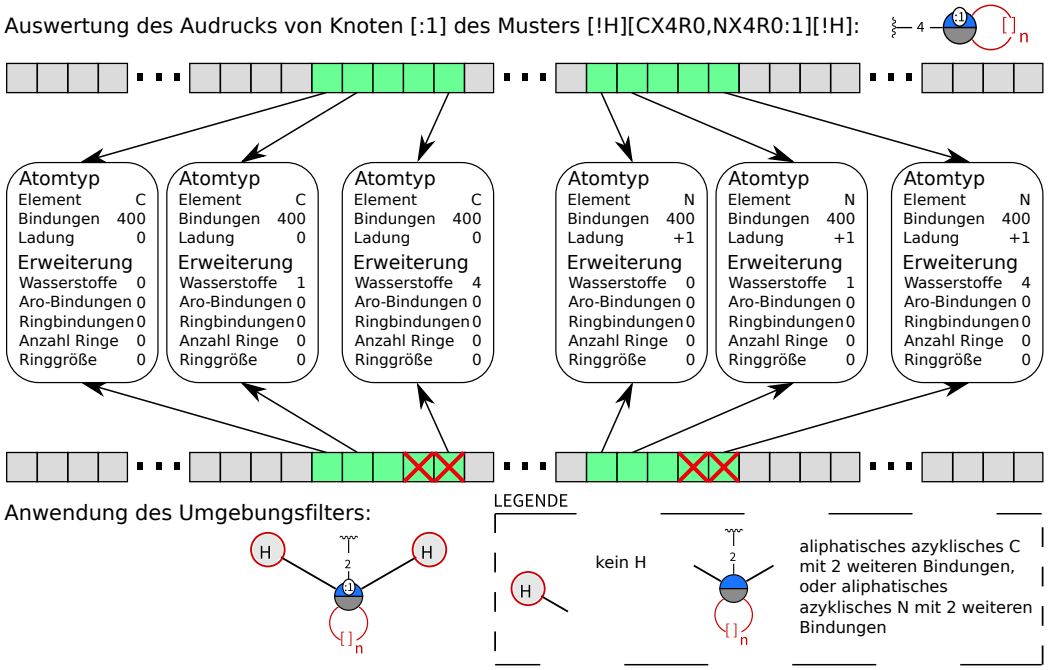


Abbildung 4.1: Symbolisierung von zwei Schritten der Fingerabdruckzuweisung zu den Knoten eines einfachen SMARTS Musters. Im oberen Teil ist der Ausdruck und eine Darstellung der im Fingerabdruck gesetzten Eigenschaften, die den Ausdruck beschreiben. Im unteren Teil wird ist der zweite Schritt dargestellt, der die Umgebung des Knotens berücksichtigt und Bits, die zu Eigenschaften gehören, die in der Umgebung des Knotens nicht mehr erfüllt sind, aus dem Fingerabdruck entfernt. Im mittleren Teil der Abbildung ist jeweils der Atomzustand als Eigenschaft eines Bits gegeben. Der drei-Zifferncode bei Bindungen steht für die Anzahlen an Einfach-, Doppel- und Dreifachbindungen an den Atomen. Diese Abbildung wurde aus [D3] übernommen und ins Deutsche übersetzt.

Die Prozedur der Fingerabdruckzuweisung ist in Abbildung 4.1 angedeutet. Ausgehend von einer initialen Zuweisung, bei der alleinig der Knotenausdruck berücksichtigt wird, werden in darauffolgenden Schritten all jene Bits aus den Fingerabdrücken wieder entfernt, die zu Atomzuständen gehören, die in der Umgebung des Knotens nicht möglich sind. Insbesondere eine (relativ) große Anzahl adjazenter Wasserstoffatome gehört zu den Eigenschaften, die in der Umgebung eines Knotens häufig nicht möglich sind.

Nach der Zuweisung der Fingerabdrücke wird ein cMCS auf den Graphrepräsentationen der Muster berechnet. Die Kompatibilität der Kanten und Knoten hängt dabei noch vom gewählten Suchmodus, „identische Muster“, „Teilmengenrelation der Muster“ oder „Ähnlichkeitssuche“ ab. Die SMARTS-spezifische Eigenschaft, dass Knoten ihre Umgebung rekursiv beschreiben können macht es erforderlich, dass auch bei der Suche nach Teilmengenrelationen ein MCS-Algorithmus benutzt werden muss. Sie ist durch eine rekursive Prozedur in die Berechnung des cMCS integriert.

Eines der größten Probleme für ungeübte Anwender der SMARTS Sprache ist der Zwang zur expliziten Angabe von Aromatizität. Innerhalb der SMARTScompare Anwendung wird bei Ringsystemen überprüft, ob diese in ihrer gegebenen Lokalisierung von Einfach- und Doppelbindungen einem aromatischen System in einem Molekül entsprechen könnten. In einem solchen Fall werden die Ausdrücke, der zu den als potenziell aromatisch befunden Knoten und Kanten angepasst und die Fingerabdrücke neu zugewiesen. Der Nutzer wird in einem solchen Fall über die interne Modifikation seines Musters informiert. Diese Handhabung ist in den zur Publikation gehörenden Anwendungen standardmäßig aktiviert und kann deaktiviert werden.

Neben der Berechnung von Teilmengenrelationen von molekularen Mustern kann SMARTScompare auch Ähnlichkeitswerte bestimmen. Für jedes Abbildungspaar von Knoten im Ergebnis des cMCS kann über die Fingerabdrücke ein Ähnlichkeitswert bestimmt werden. Die Aussagekraft dieser Ähnlichkeitswerte ist allerdings gering, da die von den Fingerabdruckbits dargestellten Eigenschaften einer systematischen Enumeration des Chemiemodells entstammen. Die Aussagekraft der Fingerabdruckähnlichkeit wird in SMARTScompare durch eine Gewichtung der einzelnen Bits erhöht. Dazu werden die Vorkommen der Bits auf einem Referenzdatensatz gezählt und als Gewichte eingesetzt. Die Gewichtung kann vom Nutzer für einen spezifischen Einsatzzweck berechnet und angepasst werden.[D4]

4.2.1 Validierung und Anwendungen der SMARTScompare Methode

Da die SMARTScompare Methode die erste ihrer Art ist, kann keine andere externe Anwendung zur Validierung oder zum Vergleich herangezogen werden. Stattdessen wurden Trefferprofile tausender SMARTS Muster auf einem Datensatz von über 100 Millionen Molekülen zu einem Referenzdatensatz möglicher Teilmengenrelationen zusammengefasst. Beim paarweisen Vergleich der SMARTS durfte SMARTScompare keine Teilmengenrelation vorhersagen, die nicht experimentell bestätigt wurde. Die gefundenen Fehler basieren auf Schwächen des Chemiemodells und einer in den Algorithmus optionalen eingebauten Vereinfachung zur Handhabung der Aromatizität in SMARTS. Zudem zeigt die Analyse der nicht von SMARTScompare gefundenen möglichen Teilmengenrelationen der SMARTS Muster die Notwendigkeit einer algorithmischen Methode, die nicht auf experimentellen Daten basiert. In vielen Fällen sind es seltene Atomzustände, die SMARTScompare eine experimentelle Teilmengenrelation nicht erkennen lassen. Da diese aber weiterhin theoretisch möglich sind, zeigt dies die Stärke dieser allgemeinen Lösung.

Für die Validierung des Ähnlichkeitskonzepts wurden Korrelationen zu Treffermengen von ausgewählten SMARTS auf zwei verschiedenen Datensätzen bestimmt. Insgesamt zeigen die Korrelationen akzeptable Ergebnisse. Beim durchgeführten Vergleich mit der Referenzähnlichkeit besteht für diese das Problem, dass diese wiederum experimentell bestimmt wurden und dass es kein unabhängiges Konzept zur Bestimmung von Ähnlichkeit von chemischen Mustern im Allgemeinen gibt.

4.3 Anwendungen auf Sammlungen von SMARTS Mustern

Die SMARTScompare[D4, D3] Anwendungen erlauben sowohl den visuellen Vergleich einzelner SMARTS Ausdrücke als auch den Vergleich ganzer Sammlungen mithilfe einer Kommandozeilenanwendung. Beide Anwendungen, der SMARTScompareViewer für den visuellen Vergleich als auch die Kommandozeilenanwendung SMARTScompare sind im SMARTS.plus-Webserver https://smarts.plus[Cen] integriert. Der Server bietet die Möglichkeiten der Visualisierung[Sch10] einzelner Ausdrücke, des Vergleichs zweier benutzerdefinierter Muster, den Vergleich eines benutzerdefinierten Musters mit neun verschiedenen Sammlungen und der Berechnung neuer SMARTS, die dazu geeignet sind, eine positive und eine negative Menge von Molekülen zu unterscheiden.[Bie15]

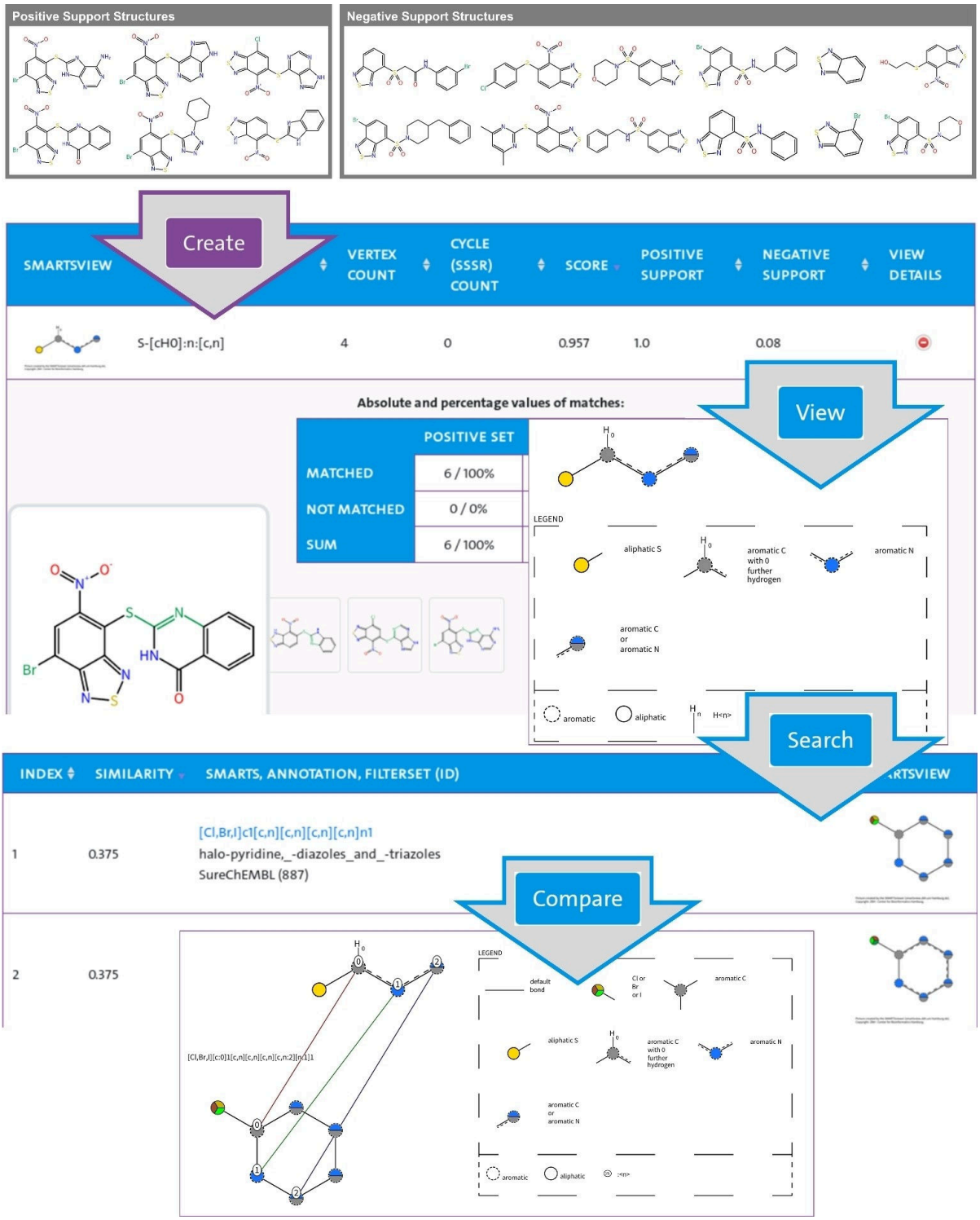


Abbildung 4.2: Eine Übersicht über die im SMARTS.plus integrierten Anwendungen an einer beispielhaften Entwicklung neuer SMARTS Muster. Für zwei hochgeladene Molekülmengen werden Kontrastmuster generiert (Create). Diese generierten SMARTS können betrachtet werden (View). Darüber hinaus besteht die Möglichkeit nach ähnlichen Mustern in bestehenden Sammlungen zu suchen (Search). Final können SMARTS auch direkt miteinander verglichen werden (Compare). Diese Abbildung ist von [D5] unverändert übernommen.

Die Integration der genannten Anwendungen in den SMARTS.plus Webserver wird in Abbildung 4.2 anhand eines beispielhaften Arbeitsablaufs dargestellt. Um herauszufinden, ob sich zwei Molekülmengen strukturell

unterscheiden, ist die Berechnung von Kontrastmustern eine sinnvolle Anwendung. Die Berechnung solcher Muster ist im SMARTS.plus integriert. Dazu müssen je zwei Moleküldateien auf den Server hochgeladen werden. Sofern eine strukturelle Eigenschaft durch ein SMARTS Muster beschrieben ist, werden häufig weitere sehr ähnliche Muster berechnet, die keine weitere Aussagekraft bezüglich der Unterscheidung der Mengen haben. Durch entsprechende Parametrisierung der Eingabe ist es sogar möglich, maximale gemeinsame Teilstrukturen auf der positiven Molekülmenge zu berechnen.

Die graphische SMARTSeditor Anwendung[Bie15] unterstützt zwei Filtermöglichkeiten nach Abschluss der Berechnung aller Kontrastmuster. Die Ergebnisansicht kann auf die nach Knotenanzahl kleinsten und größten SMARTS reduziert werden, die eine bestimmte Trennschärfe erreichen. Auf den SMARTS.plus Webserver ist erstere Filtermöglichkeit nach den kleinsten SMARTS ebenfalls in der interaktiven Ergebnisdarstellung möglich, ein Filtern nach den größten SMARTS bedingt aber immer eine abgeschlossene Berechnung. Deshalb stellt die zweite angebotene Filteroption auf dem SMARTS.plus einen Kompromiss zwischen kleinen und großen SMARTS dar. Weitere Details dazu werden im Kapitel 4.3.1 erläutert.

Nach der Berechnung von Kontrastmustern, wie in Abbildung 4.2, gezeigt folgt die visuelle Analyse der SMARTS. Das ist interaktiv in der Ergebnistabelle möglich, als auch über den „View“ Modus auf dem Server. Um herauszufinden, ob berechnete Muster in bestehenden Sammlungen vorhanden sind oder Ähnlichkeiten zu solchen SMARTS haben, bietet sich der „Search“ Modus an. Wie in der Abbildung kann nach ähnlichen SMARTS gesucht werden, vor allem wenn das eigene Muster eine geringe Anzahl an Knoten hat, kann sich die Suche nach spezifischeren SMARTS als vorteilhaft erweisen.

Der letzte Schritt wie in der Abbildung gezeigt, ist der visuelle Vergleich von SMARTS. Die angezeigten Bilder werden von der graphischen Anwendung SMARTScompareViewer über die Kommandozeile erstellt und vom Ruby Server angezeigt.

4.3.1 Berechnung der Kontrastmuster auf dem SMARTS.plus

Im Vergleich zur graphischen Anwendung werden auf dem Server maximal 300 Ergebnisse angezeigt und die Laufzeit der Anwendung wird ebenfalls zeitlich begrenzt. Darüber hinaus soll die Ergebnisauswahl bedeutsam sein und gleichzeitig einen Fortschritt mit interaktiver Visualisierung ermöglichen. Aus dem Grund wurde eine minimale Kommandozeilenanwendung entwickelt, die es ermöglicht, zwei verschiedene Filter in die Enumeration zu integrieren. Ersterer ist der Minimalitätsfilter nach Knotenzahl wie im SMARTSeditor. Da in diesem Fall kleinste SMARTS Ausdrücke von Interesse sind, ist die Enumeration über eine Breitensuche im Zustandsraum umgesetzt. Diese kleinsten Ausdrücke werden im Folgenden nur dann ausgegeben, wenn sie eine bessere Trennung der positiven und negativen Menge ermöglichen. Zur Bemessung der Trennschärfe werden die Treffermengen auf der positiven und der negativen Molekülmenge herangezogen. Sofern die Treffermenge auf der positiven Molekülmenge des enumerierten SMARTS eine Teilmenge eines bestehenden Musters ist werden die Treffer auf den negativen Molekülen verglichen. Beschreibt die Treffermenge des bestehenden Ausdrucks eine Teilmenge des enumerierten Kandidaten, wird diese als Duplikat bezügliche der Trennschärfe angesehen und verworfen. Die zweite Filteroption stellt einen Kompromiss zwischen kleinen und großen Mustern, gemessen an der Knotenanzahl dar. Die Strategie bei diesem Filter ist die Maximierung der Trennschärfe während der Enumeration. Alle Kandidaten zur weiteren Enumeration werden in einer Prioritätswarteschlange gespeichert und der Vielversprechendste wird verwendet. Daraus folgt, dass in diesem Fall kleine, aber nicht garantiert die kleinsten Muster für eine gegebene Trennschärfe berechnet werden. Zudem werden die jeweils größten Muster für die jeweilige Trennschärfe verwaltet. Nach Abbruch der Enumeration, z.B. durch ein gegebenes

Zeitkontingent oder die Anzahl der berechneten Ergebnisse, werden alle zum Zeitpunkt des Abbruchs größten SMARTS ausgegeben.

4.3.2 SMARTS Netzwerkanalyse

Die SMARTScompare Anwendungen bieten die Möglichkeit, große Sammlungen von SMARTS Mustern nach ähnlichen Mustern zu untersuchen, wie es auch in der Ähnlichkeitsanalyse in [D4] durchgeführt wurde. Wenn der Fokus hingegen auf die Substruktureigenschaft gelegt wird, ergeben sich ganz andere Netzwerkeigenschaften, die es sich zu analysieren lohnt. Unter Berechnung aller paarweisen Teilmengenrelationen ist der SMARTSexplore Webserver[Fen21] entstanden, der vom SMARTS.plus aus verlinkt wird und unter https://smartsexplore.zbh.uni-hamburg.de erreichbar ist.

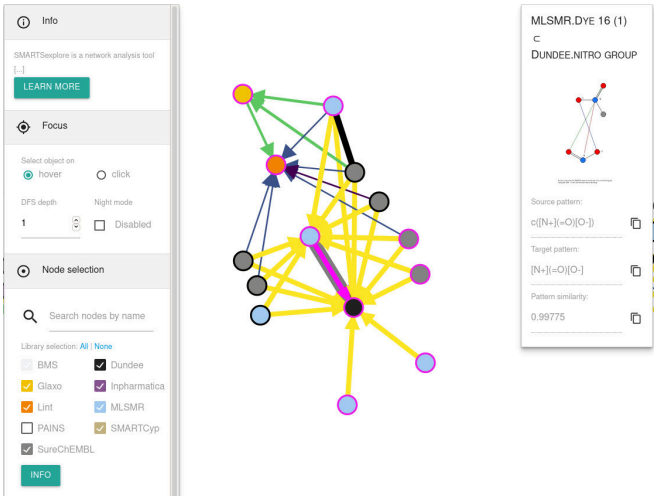


Abbildung 4.3: Die Visualisierung auf dem SMARTSexplore von zwei SMARTS Mustern, die Nitrogruppen beschreiben und in einer Teilmengenbeziehung stehen. Auf der linken Seite sind Optionen zur Darstellung und Auswahl der SMARTS und Sammlungen. Im Zentrum der Abbildung befindet sich die kraftfeldbasierte 2D-Visualisierung der Teilmengenrelationen zwischen den einzelnen SMARTS. Im Fenster auf der rechten Seite wird das SMARTS Muster oder die über Kanten bestehende Teilmengenbeziehung dargestellt. In diesem Fall ist es die Relation vom spezifischeren SMARTS „Dye 16(1)“, aus MLSMR, zum allgemeineren „Nitro Group“ aus Dundee.

Der Server selbst bietet nach Berechnung aller Teilmengenbeziehungen erstklassige Analysemöglichkeiten der Zusammenhänge von SMARTS Mustern aus verschiedenen Sammlungen an, wie in Abbildung 4.3 dargestellt. Der Server zeigt alle SMARTS der ausgewählten Sammlungen auf der zentralen Fläche als Knoten an. Jeder Knoten steht für einen SMARTS, dessen Visualisierung betrachtet werden kann. Besteht eine Teilmengenrelation, die im ausgewählten Ähnlichkeitsintervall liegt, so sind zwei Knoten mit einer Kante verbunden. Diese kann ebenfalls betrachtet werden. Für die Platzierung in der Ebene wird ein Kraftfeld verwendet. Zur besseren Analyse lassen sich die Knoten manuell verschieben.

Komplexe und längere Abhängigkeitsketten verschiedener SMARTS, wie in [D4] durch aufwendige Analyse der paarweisen Ergebnisse gefunden, lassen sich auf diese Weise interaktiv und ohne Programmierkenntnisse gewinnen. Das wird mit dem Vergleich der beiden baumartigen Hierarchien aus Abbildung 4.4 deutlich. In Teil A sind die SMARTS entsprechend der Teilmengenrelation von oben nach unten visualisiert und mit den jeweiligen Sammlungen, aus denen sie stammen, annotiert. Teil B zeigt, dass diese Hierarchie ohne weiteren Einsatz der SMARTScompare Anwendungen interaktiv auf dem SMARTSexplore Webserver nachvollzogen

werden kann. Jeder Knoten beschreibt einen SMARTS, dessen Informationen jeweils rechts angezeigt werden. Wenn Kanten ausgewählt werden, so wird die SMARTScompare Abbildung der beiden SMARTS-Muster angezeigt.

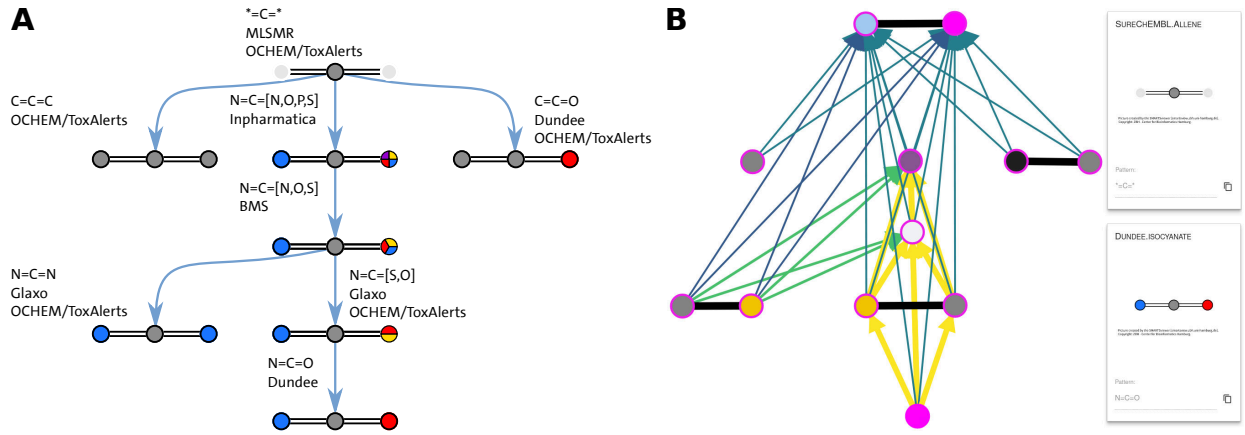


Abbildung 4.4: In beiden Teilen der Abbildung wird eine in den Sammlungen vorhandene Hierarchie von SMARTS, die allesamt Allene beschreiben, dargestellt. Die Darstellungen gehen jeweils von generischen Mustern (oben) zu immer spezifischeren SMARTS (unten). A wurde manuell aus den Daten des paarweisen Vergleichs aller SMARTS extrahiert, während B online auf dem SMARTSexplore Webserver interaktiv gesucht und nachvollzogen wurde. Im rechten Teil von B ist das oberste und unterste Muster der Hierarchie visualisiert. Für die Darstellung wurde das Ähnlichkeitsintervall für die Kantenselektion auf [0,1] eingestellt. A ist aus [D4] unverändert übernommen.

4.4 Ausblick

Das SMARTScompare Projekt, die daraus entstandenen Anwendungen sowie der Webserver haben erstmalig die Möglichkeit geschaffen, als unvergleichlich geltende chemische Muster in Teilmengen- und Ähnlichkeitsbeziehungen zu setzen. Die Publikationen [D3] und [D4] beschreiben klar und deutlich den Einsatzbereich und bekannte Schwächen für zukünftige Verbesserungen der Methode. In Anbetracht der Ähnlichkeitsanalyse kann ein eingeschränkter nicht-zusammenhängender MCS (dMCS) für bessere Ähnlichkeitswerte sorgen. Ähnlich zu Molekülen könnten über verschieden lange Ketten für das Muster wichtige funktionelle Gruppen verbunden sein. Ein dMCS kann diese Tatsachen berücksichtigen.

Für die Praxis relevanter wäre eine Kanonisierung von SMARTS. Kanonisierungen sind in der Chemieinformatik wichtige Bestandteile, wie an der Relevanz von eindeutigen SMILES (USMILES)[Dav89] gegenüber einfachen SMILES[Wei88] ersichtlich ist. Darüber hinaus stellen der InChI[Hel14, Hel15] und InChI-Key die einzigen Möglichkeiten dar, über verschiedene Chemieinformatiksysteme hinweg eindeutige Molekülrepräsentationen zu erzeugen. Der gleiche Ansatz wird auch mit dem InChI für Reaktionen (RInChI)[Gre18] umgesetzt, für die es ebenfalls eine vollständige Beschreibung über mehrere Ebenen als auch einen mit Hashing erzeugten einfach vergleichbaren Schlüsselwert gibt. Eine mögliche Kanonisierung von SMARTS wurde initial von Tim Kuhrt in einer Bachelorarbeit[S2] untersucht. Sie verfolgte den Ansatz der Berechnung einer eindeutigen Knotenreihenfolge innerhalb eines SMARTS Ausdrucks, die der Kanonisierung der USMILES nachempfunden ist. Möglich wurde das unter anderem durch die eindeutigen Fingerabdrücke in den Knoten und Kanten von SMARTS Ausdrücken. Die Kanonisierung bietet leider keine Möglichkeit SMARTS-Rekursion über ihren Einfluss auf den Fingerabdruck des Knotens, an dem sie spezifiziert ist, zu berücksichtigen. Für nicht-rekursive SMARTS Ausdrücke lässt sich an diesem Punkt ein Fingerabdruck erzeugen. Der Ansatz, eine eindeutige menschenlesbare Repräsentation zu generieren, wurde darüber hinaus noch nicht verfolgt. Ein simpler, aber ineffizienter und unleserlicher Ansatz wäre die Auflistung aller möglicher Zustände pro Knoten. Um eine Lesbarkeit zu erreichen,

müsste ein Algorithmus, der so einen Ausdruck schreibt, in der Lage sein, implizit gegebene oder redundante Eigenschaftsbeschreibungen zu erkennen und zusammenzufassen.

Eine andere und praxisnähere Verwendung der mit SMARTScompare vorgestellten Konzepte ist die Beschleunigung der Suche nach SMARTS Treffern in Moleküldatenbanken. Jedes Atom hat einen eindeutigen Erweiterten Atomtyp im Modell das SMARTScompare Algorithmus. Die Frage, ob ein SMARTS Knoten ein Atom trifft, kann folglich darauf reduziert werden, ob der entsprechende Atomtyp im Fingerabdruck des Knoten vorhanden ist. Wenn viele SMARTS Muster auf einer unveränderten Molekülmenge gesucht werden, so kann auch ein für die Molekülmenge spezifischer Fingerabdruck berechnet werden. Das bietet einerseits Vereinfachung beim Nachgucken, ob ein bestimmtes Bit gesetzt ist und hat den Vorteil, dass die Ringeigenschaften nicht mehr für zu große Zahlenwerte zusammengefasst werden müssen. Den größten Vorteil aus einer solchen Analyse bieten wahrscheinlich die präzise Beschreibung an getroffenen Atomen für einen beliebigen Knotenausdruck und die Möglichkeit zur Kompression von Molekülen zu einer Binärdarstellung, die teilweise mit unter zwei Byte pro Atom auskommt.[May19]

5 Zusammenfassung

Die in dieser Dissertation beschriebenen Algorithmen befassen sich allesamt mit dem „Maximale gemeinsame Teilstruktur“ (MCS) Problem. Im Rahmen des Promotionsprojekts wurde ein adaptiver MCS Algorithmus entwickelt und als Freie Software auf der Plattform GitHub veröffentlicht (https://github.com/rareylab/RIMACS). Für die anderen beiden Algorithmen wurden Computerprogramme geschrieben, die die Anwendung des MCS in neuartigen Anwendungsbereichen ermöglichen. Die Entwicklung des RIMACS Algorithmus zur Lösung des MCS Problems stellt die Grundlage für die Effizienz der beiden anderen in der Promotion beschriebenen Programme dar. Mit RIMACS wurde eine effiziente exakte Methode vorgestellt, die im Gegensatz zu vielen anderen MCS-Algorithmen nicht auf der Transformation in das Cliquenproblem basiert. Das ermöglicht eine schlanke Struktur des Algorithmus und den direkten Aufbau der Abbildung zwischen den Molekülen. Darüber hinaus bietet RIMACS genaue Kontrolle darüber, ob und wie ein nicht-zusammenhängender MCS berechnet werden soll. Insbesondere lassen sich die Anzahl an Zusammenhangskomponenten im Ergebnis und auch deren minimale Größe kontrollieren. In der Evaluation wurde besonderer Wert darauf gelegt, die Auswirkungen auf die Laufzeit und die Größe der gefundenen Teilstruktur des nicht-zusammenhängenden MCS mit dem zusammenhängenden zu vergleichen. Anschließend wurde für den zusammenhängenden MCS die RIMACS Methode mit anderen exakten und heuristischen Methoden aus der Literatur verglichen. Für dieses Problem wurden vergleichbare Laufzeiten mit den heuristischen Methoden erreicht und die Performance der externen exakten Methoden bei weitem übertroffen.

Die beschriebene Effizienz und Adaptivität der RIMACS Methode ermöglichten es, durch Anwendung der Konzepte der dynamischen Programmierung ein zusammenhängendes induziertes MCS Problem in unenumerierten kombinatorischen Fragmenträumen zu lösen. Der für diesen Zweck entwickelte SpaceMACS Algorithmus berechnet einen eingeschränkten cMCS. Damit mit der SpaceMACS Methode nicht nur das „beste“ Ergebnis bezüglich der Anzahl an abgebildeten Knoten berechnet werden kann, musste das MCS Problem, das tatsächlich gelöst wird, weiter angepasst werden. Für die korrekte Behandlung der Linker ist eine Gewichtung des MCS notwendig und für die Berechenbarkeit suboptimaler Ergebnisse ab Rang zwei muss bei der MCS Berechnung auf den Fragmenten eine Art der „Alle maximalen gemeinsamen Teilstrukturen“ Variante des MCS gelöst werden. Die bis zu diesem Punkt vorgestellte Methode berechnet eine Teilstruktur, die sich über mindestens zwei Fragmente ausdehnt. Damit die Ergebnisse der Methode nicht auf solche Fälle beschränkt sind, wird in einem weiteren Schritt wieder ein normaler zusammenhängender MCS mit den Fragmenten unter Aussparung der Verbindungsatome berechnet. In der Anwendung erweist sich die SpaceMACS Anwendung als äußerst effizient und erlaubt es, virtuelle Substanzbibliotheken mit 20 Milliarden oder auch 290 Billionen Molekülen innerhalb von Minuten nach ähnlichen Molekülen zu einer Anfragestruktur zu durchsuchen. Diese kann als Molekül vorliegen, aber auch eingeschränkt als molekulares Muster in der SMARTS-Sprache gestellt sein.

Im abschließenden Projekt der Promotion wurde der MCS als Methode der Wahl für den Vergleich von chemischen Mustern in der SMARTS Sprache gewählt. Durch eine Modellierung über Fingerabdrücke, die den chemischen Raum von Atomen und Bindungen abbilden, können die Kanten und Knoten der internen Graphstruktur des SMARTS miteinander verglichen werden. Das ermöglichte die Anwendung eines MCS Algorithmus, um Gemeinsamkeiten festzustellen. Die Besonderheit der SMARTS Sprache, weitere SMARTS Ausdrücke rekursiv

zu enthalten, führte dazu, dass der MCS in einem rekursiven Algorithmus ausgeführt wurde. Anschließend mussten die diversen Teilergebnisse wieder durch Enumeration des Zustandsraums zu den tatsächlichen Ergebnissen zusammengesetzt werden. Die entwickelten Anwendungen zum Vergleich von SMARTS wurden in den SMARTS.plus Webserver integriert. Dort unterstützen sie die Visualisierung, den Vergleich einzelner sowie den interaktiven Vergleich von SMARTS mit denen ganzer Sammlungen. Darüber hinaus wurde ein Verfahren zur Berechnung von Kontrastmustern in den Server integriert. Die zugrunde liegende Methode, einer Substrukturenumeration, ist die Methode der Wahl zur Lösung des cMCS in mehr als zwei Graphen, dem Multi-MCS Problems. Insgesamt ermöglicht das Projekt die interaktive visuelle Arbeit mit SMARTS Ausdrücken, die für ungeübte Anwenderinnen und Anwender nicht möglich wäre und gibt spannende Einblicke in die Struktur von bestehenden Sammlungen von SMARTS Mustern.

Insgesamt zeigen der Fokus und die in dieser Promotion erzielten Ergebnisse, dass MCS basierte Ähnlichkeitsmaße weiterhin relevant sind und bestehende Anwendungsfelder zurückgewonnen und neuartige erschlossen werden können. Insbesondere in der molekularen Ähnlichkeitssuche hat sich die Fingerabdruckähnlichkeit als Standardverfahren der Ähnlichkeitssuche etabliert. Mit der Einführung des SpaceMACS Methode ist es erstmals wieder möglich, große kombinatorische Molekülmengen effizient nach gemeinsamen Teilstrukturen zu durchsuchen.

Die vom SMARTScompare Algorithmus abgeleiteten Anwendungen erweitern die Anwendungsgebiete von MCS Algorithmen auf chemische Muster. Neben einer neuartigen automatisch generierten Beschreibung der chemischen Muster sind es vor allem rekursive Eigenschaften chemischer Mustersprachen, die den Einsatz der MCS Methodik erzwingen.

Literatur

[Ash97] Ash, Sheila; Cline, Malcolm A.; Homer, R. Webster; Hurst, Tad und Smith, Gregory B.: „SYBYL Line Notation (SLN): A Versatile Language for Chemical Structure Representation“. In: *J. Chem. Inf. Comput. Sci.* 37.1 (1997), S. 71–79. doi: 10.1021/ci960109j (siehe S. 3, 29).

[Awa17] Awale, Mahendra; Visini, Ricardo; Probst, Daniel; Arús-Pous, Josep und Reymond, Jean Louis: „Chemical space: Big data challenge for molecular diversity“. In: *Chimia (Aarau).* 71.10 (2017), S. 661–666. doi: 10.2533/chimia.2017.661 (siehe S. 17).

[Bae10] Baell, Jonathan B und Holloway, Georgina A: „New substructure filters for removal of pan assay interference compounds (PAINS) from screening libraries and for their exclusion in bioassays“. In: *J. Med. Chem.* 53.7 (2010), S. 2719–2740. doi: 10.1021/jm901137j. url: http://pubs.acs.org/doi/pdf/10.1021/jm901137j (siehe S. 29).

[Bar76] Barrow, H. G. und Burstall, R. M.: „Subgraph isomorphism, matching relational structures and maximal cliques“. In: *Inf. Process. Lett.* 4.4 (1976), S. 83–84. doi: 10.1016/0020-0190(76)90049-1 (siehe S. 2).

[Bel19] Bellmann, Louis; Penner, Patrick und Rarey, Matthias: „Connected Subgraph Fingerprints: Representing Molecules Using Exhaustive Subgraph Enumeration“. In: *J. Chem. Inf. Model.* 59.11 (2019), S. 4625–4635. doi: 10.1021/acs.jcim.9b00571 (siehe S. 1).

[Bel20] Bellmann, Louis; Penner, Patrick und Rarey, Matthias: „Topological Similarity Search in Large Combinatorial Fragment Spaces“. In: *J. Chem. Inf. Model.* (2020). doi: 10.1021/acs.jcim.0c00850 (siehe S. 10, 15, 20).

[Bem96] Bemis, Guy W und Murcko, Mark A: „The properties of known drugs. 1. Molecular frameworks“. In: *J. Med. Chem.* 39.15 (1996), S. 2887–2893. doi: 10.1021/jm9602928 (siehe S. 17, 28).

[Bie15] Bietz, Stefan; Schomburg, Karen T.; Hilbig, Matthias und Rarey, Matthias: „Discriminative Chemical Patterns: Automatic and Interactive Design“. In: *J. Chem. Inf. Model.* 55.8 (2015), S. 1535–1546. doi: 10.1021/acs.jcim.5b00323 (siehe S. 30, 33, 35, 63).

[Bio] BioSolveIT GmbH: infiniSee. url: https://www.biosolveit.de/infiniSee/ (besucht am 20. 10. 2021) (siehe S. 25).

[Bio19] BioSolveIT GmbH: KnowledgeSpace_290tr_2019-05.space. 2019. url: https://www.biosolveit.de/infiniSee/#knowledgespace (besucht am 28. 05. 2021) (siehe S. 16, 25).

[Bla05] Blake, James F: „Identification and evaluation of molecular properties related to preclinical optimization and clinical fate.“ In: *Med. Chem. (Los. Angeles).* 1.6 (2005), S. 649–655. doi: 10.2174/157340605774598081. url: http://www.ingentaconnect.com/content/ben/mc/2005/00000001/00000006/art00013 (siehe S. 29).

[Blu09] Blum, Lorenz C. und Reymond, Jean Louis: „970 Million druglike small molecules for virtual screening in the chemical universe database GDB-13“. In: *J. Am. Chem. Soc.* 131.25 (2009), S. 8732–8733. doi: 10.1021/ja902302h (siehe S. 17).

[Bre08a] Brecher, Jonathan: Graphical representation standards for chemical structure diagrams: (IUPAC Recommendations 2008). Bd. 80. 2. 2008, S. 277–410. doi: 10.1351/pac200880020277 (siehe S. 30).

[Bre08b] Brenk, Ruth; Schipani, Alessandro; James, Daniel; Krasowski, Agata; Gilbert, Ian Hugh; Frearson, Julie und Wyatt, Paul Graham: „Lessons learnt from assembling screening libraries for drug discovery for neglected diseases“. In: *ChemMedChem* 3.3 (2008), S. 435–444. doi: 10.1002/cmdc.200700139. arXiv: 1408.1149 (siehe S. 29).

[Bro09] Brown, Nathan: „Chemoinformatics—an introduction for computer scientists“. In: *ACM Comput. Surv.* 41.2 (2009), S. 1–38. doi: 10.1145/1459352.1459353. url: http://doi.acm.org/10.1145/ (siehe S. 29).

[Bro73] Bron, Coen und Kerbosch, Joep: „Algorithm 457: finding all cliques of an undirected graph“. In: *Commun. ACM* 16.9 (1973), S. 575–577. doi: 10.1145/362342.362367. arXiv: citation.cfm?doid=362342.362367 [dl.acm.org]. url: http://portal.acm.org/citation.cfm?doid=362342.362367 (siehe S. 2).

[Bru12] Bruns, Robert F. und Watson, Ian A.: „Rules for identifying potentially reactive or promiscuous compounds“. In: *J. Med. Chem.* 55.22 (2012), S. 9763–9772. doi: 10.1021/jm301008n (siehe S. 29).

[Cao08] Cao, Yiqun; Jiang, Tao und Girke, Thomas: „A maximum common substructure-based algorithm for searching and predicting drug-like compounds“. In: *Bioinformatics* 24.13 (2008), S. 366–374. doi: 10.1093/bioinformatics/btn186 (siehe S. 7–10).

[Caz05] Cazals, F. und Karande, C.: „An algorithm for reporting maximal c-cliques“. In: *Theor. Comput. Sci.* 349.3 (2005), S. 484–490. doi: 10.1016/j.tcs.2005.09.038 (siehe S. 8, 10).

[Cen] Center for Bioinformatics Hamburg: SMARTS.plus. url: https://smarts.plus (besucht am 20. 10. 2021) (siehe S. 30, 33).

[Cha09] Chaudhaery, Shailendra S; Roy, Kuldeep K und Saxena, Anil K: „Consensus superiority of the pharmacophore-based alignment, over maximum common substructure (MCS): 3D-QSAR studies on carbamates as acetylcholinesterase inhibitors“. In: *J. Chem. Inf. Model.* 49.6 (2009), S. 1590–1601. doi: 10.1021/ci900049e (siehe S. 2).

[Cla20] Clark, David E.: „Virtual screening: Is bigger always better? Or can small be beautiful?“ In: *J. Chem. Inf. Model.* 60.9 (2020), S. 4120–4123. doi: 10.1021/acs.jcim.0c00101 (siehe S. 18).

[Cor04] Cordella, Luigi P.; Foggia, Pasquale; Sansone, Carlo und Vento, Mario: „A (sub)graph isomorphism algorithm for matching large graphs“. In: *IEEE Trans. Pattern Anal. Mach. Intell.* 26.10 (2004), S. 1367–1372. doi: 10.1109/TPAMI.2004.75 (siehe S. 8).

[Dal13] Dalke, Andrew und Hastings, Janna: „FMCS: a novel algorithm for the multiple MCS problem“. In: *J. Cheminform.* 5.Suppl 1 (2013), O6. doi: 10.1186/1758-2946-5-s1-o6 (siehe S. 7, 8).

[Dav89] David Weininger; Weininger, Arthur und Weininger, Joseph L.: „SMILES . 2 . Algorithm for Generation of Unique SMILES Notation“. In: *J. Chem. Inf. Comput. Sci.* 29.2 (1989), S. 97–101. doi: 10.1021/ci00062a008 (siehe S. 3, 30, 37).

[Day] Daylight Chemical Information Systems Inc: SMARTS. url: https://www.daylight.com/dayhtml/doc/theory/theory.smarts.html (besucht am 20. 10. 2021) (siehe S. 3, 29).

[Deg08] Degen, Jörg; Wegscheid-Gerlach, Christof; Zaliani, Andrea und Rarey, Matthias: „On the art of compiling and using 'drug-like' chemical fragment spaces“. In: *ChemMedChem* 3.10 (2008), S. 1503–1507. doi: 10.1002/cmdc.200800178 (siehe S. 15).

[Det10] Detering, Carsten; Claussen, H; Gastreich, Marcus und Lemmen, Christian: „KnowledgeSpace - a publicly available virtual chemistry space“. In: *J. Cheminform.* 2.S1 (2010), S. 53757. doi: 10.1186/1758-2946-2-s1-o9 (siehe S. 16).

[Det91] Dethlefsen, Winfried; Lynch, Michael F.; Gillet, Valerie J.; Downs, Geoffrey M.; Holliday, John D. und Barnard, John M.: „Computer Storage and Retrieval of Generic Chemical Structures in Patents. 11. Theoretical Aspects of the Use of Structure Languages in a Retrieval System“. In: *J. Chem. Inf. Comput. Sci.* 31.2 (1991), S. 233–253. doi: 10.1021/ci00002a009 (siehe S. 3).

[Drua] Drugbank: Nilutamide. url: https://go.drugbank.com/drugs/DB00665 (besucht am 20. 10. 2021) (siehe S. 22).

[Drub] Drugbank: Torasemide. url: https://go.drugbank.com/drugs/DB00214 (besucht am 20. 10. 2021) (siehe S. 23, 24).

[Due16] Duesbury, Edmund; Holliday, John D. und Willett, Peter: „Maximum Common Subgraph Isomorphism Algorithms: A Review“. In: *MATCH Commun. Math. Comput. Chem.* 77 (2016), S. 213–232. url: http://eprints.whiterose.ac.uk/102232/ (siehe S. 2, 7).

[Due18] Duesbury, Edmund; Holliday, John und Willett, Peter: „Comparison of Maximum Common Subgraph Isomorphism Algorithms for the Alignment of 2D Chemical Structures“. In: *ChemMedChem* 13.6 (2018), S. 588–598. doi: 10.1002/cmdc.201700482 (siehe S. 7, 8, 10, 13).

[Ehr11] Ehrlich, Hans Christian und Rarey, Matthias: „Maximum common subgraph isomorphism algorithms and their applications in molecular science: A review“. In: *Wiley Interdiscip. Rev. Comput. Mol. Sci.* 1.1 (2011), S. 68–79. doi: 10.1002/wcms.5 (siehe S. 2, 7).

[Ehr12] Ehrlich, Hans Christian; Volkamer, Andrea und Rarey, Matthias: „Searching for Substructures in Fragment Spaces.pdf“. In: *J. Chem. Inf. Model.* 52.12 (2012), S. 3181–3189 (siehe S. 19, 27).

[Ehr13] Ehrlich, Hans Christian; Henzler, Angela M. und Rarey, Matthias: „Searching for recursively defined generic chemical patterns in nonenumerated fragment spaces“. In: *J. Chem. Inf. Model.* 53.7 (2013), S. 1676–1688. doi: 10.1021/ci400107k (siehe S. 19, 27).

[Ena] Enamine Ltd. url: https://enamine.net/ (besucht am 20. 10. 2021) (siehe S. 15).

[Ena21] Enamine Ltd.: REALSpace_19bn_2021-04.space. 2021. url: https://www.biosolveit.de/infiniSee/#realspace (besucht am 28. 05. 2021) (siehe S. 16, 22, 24–26).

[Eng15] Englert, Péter und Kovács, Péter: „Efficient heuristics for maximum common substructure search“. In: *J. Chem. Inf. Model.* 55.5 (2015), S. 941–955. doi: 10.1021/acs.jcim.5b00036 (siehe S. 7, 8, 10, 11, 13).

[Fen21] Fender, Inken; Plümer, Pia und Welker, Simon: SMARTSexplore. 2021. url: https://smartsexplore.zbh.uni-hamburg.de (besucht am 20. 10. 2021) (siehe S. 36).

[Fin07] Fink, Tobias und Reymond, Jean Louis: „Virtual exploration of the chemical universe up to 11 atoms of C, N, O, F: Assembly of 26.4 million structures (110.9 million stereoisomers) and analysis for new ring systems, stereochemistry, physicochemical properties, compound classes, and drug discove“. In: *J. Chem. Inf. Model.* 47.2 (2007), S. 342–353. doi: 10.1021/ci600423u (siehe S. 17).

[Fri17] Friedrich, Nils Ole; De Bruyn Kops, Christina; Flachsenberg, Florian; Sommer, Kai; Rarey, Matthias und Kirchmair, Johannes: „Benchmarking Commercial Conformer Ensemble Generators“. In: *J. Chem. Inf. Model.* 57.11 (2017), S. 2719–2728. doi: 10.1021/acs.jcim.7b00505 (siehe S. 13, 14).

[Gau17] Gaulton, Anna u. a.: „The ChEMBL database in 2017“. In: *Nucleic Acids Res.* 45.D1 (2017), S. D945–D954. doi: 10.1093/nar/gkw1074. arXiv: 1612.04326 (siehe S. 29).

[Gre18] Grethe, Guenter; Blanke, Gerd; Kraut, Hans und Goodman, Jonathan M.: „International chemical identifier for reactions (RInChI)“. In: *J. Cheminform.* 10.1 (2018), S. 1–9. doi: 10.1186/s13321-018-0277-8. url: https://doi.org/10.1186/s13321-018-0277-8 (siehe S. 37).

[Gre20] Grebner, Christoph; Malmerberg, Erik; Shewmaker, Andrew; Batista, Jose; Nicholls, Anthony und Sadowski, Jens: „Virtual screening in the cloud: How big is big enough?“ In: *J. Chem. Inf. Model.* 60.9 (2020), S. 4274–4282. doi: 10.1021/acs.jcim.9b00779 (siehe S. 17).

[Gro08] Grosso, Andrea; Locatelli, Marco und Pullan, Wayne: „Simple ingredients leading to very efficient heuristics for the maximum clique problem“. In: *J. Heuristics* 14.6 (2008), S. 587–612. doi: 10.1007/s10732-007-9055-x (siehe S. 7, 8).

[Han07] Han, Jiawei; Cheng, Hong; Xin, Dong und Yan, Xifeng: „Frequent pattern mining: Current status and future directions“. In: *Data Min. Knowl. Discov.* 15.1 (2007), S. 55–86. doi: 10.1007/s10618-006-0059-1 (siehe S. 30).

[Han98] Handschuh, Sandra; Wagener, Markus und Gasteiger, Johann: „Superposition of three-dimensional chemical structures allowing for conformational flexibility by a hybrid method“. In: *J. Chem. Inf. Comput. Sci.* 38.2 (1998), S. 220–232. doi: 10.1021/ci970438r (siehe S. 2).

[Han99] Hann, Mike; Hudson, Brian; Lewell, Xiao; Lifely, Rob; Miller, Luke und Ramsden, Nigel: „Strategic Pooling of Compounds for High-Throughput Screening“. In: *J. Chem. Inf. Model.* 39.5 (1999), S. 897–902. doi: 10.1021/ci990423o. url: http://pubs.acs.org/doi/pdf/10.1021/ci990423o%20http://www.ncbi.nlm.nih.gov/pubmed/10529988%5Cnhttp://pubs.acs.org/cgi-bin/doilookup/?10.1021/ci990423o (siehe S. 29).

[Har11] Hariharan, Ramesh; Janakiraman, Anand; Nilakantan, Ramaswamy; Singh, Bhupender; Varghese, Sajith; Landrum, Gregory und Schuffenhauer, Ansgar: „MultiMCS : A Fast Algorithm for the Maximum Common Substructure Problem on Multiple Molecules“. In: *J. Chem. Inf. Model.* 51 (2011), S. 788–806. doi: 10.1021/ci100297y (siehe S. 7, 8).

[Haw07] Hawkins, Paul C.D.; Skillman, A. Geoffrey und Nicholls, Anthony: „Comparison of shape-matching and docking as virtual screening tools“. In: *J. Med. Chem.* 50.1 (2007), S. 74–82. doi: 10.1021/jm0603365 (siehe S. 1).

[Hel14] Heller, Stephen: „InChI – the worldwide chemical structure standard“. In: *J. Cheminform.* 6.S1 (2014), S. 1–9. doi: 10.1186/1758-2946-6-s1-p4 (siehe S. 30, 37).

[Hel15] Heller, Stephen R.; McNaught, Alan; Pletnev, Igor; Stein, Stephen und Tchekhovskoi, Dmitrii: InChI, the IUPAC International Chemical Identifier. Bd. 7. 1. Journal of Cheminformatics, 2015, S. 1–34. doi: 10.1186/s13321-015-0068-4. url: http://dx.doi.org/10.1186/s13321-015-0068-4 (siehe S. 37).

[Hil13] Hilbig, Matthias; Urbaczek, Sascha; Groth, Inken; Heuser, Stefan und Rarey, Matthias: „MONA - Interactive manipulation of molecule collections“. In: *J. Cheminform.* 5.8 (2013), S. 1–10. doi: 10.1186/1758-2946-5-38 (siehe S. 1).

[Hil15] Hilbig, Matthias und Rarey, Matthias: „MONA 2: A Light Cheminformatics Platform for Interactive Compound Library Processing“. In: *J. Chem. Inf. Model.* 55.10 (2015), S. 2071–2078. doi: 10.1021/acs.jcim.5b00292 (siehe S. 1).

[Hof19] Hoffmann, Torsten und Gastreich, Marcus: „The next level in chemical space navigation: going far beyond enumerable compound libraries“. In: *Drug Discov. Today* 24.5 (2019), S. 1148–1156. doi: 10.1016/j.drudis.2019.02.013. url: https://doi.org/10.1016/j.drudis.2019.02.013 (siehe S. 15–17).

[Irw20] Irwin, John J.; Tang, Khanh G.; Young, Jennifer; Dandarchuluun, Chinzorig; Wong, Benjamin R.; Khurelbaatar, Munkhzul; Moroz, Yurii S.; Mayfield, John und Sayle, Roger A.: „ZINC20 - A Free Ultralarge-Scale Chemical Database for Ligand Discovery“. In: *J. Chem. Inf. Model.* 60.12 (2020), S. 6065–6073. doi: 10.1021/acs.jcim.0c00675 (siehe S. 16–18, 24).

[Kaw11] Kawabata, Takeshi: „Build-up algorithm for atomic correspondence between chemical structures“. In: *J. Chem. Inf. Model.* 51.8 (2011), S. 1775–1782. doi: 10.1021/ci2001023 (siehe S. 2, 7, 8, 10, 11).

[Koc01] Koch, Ina: „Enumerating all connected maximal common subgraphs in two graphs“. In: *Theor. Comput. Sci.* 250.1-2 (2001), S. 1–30. doi: 10.1016/S0304-3975(00)00286-3. url: http://linkinghub.elsevier.com/retrieve/pii/S0304397500002863 (siehe S. 8, 10).

[Lar16] Larsen, Simon J.; Alkærsig, Frederik G.; Ditzel, Henrik J.; Jurisica, Igor; Alcaraz, Nicolas und Baumbach, Jan: „A simulated annealing algorithm for maximum common edge subgraph detection in biological networks“. In: *GECCO 2016 - Proc. 2016 Genet. Evol. Comput. Conf.* (2016), S. 341–348. doi: 10.1145/2908812.2908858 (siehe S. 7, 8, 10).

[Lau16] Lauck, Florian und Rarey, Matthias: „FSees: Customized Enumeration of Chemical Subspaces with Limited Main Memory Consumption“. In: *J. Chem. Inf. Model.* 56.9 (2016), S. 1641–1653. doi: 10.1021/acs.jcim.6b00117 (siehe S. 15).

[Les19] Lessel, Uta und Lemmen, Christian: „Comparison of Large Chemical Spaces“. In: *ACS Med. Chem. Lett.* 10.10 (2019), S. 1504–1510. doi: 10.1021/acsmedchemlett.9b00331 (siehe S. 10, 24, 25).

[Lev73] Levi, G.: „A note on the derivation of maximal common subgraphs of two directed or undirected graphs“. In: *CALCOLO* 9 (1973), S. 341–352 (siehe S. 2, 10).

[Lew98] Lewell, Xiao Qing; Judd, Duncan B.; Watson, Stephen P. und Hann, Michael M.: „RECAP - Retrosynthetic Combinatorial Analysis Procedure: A powerful new technique for identifying privileged molecular fragments with useful applications in combinatorial chemistry“. In: *J. Chem. Inf. Comput. Sci.* 38.3 (1998), S. 511–522. doi: 10.1021/ci970429i (siehe S. 15).

[Lip97] Lipinski, Christopher; Lombardo, Franco; Dominy, Beryl W. und Feeney, Paul J.: „Experimental and computational approaches to estimate solubility and permeability in drug discovery and development settings“. In: *Adv. Drug Deliv. Rev.* 23.1-3 (1997), S. 3–25. doi: 10.1016/s0169-409x(96)00423-1 (siehe S. 29).

[Lyu19] Lyu, Jiankun u. a.: „Ultra-large library docking for discovering new chemotypes“. In: *Nature* 566.7743 (2019), S. 224–229. doi: 10.1038/s41586-019-0917-9. url: http://dx.doi.org/10.1038/s41586-019-0917-9 (siehe S. 17, 18).

[Mar07] Marialke, J.; Körner, R.; Tietze, S. und Apostolakis, Joannis: „Graph-based Molecular Alignment (GMA)“. In: *J. Chem. Inf. Model.* 47.2 (2007), S. 591–601. doi: 10.1021/ci600387r (siehe S. 2, 8).

[May19] Mayfield, John: The Secrets of Fast Smarts Matching. 2019. url: https://nextmovesoftware.com/talks/Mayfield_SecretsOfFastSmartsMatching_Sheffield_201906.pdf (besucht am 20. 10. 2021) (siehe S. 8, 38).

[Mcc16] Mccreesh, Ciaran; Samba Ndojh, Ndiaye; Patrick, Prosser und Christine, Solnon: „Clique and Constraint Models for Maximum Common (Connected) Subgraph Problems“. In: *Princ. Pract. Constraint Program. 22nd Int. Conf. CP 2016, Toulouse, Fr. Sept. 5-9, 2016, Proc.* Hrsg. von Rueher, Michel. Bd. 9892. Cham: Springer International Publishing, 2016, S. 350–368. doi: 10.1007/978-3-319-44953-1 (siehe S. 8).

[McG20] McGann, Mark: Giga Docking - Structure Based Virtual Screening of Over 1 Billion Molecules. 2020. url: https://openeye.app.box.com/s/xup6pxgyhbom5d3ecc0e8qfv1y2epxu4 (besucht am 20. 10. 2021) (siehe S. 17).

[McG82] McGregor, James J.: „Backtrack search algorithms and the maximal common subgraph problem“. In: *Softw. Pract. Exp.* 12.1 (1982), S. 23–34. doi: 10.1002/spe.4380120103 (siehe S. 8).

[Mys12] Mysinger, Michael M.; Carchia, Michael; Irwin, John J. und Shoichet, Brian K.: „Directory of useful decoys, enhanced (DUD-E): Better ligands and decoys for better benchmarking“. In: *J. Med. Chem.* 55.14 (2012), S. 6582–6594. doi: 10.1021/jm300687e (siehe S. 11).

[Nic87] Nicholson, V.; Tsai, C.-C.; Johnson, M. und Naim, M.: „A Subgraph Isomorphism Theorem for Molecular Graphs“. In: *Graph theory Topol. Chem.* Hrsg. von King, R.B. und Rouvray, D.H. Amsterdam: Elsevier, 1987, S. 226–230 (siehe S. 2, 7, 10).

[Nij21] Nijampatnam, Bhavitavya; Ahirwar, Parmanand; Pukkanasut, Piyasuda; Womack, Holly; Casals, Luke; Zhang, Hua; Cai, Xia; Michalek, Suzanne M.; Wu, Hui und Velu, Sadanandan E.: „Discovery of Potent Inhibitors of Streptococcus mutans Biofilm with Antivirulence Activity“. In: *ACS Med. Chem. Lett.* 12.1 (2021), S. 48–55. doi: 10.1021/acsmedchemlett.0c00373 (siehe S. 26).

[OBo12] O'Boyle, Noel M: „Towards a Universal SMILES representation - A standard method to generate canonical SMILES based on the InChI“. In: *J. Cheminform.* 4.1 (2012), S. 22. doi: 10.1186/1758-2946-4-22. url: http://jcheminf.springeropen.com/articles/10.1186/1758-2946-4-22 (siehe S. 30).

[Ope] OpenEye Scientific Software Inc.: OpenEye Academic License Agreement. url: https://www.eyesopen.com/academic-license-agreement (besucht am 20. 10. 2021) (siehe S. 13).

[OTA] OTAVA CHEMICALS MB. url: https://www.otavachemicals.com/ (besucht am 20. 10. 2021) (siehe S. 15).

[OTA21] OTAVA CHEMICALS MB: CHEMriya_11bn_2021-05.space. 2021. url: https://www.biosolveit.de/infiniSee/#chemriya (besucht am 28. 05. 2021) (siehe S. 16, 26).

[Pau21] Paulsen, John: Seagate Has Now Shipped Over 3 Zettabytes of Data Storage. 2021. url: https://blog.seagate.com/enterprises/seagate-has-shipped-three-zettabytes-of-data-storage/ (besucht am 20. 10. 2021) (siehe S. 16).

[Pea06] Pearce, Bradley C; Sofia, Michael J; Good, Andrew C; Drexler, Dieter M und Stock, David A: „An empirical process for the design of high-throughput screening deck filters“. In: *J. Chem. Inf. Model.* 46.3 (2006), S. 1060–1068. doi: 10.1021/ci050504m. url: http://pubs.acs.org/doi/pdf/10.1021/ci050504m (siehe S. 29).

[Pro07] Proschak, Ewgenij; Wegner, Jörg K.; Schüller, Andreas; Schneider, Gisbert und Fechner, Uli: „Molecular Query Language (MQL) - A context-free grammar for substructure matching“. In: *J. Chem. Inf. Model.* 47.2 (2007), S. 295–301. doi: 10.1021/ci600305h (siehe S. 3, 29).

[Rar01] Rarey, Matthias und Stahl, Martin: „Similarity searching in large combinatorial chemistry spaces“. In: *J. Comput. Aided. Mol. Des.* 15.6 (2001), S. 497–520. doi: 10.1023/A:1011144622059 (siehe S. 15, 19, 27).

[Rar98] Rarey, Matthias und Dixon, J Scott: „Feature trees: A new molecular similarity measure based on tree matching“. In: *J. Comput. Aided. Mol. Des.* 12.5 (1998), S. 471–490. doi: 10.1023/A:1008068904628 (siehe S. 1, 15, 19).

[Ray02a] Raymond, John W.; Gardiner, Eleanor J. und Willett, Peter: „RASCAL: Calculation of graph similarity using maximum common edge subgraphs“. In: *Comput. J.* 45.6 (2002), S. 631–644. doi: 10.1093/comjnl/45.6.631 (siehe S. 2, 7–10).

[Ray02b] Raymond, John W. und Willett, Peter: „Maximum common subgraph isomorphism algorithms for the matching of chemical structures“. In: *J. Comput. Aided. Mol. Des.* 16.7 (2002), S. 521–533. doi: 10.1023/A:1021271615909 (siehe S. 2, 7).

[Rin13] Riniker, Sereina und Landrum, Gregory A.: „Open-source platform to benchmark fingerprints for ligand-based virtual screening“. In: *J. Cheminform.* 5.5 (2013). doi: 10.1186/1758-2946-5-26 (siehe S. 1).

[Rog10] Rogers, David und Hahn, Mathew: „Extended-Connectivity Fingerprints“. In: *J. Chem. Inf. Model.* 50.5 (2010), S. 742–754. doi: 10.1021/ci100050t. url: http://pubs.acs.org/doi/abs/10.1021/ci100050t (siehe S. 1).

[Rud12] Ruddigkeit, Lars; Van Deursen, Ruud; Blum, Lorenz C. und Reymond, Jean Louis: „Enumeration of 166 billion organic small molecules in the chemical universe database GDB-17“. In: *J. Chem. Inf. Model.* 52.11 (2012), S. 2864–2875. doi: 10.1021/ci300415d (siehe S. 17).

[San14] San Segundo, Pablo und Tapia, Cristobal: „Relaxed approximate coloring in exact maximum clique search“. In: *Comput. Oper. Res.* 44 (2014), S. 185–192. doi: 10.1016/j.cor.2013.10.018. url: http://dx.doi.org/10.1016/j.cor.2013.10.018 (siehe S. 8).

[Say20] Sayle, Roger A.; Gowers, Richard und Mayfield, John: Accelerating graph edit distance search by chemical space enumeration. 2020. url: https://nextmovesoftware.com/talks/Sayle_SmallWorld_Oxford_202003.pdf (besucht am 20. 10. 2021) (siehe S. 1, 18).

[Say97] Sayle, Roger A.: 1st-class SMARTS patterns. 1997 (siehe S. 3, 30).

[Sch] Schrödinger Inc.: Schrödinger License Agreement. url: https://www.schrodinger.com/salesagreements (besucht am 20. 10. 2021) (siehe S. 13).

[Sch10] Schomburg, Karen T.; Ehrlich, Hans Christian; Stierand, Katrin und Rarey, Matthias: „From structure diagrams to visual chemical patterns“. In: *J. Chem. Inf. Model.* 50.9 (2010), S. 1529–1535. doi: 10.1021/ci100209a (siehe S. 30, 33).

[Sch13] Schomburg, Karen T.; Wetzer, Lars und Rarey, Matthias: „Interactive design of generic chemical patterns“. In: *Drug Discov. Today* 18.13-14 (2013), S. 651–658. doi: 10.1016/j.drudis.2013.02.001. url: http://dx.doi.org/10.1016/j.drudis.2013.02.001 (siehe S. 30).

[She02] Sheridan, Robert P. und Kearsley, S K: „Why do we need so many chemical simialrity search methods?“ In: *Ddt* 7.17 (2002), S. 903–911 (siehe S. 1, 26).

[She98] Sheridan, Robert P. und Miller, Michael D.: „A method for visualizing recurrent topological substructures in sets of active molecules“. In: *J. Chem. Inf. Comput. Sci.* 38.5 (1998), S. 915–924. doi: 10.1021/ci980044f (siehe S. 2, 8).

[Sta05] Stahl, Martin; Mauser, Harald; Tsui, M und Taylor, N R: „A robust clustering method for chemical structures“. In: *J. Med. Chem.* 48.13 (2005), S. 4358–4366. doi: 10.1021/jm040213p. url: http://www.ncbi.nlm.nih.gov/pubmed/15974588%5Cnhttp://pubs.acs.org/doi/pdfplus/10.1021/jm040213p (siehe S. 1, 8, 13).

[Stu20] Stumpfe, Dagmar und Bajorath, Jürgen: „Current trends, overlooked issues, and unmet challenges in virtual screening“. In: *J. Chem. Inf. Model.* 60.9 (2020), S. 4112–4115. doi: 10.1021/acs.jcim.9b01101 (siehe S. 18).

[Sus12] Sushko, Iurii; Salmina, Elena; Potemkin, Vladimir A.; Poda, Gennadiy und Tetko, Igor V.: „ToxAlerts: A web server of structural alerts for toxic chemicals and compounds with potential adverse reactions“. In: *J. Chem. Inf. Model.* 52.8 (2012), S. 2310–2316. doi: 10.1021/ci300245q. url: https://www.surechembl.org/knowledgebase/169485-non-medchem-friendly-smarts (siehe S. 29).

[Tak87] Takahashi, Yoshimasa; Satoh, Yuzuru; Suzuki, Hajime und Sasaki, Shin Ichi: „Recognition of largest common structural fragment among a variety of chemical structures“. In: *Anal. Sci.* 3.1 (1987), S. 23–28. doi: 10.2116/analsci.3.23 (siehe S. 8).

[Urb11] Urbaczek, Sascha; Kolodzik, Adrian; Fischer, J. Robert; Lippert, Tobias; Heuser, Stefan; Groth, Inken; Schulz-Gasch, Tanja und Rarey, Matthias: „NAOMI: On the almost trivial task of reading molecules from different file formats“. In: *J. Chem. Inf. Model.* 51.12 (2011), S. 3199–3207. doi: 10.1021/ci200324e (siehe S. 7, 11, 31).

[Urb14] Urbaczek, Sascha; Kolodzik, Adrian und Rarey, Matthias: „The valence state combination model: A generic framework for handling tautomers and protonation states“. In: *J. Chem. Inf. Model.* 54.3 (2014), S. 756–766. doi: 10.1021/ci400724v (siehe S. 11, 31).

[Van13] Van Berlo, Rogier J.P.; Winterbach, Wynand; De Groot, Marco J.L.; Bender, Andreas; Verheijen, Peter J.T.; Reinders, Marcel J.T. und De Ridder, Dick: „Efficient calculation of compound similarity based on maximum common subgraphs and its application to prediction of gene transcript levels“. In: *Int. J. Bioinform. Res. Appl.* 9.4 (2013), S. 407–432. doi: 10.1504/IJBRA.2013.054688 (siehe S. 8, 10).

[Var79] Varkony, Tomas H.; Shiloach, Yossi und Smith, Dennis H.: „Computer-Assisted Examination of Chemical Compounds for Structural Similarities“. In: *J. Chem. Inf. Comput. Sci.* 19.2 (1979), S. 104–111. doi: 10.1021/ci60018a014 (siehe S. 8).

[Wei88] Weininger, David: „SMILES, a chemical language and information system. 1. Introduction to methodology and encoding rules“. In: *J. Chem. Inf. Comput. Sci.* 28.1 (1988), S. 31–36. doi: 10.1021/ci00057a005 (siehe S. 3, 30, 37).

[Whi32] Whitney, Hassler: „Congruent Graphs and the Connectivity of Graphs“. In: *Am. J. Math.* 54.1 (1932), S. 150. doi: 10.2307/2371086. url: https://www.jstor.org/stable/2371086 (siehe S. 7, 10).

[Wil98] Willett, Peter; Barnard, John M. und Downs, Geoffrey M.: „Chemical Similarity Searching“. In: *J. Chem. Inf. Comput. Sci.* 38.6 (1998), S. 983–996. doi: 10.1021/ci9800211. url: http://pubs.acs.org/doi/abs/10.1021/ci9800211 (siehe S. 1).

[Wol05] Wolber, Gerhard und Langer, Thierry: „LigandScout: 3-D pharmacophores derived from protein-bound ligands and their use as virtual screening filters“. In: *J. Chem. Inf. Model.* 45.1 (2005), S. 160–169. doi: 10.1021/ci049885e (siehe S. 1).

[WuX] WuXi LabNetwork. url: https://www.labnetwork.com/ (besucht am 20. 10. 2021) (siehe S. 15).

[WuX20] WuXi LabNetwork: GalaXi_2.1bn_2020-11.space. 2020. URL: https://www.biosolveit.de/infiniSee/#galaxi (besucht am 28. 05. 2021) (siehe S. 16, 23, 24, 26).

[Xue03] Xue, Ling; Godden, Jeffrey W.; Stahura, Florence L. und Bajorath, Jürgen: „Profile scaling increases the similarity search performance of molecular fingerprints containing numerical descriptors and structural keys". In: *J. Chem. Inf. Comput. Sci.* 43.4 (2003), S. 1218–1225. DOI: 10.1021/ci030287u (siehe S. 1).

Eigene Publikationen

Dieser Abschnitt enthält ein vollständiges Verzeichnis der eigenen Veröffentlichungen. Die Publikationen Die erste Publikation ist RIMACS [D1], danach kommt SpaceMACS [D2] und abschließend SMARTScompare mit [D3] ,[D4] und [D5].

[D1] Schmidt, Robert; Krull, Florian; Heinzke, Anna Lina und Rarey, Matthias: „Disconnected Maximum Common Substructures under Constraints“. In: *J. Chem. Inf. Model.* 61.1 (2021), S. 167–178. doi: 10.1021/acs.jcim.0c00741.

[D2] Schmidt, Robert; Klein, Raphael und Rarey, Matthias: „Maximum Common Substructure Searching in Combinatorial Make-on-Demand Compound Spaces“. In: *J. Chem. Inf. Model.* 61.X (2021), S. XXX–XXX. doi: 10.1021/acs.jcim.1c00640.

[D3] Schmidt, Robert; Ehmki, Emanuel S.R.; Ohm, Farina; Ehrlich, Hans Christian; Mashychev, Andriy und Rarey, Matthias: „Comparing Molecular Patterns Using the Example of SMARTS: Theory and Algorithms“. In: *J. Chem. Inf. Model.* 59.6 (2019), S. 2560–2571. doi: 10.1021/acs.jcim.9b00250.

[D4] Ehmki, Emanuel S.R.; Schmidt, Robert; Ohm, Farina und Rarey, Matthias: „Comparing Molecular Patterns Using the Example of SMARTS: Applications and Filter Collection Analysis“. In: *J. Chem. Inf. Model.* 59.6 (2019), S. 2572–2586. doi: 10.1021/acs.jcim.9b00249.

[D5] Ehrt, Christiane; Krause, Bennet; Schmidt, Robert; Ehmki, Emanuel S.R. und Rarey, Matthias: „SMARTS.plus – A Toolbox for Chemical Pattern Design“. In: *Mol. Inform.* 39.12 (2020), S. 1–4. doi: 10.1002/minf.202000216.

Betreute studentische Arbeiten

[S1] Heinzke, Anna Lina: „Berechnung von nicht-zusammenhängenden maximalen gemeinsamen Teilgraphen unter Randbedingungen“. Bachelorarbeit. Universität Hamburg, 2018.

[S2] Kuhrt, Tim: „Erzeugung eindeutiger molekularer Muster am Beispiel der Sprache SMARTS“. Bachelorarbeit. Universität Hamburg, 2018.

[S3] Krause, Bennet: „Webbasierte Suche nach ähnlichen chemischen Mustern am Beispiel SMARTS“. Bachelorarbeit. Universität Hamburg, 2019.

A Benutzungsanleitung der SpaceMACS Anwendung

Im Zusammenhang der Entwicklung der SpaceMACS Methode wurde eine dazugehörige Kommandozeilenanwendung entwickelt. Diese ermöglicht es einen Fragmentraum in verschiedenen Formaten zu laden und mit einer Vielzahl von Anfragen zu durchsuchen. Die SpaceMACS Anwendung kann nach dem folgenden Schema über die Kommandozeile verwendet werden:

```
./SpaceMACS [Optionen] Fragmentraumdatei Anfragen...
```

Alle Anfragen können sowohl Dateien als auch SMILES oder SMARTS Zeichenketten sein. Argumente, die keiner Kommandozeilenoption zugeordnet sind, werden als Anfrage interpretiert. Insgesamt sind die folgenden Optionen vorhanden:

-f, --fragment-space Angabe einer Datei, die einen Fragmentraum enthält. Die Formate „*.space“ und „*.fsf“ werden unterstützt. Es können mehrere Räume durch mehrfache Angabe von -f gleichzeitig durchsucht werden. Wenn nicht explizit als Kommandozeilenoption gegeben, so wird das erste positionale Argument als Fragmentraum interpretiert.

-c, --continuous Option, ob eine interaktive Sitzung gestartet werden soll. In diesem Fall ist die explizite Angabe von Anfragen nicht nötig.

-n, --nof-results Angabe der Anzahl an Ergebnissen, die für eine Anfrage ausgegeben werden sollen.

-v, --verbose Angabe über die Art der Meldungen, die vom Programm ausgegeben werden sollen.

Quiet	Keine Meldungen.
Error	Ausgabe von Fehlermeldungen.
Warning	Ausgabe von Warnungen.
Info	Ausgabe von Informationsmeldungen.

-m, --mode Angabe des verwendeten Suchmodus.

Size	Suche mit dem MCS-Size Ähnlichkeitsmaß.
Similarity	Suche mit dem MCS-Similarity Ähnlichkeitsmaß.
Substructure	Durchführen einer Substruktursuche.

-s, --skip-single-matches Option, den letzten Schritt des Algorithmus, die Suche nach Teilstrukturen innerhalb der Fragmente, auszulassen.

-l, --result-storage-limit Angabe des Parameters, wie viele Zwischenergebnisse während der Enumeration gespeichert werden sollen.

-e, --extensions-per-link-type Angabe des Parameters, wie viele Fragmente zur Absättigung offener Linker für jeden Linktyp gespeichert werden sollen.

-t, --threads Angabe des Parallelismus der Anwendung.

-i, --inner-parallelism Option, die einzelne Suche zu parallelisieren anstatt verschiedene Anfragen parallel zu suchen.

-M, --show-mapping-insertion Option die berechnete Abbildung in die SMILES Repräsentation der Ergebnismoleküle zu integrieren.

-C, --config-file Angabe einer Konfigurationsdatei, die es ermöglicht einen Aufruf unabhängig von der Kommandozeile zu parametrisieren.

-r, --ring-chain-atom-compatibility Option, Ring- und Kettenatome aufeinander abzubilden.

-o, --output Angabe einer Ausgabedatei. Wenn keine gegeben ist, wird die Standardausgabe verwendet.

--license Angabe und Aktivierung einer Lizenz für die SpaceMACS Anwendung.

-h, --help Alleinige Ausgabe der Hilfeseite und anschließende Termination des Programms.

Die SpaceMACS Anwendung ermöglicht es zusätzlich eine interaktive Sitzung auf der Kommandozeile zu starten. Auf diese Art kann ein Raum mit verschiedenen Anfragen exploriert werden, ohne diesen immer wieder neu laden zu müssen.

Die interaktive Sitzung ist in Kommandos aufgeteilt, die sich ähnlich zu den Kommandozeilenoptionen verhalten. Kommandos verwenden den Doppelpunkt als Präfix, gefolgt von einem Buchstaben, der das Kommando darstellt. Abschließend folgt der Parameter für das Kommando. Angaben, die keinem Kommando entsprechen werden als Anfragen interpretiert. Anfragen können Moleküldateien, SMILES-Zeichenketten oder SMARTS-Zeichenketten sein. Bei der Angabe von SMARTS gelten die in der Publikation beschriebenen Einschränkungen. Insgesamt werden die folgenden Kommandos akzeptiert:

:h Ausgabe der Hilfe und Übersicht über die akzeptierten Kommandos der interaktiven Sitzung.

:d Ausgabe der aktuellen Konfiguration. Diese Beinhaltet den zu durchsuchenden Fragmentraum und alle veränderlichen Optionen der Sitzung.

:i Überschreiben der „--inner-parallelism“ Kommandozeilenoption.

:l Überschreiben der „--result-storage-limit“ Kommandozeilenoption.

:e Überschreiben der „--extensions-per-link-type“ Kommandozeilenoption.

:m Überschreiben der „--mode“ Kommandozeilenoption.

:M Überschreiben der „--show-mapping-insertion“ Kommandozeilenoption.

:n Überschreiben der „--nof-results“ Kommandozeilenoption.

:r Überschreiben der „--ring-chain-atom-compatibility“ Kommandozeilenoption.

:c Angabe von Größenbeschränkungen für die Ergebnisenumeration. Es werden bis zu vier Zahlenwerte eingelesen. Ausgelassene werden als -1 interpretiert und setzen eine gegeben Einschränkung zurück. Die angegebenen Zahlen werden wie folgt interpretiert: 1.: Mindestanzahl abgebildeter Atome im MCS, 2.: Mindestanzahl an Schweratomen eines Ergebnismoleküls, 3.: Maximalanzahl an Schweratomen eines Ergebnismoleküls, 4.: Mindest-Ähnlichkeitswert eines Ergebnismoleküls.]

:s Überschreiben der „--skip-single-matches“ Kommandozeilenoption.

:t Überschreiben der „--threads“ Kommandozeilenoption.

:v Überschreiben der „--verbose“ Kommandozeilenoption.

:o Überschreiben der „--output“ Kommandozeilenoption. Angegebene Dateien werden direkt geöffnet und vorheriger Inhalt wird ohne Rückfrage überschrieben. Bei Angabe von „-“ wird die Standardausgabe verwendet.

:q Beenden der Interaktiven Sitzung und Termination der Anwendung.

Weitere Informationen und Beispiele zur Benutzung von SpaceMACS liegen in der Datei README, die im entsprechenden SpaceMACS-Softwarepaket enthalten ist.

B Benutzungsanleitung der SMARTScompare Anwendungen

Im Zusammenhang der Entwicklung der SMARTScompare Methode wurde eine dazugehörige Kommandozeilenanwendung entwickelt und eine graphische Anwendung erweitert. Beide ermöglichen einen Vergleich von SMARTS Mustern. Die Kommandozeilenanwendung ist dazu geeignet, große Sammlungen von SMARTS zu verarbeiten. Mit der graphischen Anwendung kann der Vergleich und das berechnete Ergebnis im Detail interaktiv nachvollzogen werden. Die SMARTScompare Anwendung kann nach dem folgenden Schema über die Kommandozeile verwendet werden:

```
./SMARTScompare [Optionen] SMARTS-Dateien oder SMARTS-Zeichenketten...
```

Insgesamt werden die folgenden Optionen akzeptiert:

-h, --help Alleinige Ausgabe der Hilfeseite und anschließende Termination des Programms.

-v, --verbose Option mehr Informationen auf der Kommandozeile über den Programmablauf auszugeben.

-m, --mode Legt den Modus für den Vergleich der SMARTS fest.

1 Identitätssuche.
2 Teilmengenrelation ausgehend von ersten SMARTS.
3 Teilmengenrelation ausgehend vom zweiten SMARTS.
4 Ähnlichkeitssuche.

-n, --mapping-normalisation Angabe über die Art, wie aus der Größe der MCS-Abbildung ein Ähnlichkeitswert berechnet werden soll.

1 Tanimoto-Ähnlichkeit.
2 Maximum der Knotenanzahl beider SMARTS.
3 Durchschnittliche Knotenanzahl beider SMARTS.
4 Minimum der Knotenanzahl beider SMARTS.
5 Knotenanzahl des ersten SMARTS.
6 Knotenanzahl des zweiten SMARTS.

-V, –valence-state-statistics Angabe einer Anwendungsspezifischen Statistik über die Verteilung der Valenzzustände für eine angepasste Ähnlichkeitsbewertung.

--license Angabe und Aktivierung einer Lizenz für die SMARTScompare Anwendung.

-f, --files Explizite Angabe von Eingabedateien.

-s, --smarts Explizite Angabe von SMARTS Ausdrücken.

-p, --parallel Anzahl der zu benutzenden Threads für den paarweisen Vergleich.

-t, --thres Filterwert für die Ausgabe von Ergebnissen.

-A, --no-aromatic-ring-bond-detection Option die Aromatizitätserkennung auf Ringsystemen zu unterlassen. Es werden keine aromatischen Zustände in die SMARTS Muster eingefügt.

-N, --no-kekule-correction Option die Bereinigung aliphatischer Bindungszustände ein kleinen Ringen, bei denen jedes Knoten alleinig aromatisch ist, zu unterlassen.

--no-statistical-scoring Option die Ähnlichkeitswerte alleinig auf den ungewichteten Zuständen in den Fingerabdrücken der Knoten zu berechnen.

-u, --use-header Option die Kommandozeilenausgabe mit einer Kopfzeile zu beginnen.

-l, --score-level Angabe, auf welche Art die Fingerabdrücke der Abbildungspaare bewertet werden sollen.
1 Ähnlichkeit der Elementfingerabdrücke.
2 Mit der Valenzzustandsstatistik gewichtete Fingerabdruckähnlichkeit.

-M, --maximum-results Maximale Anzahl an Ergebnissen, die für jeden SMARTS ausgegeben werden.

-S, --single-line-output Option, für jeden SMARTS die Ergebnisse in einer Zeile auszugeben.

-d, --delimiter Trennzeichen zwischen den einzelnen Ergebniseinträgen innerhalb einer Zeile.

-D, --Delimiter Trennzeichen zwischen einem gefundenen SMARTS und der Beschreibung innerhalb eines Ergebniseintrags.

Die SMARTScompare Kommandozeilenanwendung betrachtet die Eingabe als eine Liste von Dateien. Sofern SMARTS-Ausdrücke explizit angegeben werden, stellen diese die erste Datei dar. Grundsätzlich werden alle SMARTS der ersten Datei mit denen aller anderen verglichen. Sollte SMARTScompare nur eine Datei betrachten, so werden die SMARTS dieser Datei mit allen anderen verglichen. Aus dieser Modellierung leiten sich auch die Angaben „Teilmengenrelation ausgehend vom ersten/zweiten SMARTS“ ab.

Zuzüglich der Kommandozeilenanwendung wurde die bestehende graphische Anwendung SMARTSviewer modifiziert und zum SMARTScompareViewer weiterentwickelt. Diese akzeptiert die folgenden Kommandozeilenparameter:

-h, --help Alleinige Ausgabe der Hilfeseite und anschließende Termination des Programms.

-s, --smarts Angabe von SMARTS die visualisiert werden sollen.

-o, --outfile Ausgabedatei, in die die Visualisierung der Eingabe-SMARTS Muster gespeichert werden soll. Als Dateiformate werden „*.png“ und „*.svg“ unterstützt.

-d, --dimension Angabe der Größe des gespeicherten Bildes, sofern eine png-Datei als Ausgabe vorgesehen ist.

-p, --parameter

-m, --mode Legt den Modus für den Vergleich der SMARTS fest.
1 Identitätssuche.
2 Teilmengenrelation ausgehend von ersten SMARTS.
3 Teilmengenrelation ausgehend vom zweiten SMARTS.
4 Ähnlichkeitssuche.

-A, --no-aromatic-ring-bond-detection Option die Aromatizitätserkennung auf Ringsystemen zu unterlassen. Es werden keine aromatischen Zustände in die SMARTS Muster eingefügt.

-N, --no-kekule-correction Option die Bereinigung aliphatischer Bindungszustände ein kleinen Ringen, bei denen jedes Knoten alleinig aromatisch ist, zu unterlassen.

--license Angabe und Aktivierung einer Lizenz für die SMARTScompareViewer Anwendung.

Der SMARTScompareViewer hat zwei verschiedene Einsatzzwecke. Einerseits kann mit der graphischen Anwendung interaktive ein Vergleich von SMARTS analysiert werden. Andererseits können über die Kommandozeile einzelne Vergleiche auch als Bild gespeichert werden. Die Information über den Ähnlichkeitswert erfolgt in der graphischen Anwendung im Informationsfenster unterhalb der Visualisierung und im Kommandozeilenmodus über die Standardausgabe.

Weitere Informationen und Beispiele zur Benutzung von SMARTScompare und dem SMARTScompareViewer liegen in der Datei README, die im entsprechenden SMARTScompareViewer-Softwarepaket enthalten ist. Das Paket enthält beide Anwendungen und eine weitere, die es ermöglicht Valenzzustandsstatistiken zu erzeugen.

C Publikationen

C.1 Beiträge zu Artikeln der kumulativen Dissertation

[D1] Schmidt, Robert u. a.: „Disconnected Maximum Common Substructures under Constraints“. In: *J. Chem. Inf. Model.* 61.1 (2021), S. 167–178. DOI: 10.1021/acs.jcim.0c00741
Das initiale Konzept des Algorithmus wurde von F. Krull währen einer Masterarbeit erarbeitet. Die Evaluation des eingeschränkten Nichtzusammenhangs wurde von A. L. Heinzke mit einem anderen Algorithmus in einer Bachelorarbeit durchgeführt. R. Schmidt hat die Approximationen der Ergebnisgüte und Heuristiken zur Beschleunigung des exakten Verfahrens implementiert und evaluiert. R. Schmidt hat die beschriebenen Experimente durchgeführt.

[D2] Schmidt, Robert u. a.: „Maximum Common Substructure Searching in Combinatorial Make-on-Demand Compound Spaces“. In: *J. Chem. Inf. Model.* 61 (2021). DOI: 10.1021/acs.jcim.1c00640
M. Rarey und R. Schmidt haben das Konzept erarbeitet und R. Schmidt hat die Methode und die Anwendung programmiert und validiert. R. Klein hat das Potential der SpaceMACS Kommandozeilenprogramms in einem beispielhaften Anwendungsszenario der medizinischen Chemie demonstriert und mit mit existierenden Applikationen zur Suche in Fragmenträumen verglichen.

[D3] Schmidt, Robert u. a.: „Comparing Molecular Patterns Using the Example of SMARTS: Theory and Algorithms“. In: *J. Chem. Inf. Model.* 59.6 (2019), S. 2560–2571. DOI: 10.1021/acs.jcim.9b00250
A. Mashychev und H-C. Ehrlich haben in der Masterarbeit von A. Mashychev ein Konzept zum Vergleich von SMARTS initial entwickelt. Dieses wurde von F. Ohm in einer Bachelorarbeit erneut aufgegriffen und zu eine Demonstration auf einfachen Teilmengenrelationen weiterentwickelt. R. Schmidt und M. Rarey haben die Konzepte zur Bereinigung der Fingerabdrücke, zur Integration der SMARTS Rekursion und die Verwendung eines MCS-Algorithmus entwickelt. R. Schmidt hat die Konzepte implementiert und validiert. E. S. R. Ehmki die entwickelte Anwendung in den beschriebenen Szenarien angewendet.

[D4] Ehmki, Emanuel S.R. u. a.: „Comparing Molecular Patterns Using the Example of SMARTS: Applications and Filter Collection Analysis“. In: *J. Chem. Inf. Model.* 59.6 (2019), S. 2572–2586. DOI: 10.1021/acs.jcim.9b00249
R. Schmidt und M. Rarey haben das Konzept zur Ähnlichkeitsberechnung auf SMARTS und für die Behandlung von Aromatizität entwickelt. R. Schmidt hat diese implementiert. F. Ohm hat in ihrer Bachelorarbeit initiale Untersuchungen auf den verwendeten Sammlungen von SMARTS-Mustern durchgeführt. E. S. R. Ehmki hat die Anwendungsbeispiele Entwickelt und ausführlich evaluiert.

[D5] Ehrt, Christiane u. a.: SMARTS.plus – A Toolbox for Chemical Pattern Design. 2020. DOI: 10.1002/minf.202000216
In diesem Artikel wird eine Übersicht über die Verknüpfung einzelnen Methoden auf SMARTS und deren Integration in einen Webserver gegeben. B. Krause hat die von R. Schmidt entwickelten Anwendungen aus [D3] und [D4], sowie eine Anwendung zur automatischen Erzeugung von SMARTS-Mustern[Bie15] in den Webserver integriert. R. Schmidt hat die verwendete Strategie zur Auswahl relevanten Muster entwickelt. C. Ehrt hat das beschriebene Anwendungsszenario entworfen und durchgeführt.

M. Rarey betreute alle zuvor genannten Projekte und war an der Konzipierung der Methoden und Evaluierungen beteiligt.

D Publikationen der kumulativen Dissertation

Disconnected Maximum Common Substructures under Constraints

[D1] Schmidt, Robert u. a.: „Disconnected Maximum Common Substructures under Constraints“. In: *J. Chem. Inf. Model.* 61.1 (2021), S. 167–178. DOI: 10.1021/acs.jcim.0c00741

http://pubs.acs.org/articlesonrequest/AOR-ICP8QHWSVZRIKPSCCVRB

JCIM JOURNAL OF CHEMICAL INFORMATION AND MODELING

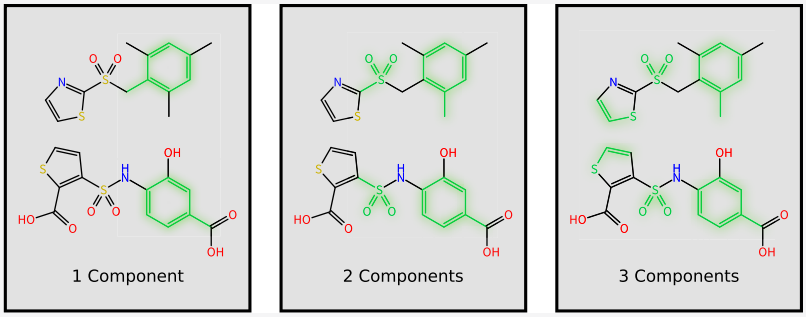
pubs.acs.org/jcim Article

Disconnected Maximum Common Substructures under Constraints

Robert Schmidt, Florian Krull, Anna Lina Heinzke, and Matthias Rarey*

Cite This: J. Chem. Inf. Model. 2021, 61, 167−178 Read Online

ACCESS | Metrics & More | Article Recommendations | Supporting Information



ABSTRACT: The maximum common substructure (MCS) problem is an important, well-studied problem in cheminformatics. It is applied in several application scenarios like molecule superimposition and scaffold detection or as a similarity measure in virtual screening and clustering. In many cases, the connected MCS is preferred since it is faster to calculate and a highly fragmented MCS is not very meaningful from a chemical point of view. Nevertheless, a disconnected MCS (dMCS) can be very instructive if it consists of reasonably sized molecular parts connected by variable groups. We present a new algorithm named RIMACS, which is able to calculate the dMCS under constraints. We can control the maximum number of connected components and their minimal size using a modified local substructure mapping approach. A formal proof of correctness is provided as well as extended runtime evaluations on chemical data. The evaluation of RIMACS shows that a small number of connected components helps us to improve MCS similarity in a meaningful way while keeping the runtime requirements in a reasonable range.

INTRODUCTION

The maximum common substructure problem is a well-known NP-hard problem that has many uses in cases of drug discovery ranging from similarity searching and diversity clustering to reaction mapping.[1,2] Several approaches have been proposed and evaluated for specific applications.[3−11] Heuristic approaches[5,9] usually have the advantage of much faster computation times but cannot guarantee to find the optimum result.

In order to reduce problem complexity and improve practical performance, exact algorithms for the MCS problem are often limited to find only connected substructures.[8] From the application point of view, disconnected maximum common substructures might find several mappings of single atoms or bonds. Since the relative arrangement of the mapped components is neglected, the result might not be meaningful and overestimate compound similarity. On the other hand, connected maximum common substructures have their disadvantages as well. Single mismatching atoms within a structure might result in substantially underestimated similarity. Therefore, Stahl et al. already suggested heuristic approaches to overcome the connectivity constraint.[9,12] They computed a dMCES, which they name maximum overlapping set (MOS). Based on the arrangements of the largest three components of the dMCES, penalty terms were added to correct the similarity value.

From the point of algorithms, the MCS problem is commonly solved using clique-based approaches.[1,3,5,8,12−15] The basic idea is to transfer the MCS problem to the maximum clique problem by a compatibility graph (or association graph) of the two input graphs. Compatibility graphs are usually dense and clique-based algorithms try to cope with this describing them as a bit-graph.[15] If the input graphs are sparse, as in the case of molecule graphs, the compatibility graph can be implemented as nodes with complementary adjacency lists (i.e., storing the nodes not connected rather than those connected) resulting in less memory consumption.[5] Heuristic approaches use, e.g., local iterations, to efficiently find good approximations.[5,16] Exact

Received: June 30, 2020
Published: December 16, 2020

approaches are often based on the Bron−Kerbosch[17] algorithm,[8,13] handling the current clique and sets of nodes for extension and forbidden nodes. More advanced approaches use heuristic graph colorings as internal upper bound estimations[1,15] to guide the branch and bound procedure. One of the more prominent clique-based approaches, RASCAL,[18] uses random partitions into independent sets as its main efficiency-enhancing trick.[14]

The disadvantage of these methods is that they require a large compatibility graph to be calculated and stored. Contrary to the use of cliques, more structure-focused approaches[4,7,9,19,20] avoid this step, usually resulting in less memory consumption and shorter runtimes.

To respect intuitive similarity and increase the chemical feasibility of the results, the disconnected MCS can be restricted to obey topological distances.[9,14] Marialke et al.[14] invented this as "embedded subgraph isomorphism" for the dMCS problem. They were initially implemented in a clique-based algorithm, increasing performance and chemical feasibility. Kawabata[9] based his build-up algorithm on topological distances as well. Beyond topological distances, it is possible to constrain the result with a maximum number of connected components of a minimum size.[4,6] The first approach mentioned by Cao et al.[4] is substructure-based, and the latter by Hariharan et al.[6] is a clique-based method, also suitable for the multi-MCS problem. They handle the multi-MCS problem using all maximal cliques of the pairwise MCS of a pivot molecule to all other compounds.

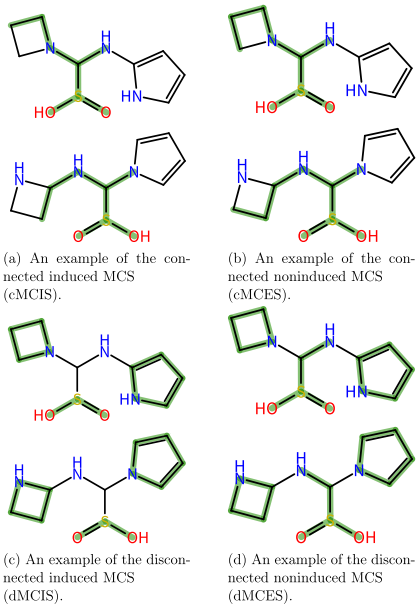
In this paper, we address the problem of efficiently calculating the disconnected MCS obeying an upper bound for the number of connected components with each connected component following a lower bound for its size. Both types of constraints are mentioned in the literature but have never been evaluated in detail before. For better performance, we adapt an incremental matching approach (see also the work of Cao et al.[4]) rather than a clique-based method. We further introduce simple and efficient upper bound estimations improving the performance of the backtracking scheme. Finally, our algorithm RIMACS is exact with a proof of correctness given in the Supporting Information. With several chemical datasets, we show the practical performance of the algorithm and evaluate the influence of the runtime from the maximum number of components and their minimal size.

METHODS

MCS Problem and Its Variants. In the following, molecules are modeled as labeled, undirected simple graphs (no parallel edges or loops) G whereas $V(G)$ are the nodes or atoms and $E(G)$ are the edges between nodes or bonds between atoms. A graph $G' \subseteq G$ is called an induced subgraph of G if there exists an injective function $f:V(G')\rightarrow V(G)$ such that $\forall u, v \in V(G'):\{u, v\} \in E(G')\Leftrightarrow\{f(u), f(v)\} \in E(G)$. A common induced substructure S of two graphs G_1 and G_2 is a subgraph of both graphs, i.e., $S \subseteq G_1 \wedge S \subseteq G_2$. The maximum common induced substructure S_{max} is defined to be a common induced substructure of maximum size. A substructure is maximal if and only if it cannot be extended.

In cheminformatics, the non-induced maximum common substructure problem is deemed to be more relevant. Non-induced maximum common substructures are defined with less restrictions: $f:V(G')\rightarrow V(G):\forall\{u, v\} \in E(G')\Rightarrow\{f(u), f(v)\} \in E(G)$. Furthermore, in cheminformatics, maximizing the number of common edges is more relevant in practice, resulting in the so-called MCES problem. Interestingly, the MCES problem can be reduced to an induced MCS problem using a so-called line graph.[18,21,22] In this paper, we will therefore focus on the induced MCS problem. We refer to ref 1 for a detailed discussion on MCS problem variants and their relationships. Additionally, we describe our efforts to avoid so-called ΔY exchanges to guarantee valid mappings if the MCES is computed in the Supporting Information.

With respect to semantics and runtime as well, it is important to differentiate between the connected and disconnected MCS. The connected maximum common substructure problem (cMCS) restricts the common subgraph to exactly one connected component. The disconnected problem variant without limitation on the number of connected components will be called dMCS. The four common problem instances for the MCS are shown in Figure 1.



(a) An example of the connected induced MCS (cMCIS).

(b) An example of the connected noninduced MCS (cMCES).

(c) An example of the disconnected induced MCS (dMCIS).

(d) An example of the disconnected noninduced MCS (dMCES).

Figure 1. Four common cases of the MCS problem. The two examples at the top show the cMCS, whereas the two lower examples show the dMCS. On the left, the examples represent the MCIS, and on the right, the MCES. The difference in inducedness concerns adjacency of atoms in both molecules. The examples in each row illustrate this difference.

To distinguish the two input graphs in a reasonable way, in the following, the first graph will be called query graph Q since the backtracking scheme selects its nodes to be mapped next. Accordingly, the second graph will be called target graph T. Note that the results do not depend on the assignment which graph is Q and which one is T. For efficiency reasons, we assume that Q has the smaller number of nodes of the two input graphs.

Calculating the MCS by Incremental Matching. The RIMACS (recursive incremental maximum common substructure) algorithm is inspired by subgraph matching algorithms like VF2.[23] The VF2 algorithm is based on local extensions building a mapping between the graph nodes using

https://dx.doi.org/10.1021/acs.jcim.0c00741
J. Chem. Inf. Model. 2021, 61, 167−178

index pairs. The backtracking scheme is organized as a depth first search on the graph structure of the query graph. Especially at the point of the feasibility tests of the VF2 algorithm, RIMACS employs upper-bound estimations since the MCS problem is more general than the subgraph isomorphism problem.

In contrast to other substructure-based MCS algorithms using subgraph enumeration,[7] this approach can be easily adapted to calculate the dMCS. In Figure 2, we present our pseudo-code that is highly similar to the algorithm presented by Cao et al.[4]

```
MCS()
1  return MCS-PROCEEDWITHNEXTCOMPONENT(∅, ∅, ∅)

MCS-PROCEEDWITHNEXTCOMPONENT(M, F, M_CurrMax)
1  for v ∈ V(Q) \ F
2      M_CurrMax = MCS-INCREMENTAL(M, {v}, F, M_CurrMax)
3      F = F ∪ {v}
4  return M_CurrMax

MCS-INCREMENTAL(M, C, F, M_CurrMax)
1  if C == ∅
2      if COMPONENTCONSTRAINTSVALID(M)
3          if MinSize ≤ |V(Q) \ F| ∧ NOFCOMPONENTS(M) < MaxComp
4              M_CurrMax = MCS-PROCEEDWITHNEXTCOMPONENT(M, F, M_CurrMax)
5          elseif |M_CurrMax| < |M|
6              M_CurrMax = M
7      return M_CurrMax
8  v = SELECTIONSTRATEGY1(C)
9  C = C \ {v}
10 P = SELECTCOMPATIBLENODES(M, v, F) // P is sorted by SELECTIONSTRATEGY2
11 for p ∈ P
12     F_next = UPDATEFORBIDDENNODES(M, F, (v, p))
13     C_next = (C ∪ v.adjacentNodes) \ F_next
14     if |M_CurrMax| < UPPERBOUND(M ∪ {(v, p)}, C_next, F_next)
15         M_CurrMax = MCS-INCREMENTAL(M ∪ {(v, p)}, C_next, F_next, M_CurrMax)
16         if(UPPERBOUND(M, C, F) ≤ |M_CurrMax|) break
17 return MCS-INCREMENTAL(M, C, F ∪ {v}, M_CurrMax)
```

Figure 2. Main algorithm including the setup of parameters for the mapping M, the next extension candidates C, the forbidden nodes F, and the maximum mapping M_{CurrMax} including handling of the dMCS. The graphs Q and T as well as the component constraints are global variables. The basic loop MCS-INCREMENTAL is optimized for a connected extension. Combined with the processing of further components, it results in an algorithm that is also capable of the dMCS. All remaining functions used are briefly described in Figure 3.

The starting point for the algorithm is a backtracking loop for a connected extension of a mapping of nodes from the query graph Q to the target graph T. Given the mapping M, the candidate nodes for extension $C \subseteq V(Q)$, the forbidden nodes for extension $F \subseteq V(Q)$, and the currently known maximum mapping M_{CurrMax}, the basic algorithm is organized as follows. If there are no candidates left, the result M_{CurrMax} is maximized with the current mapping M. Otherwise, any candidate node $v \in C$ is chosen for extension. Then, the mapping is extended using all available mapping partners, $p \in V(T)$, that are compatible to v and respect adjacency in Q and T. This results in the main recursion. Finally, the mapping is extended without mapping v. v is now a forbidden node.

The support of disconnected mappings requires one minor modification. If there is no connected extension possible but there are still nodes that have mapping partners available, then any of them is selected as a candidate and the backtracking scheme is continued.

To improve performance of the RIMACS algorithm calculating a dMCS and increase the chemical feasibility of the results, we suggest to employ two additional constraints.

MaxComp: An upper bound for the number of connected components (as mentioned by Cao et al.[4]).

MinSize: A lower bound for the number of matches in each connected component (as mentioned by Hariharan et al.[6]).

Note that all variables and parameters in the pseudo-code (see Figures 2 and 3) have the same meaning as described above. The graphs Q and T as well as the component constraints MaxComp and MinSize are global read-only variables.

The performance of this backtracking scheme is mostly influenced by three functions. The UPPERBOUND function calculates an upper bound on the reachable MCS size. The tighter this bound is, the more partial solutions can be pruned. SELECTIONSTRATEGY1 and SELECTIONSTRATEGY2 determine the order in which the matching is extended. Good selection strategies will result in finding large matchings and achieving high lower bounds for the result early. Note that SELECTIONSTRATEGY2 can be precalculated before the incremental search starts.

Extensions to the Basic Algorithm. The algorithm can be easily extended to support node weights or result in arrangement properties like those proposed by Stahl et al.[12] Since the mapping components are already known, one can easily test for their arrangement. This could influence similarity or restrict the result to enforce the same arrangement of connected components.

The performance of clique-based algorithms can be significantly improved using topological distances.[9,14] They can be included into the compatibility graph, thus improving runtime behavior and increasing the chemical feasibility of the results. Topological distances can be included into substructure-based algorithms like RIMACS as well, but an inclusion into the model as it is for clique-based algorithms is not possible.

The algorithm in the presented form is designed and able to calculate the maximum MCS. The calculation of all maximal MCS,[8,13] i.e., common substructures, which cannot be extended further, is possible in some cases. If the MCS is required to be connected, it is sufficient to differentiate the forbidden nodes F in unmappable and finished nodes. Nodes that were selected for extension before have to be considered as finished. The remainder of F is simply unmappable. A connected component is maximal if there are no candidates left and none of the finished nodes adjacent to any mapped node has mapping partners available.

This can be adapted for the calculation of the dMCS as long as there is no lower bound constraint on the component size. In the general case, namely, the calculation of all maximal dMCS with a non-trivial minimum number of nodes, we are not able to present an efficient algorithmic solution.

Upper Bounds for the Achievable MCS. This kind of substructure-based MCS algorithms offers a trivial upper bound at each stage of the calculation by counting the unmapped nodes in the query graph.

To improve the upper bound in the case of the constrained dMCS, we propose tracking the connected components in the query graph using the existing node mapping M and the remaining mappable nodes $V(Q)\backslash F$. By removing the mapped and forbidden nodes, Q decays into a set of connected components. Nodes of components adjacent to mapped nodes are always counted for the upper bound. All remaining components are sorted decreasingly by the number of nodes. The nodes of the largest components are counted for the bound until MaxComp components have been reached or the node count is below the threshold MinSize. We will refer to

https://dx.doi.org/10.1021/acs.jcim.0c00741
J. Chem. Inf. Model. 2021, 61, 167−178

ComponentConstraintsValid(M)
// Ensure that each connected component of the mapping contains at least
// $MinSize$ nodes and there are at most $MaxComp$ connected components

NofComponents(M)
// Returns the number of connected components of the mapping M

SelectionStrategy1(C)
// Returns the most promising node $v \in V(Q)$ for the next mapping extension.

SelectionStrategy2(P)
// Sort the list of compatible nodes $P \subseteq V(T)$ for a node $v \in V(Q)$.
// This aims to prefer $p \in P$ similar to v when extending M with (v, p).

SelectCompatibleNodes(M, v, F)
// Select all available mapping partners $t \in V(T)$ for a node $v \in V(Q)$.
// Selection follows the predefined order of SelectionStrategy2.
// Here, the definition of available nodes of UpdateForbiddenNodes is used as well.

UpdateForbiddenNodes($M, F, (v, p)$)
// Extends F with v and ensures for all nodes $u \in V(Q) \setminus F$ that
// there is at least one possible mapping partner $t \in V(T)$ available.
// $t \in V(T)$ is available if and only if the induced subgraphs of Q and T described by
// $M \cup \{(v, p)\} \cup \{(u, t)\}$ are isomorphic. F is extended with those nodes u that have
// no such t available. The incremental isomorphism test is implemented assuring identical
// adjacency relations of (u, t) to all mapping pairs in $M \cup \{(v, p)\}$

UpperBound(M, C, F)
// Calculate an upper bound for the maximum achievable result size based on the current result.
// Currently only the nodes of the query graph are used for the upper bound estimation.

Figure 3. Brief descriptions of all helper functions used by the algorithm. The functions take the current mapping M, the candidate nodes $C \subseteq V(Q)$, forbidden nodes for extension $F \subseteq V(Q)$, or the possible mapping partners $P \subseteq V(T)$ for a node $v \in V(Q)$ as a parameter.

the resulting value as an upper estimation with component tracking (CT).

State Space Traversal Strategies. For an exact MCS algorithm, the whole state space of possible mappings must be covered. If any part of the state space can be proven to be non-optimal, the algorithm can still ignore it as part of its branch and bound procedures. Efficient algorithms therefore try to avoid most of the state space by identifying promising regions first such that the lower bound for the optimum is high early on.

The presented algorithm relies on the following state space traversal strategies.

Low Variability First (LVF). Start mapping nodes with the least number of available mappings. This strategy keeps the "degree" of the state space search tree low in the regions close to its root.

Maximize Components First (MCF). Extend connected components as long as possible. Due to the limited number of components, large common substructures require large connected components.

Similar Partners First (SPF). Inspired by the paper by Englert and Kovács,[5] the algorithm uses node similarities as an order criterion for node mapping partners. In this approach, the compatible nodes are sorted by their difference in degree, the difference of the sum of the degree of all neighbors, and their atom similarity score. The atom similarity score is the Tanimoto similarity of the ECFP6-fingerprint bits centered at the atoms.

Relative Node Positions (RNP). For each node, the maximum distance in the number of edges to any other node is calculated. The distances are transformed into ranks and normalized onto the interval [0.5,1]. These values, named relative positions, are integrated into the SPF strategy in the following way: The relative position similarity for two nodes is:

$$\mathrm{sim}_{\mathrm{RNP}}(n_1, n_2) = 1 - \mathrm{abs}(\mathrm{relative\_position}(n_1) - \mathrm{relative\_position}(n_2))$$

If the RNP strategy and the SPF strategy are applied both, the similarity values of both strategies are multiplied.

If applying the strategies does not result in a unique order, the order of input is used to resolve conflicts.

CORRECTNESS

A proof of correctness for the basic algorithm and all extensions is given in the Supporting Information. Here, we want to give only a brief outline of the strategy. The proof is organized in the following parts.

First, it is shown that the MCS-INCREMENTAL function can enumerate any connected mapping M_c with no further candidates for mapping extension under some constraints. This includes correct handling of the candidate list C and the forbidden nodes F.

This ability to enumerate M_c is combined with validity of the upper bound estimation, which is key to prove the correct calculation of the cMCIS. To prove the correctness for the dMCIS, it is additionally necessary to show that MCS-INCREMENTAL can enumerate any mapping M under some constraints.

The proof of correctness for the MCES is primarily based on the work of Whitney[21] and Nicholson et al.[22] We extend their proofs for graph isomorphism to connected components of a dMCES.

RESULTS

The RIMACS algorithm solely computes an MCS represented as a list of index pairs representing the mapped nodes of Q to T. To perform the following benchmarks, we implemented simple adapters wrapping NAOMI[24] molecules. For optimal

https://dx.doi.org/10.1021/acs.jcim.0c00741
J. Chem. Inf. Model. 2021, 61, 167−178

Table 1. Overview of Compound Sets Used for Benchmarking[a]

name	EvalSet	NCI-Key	NCI1[5]	NCI2[5]	NCI3[5]	NCI-S[5]	NEU[9]	CAH2[9]	CDK2[9]
N	100	535	32	16	16	39	15	111	154
n_{min}	2	5	24	37	71	20	20	8	9
n_{max}	43	76	24	46	78	45	28	28	36
n_{avg}	26.6	29.9	24.0	42.5	74.8	30.4	21.7	19.7	24.7
n_{std}	7.84	8.83	0.00	2.47	2.17	6.50	2.11	4.80	5.40
s_{CIMin}	0.02	0.01	0.14	0.12	0.03	0.13	0.26	0.03	0.03
s_{CNMin}	0.00	0.00	0.14	0.14	0.03	0.13	0.26	0.00	0.00
s_{CIMax}	1.00	0.88	0.92	1.00	0.97	0.96	0.91	1.00	0.97
s_{CNMax}	1.00	1.00	1.00	1.00	0.97	1.00	0.95	1.00	0.97
s_{CIAvg}	0.17	0.18	0.31	0.28	0.25	0.38	0.53	0.29	0.22
s_{CNAvg}	0.18	0.19	0.33	0.31	0.25	0.38	0.56	0.30	0.23
s_{CIStd}	0.07	0.09	0.18	0.20	0.24	0.23	0.17	0.15	0.11
s_{CNStd}	0.07	0.10	0.20	0.19	0.24	0.23	0.17	0.15	0.11
s_{DIMin}	0.02	0.03	0.55	0.35	0.32	0.34	0.40	0.14	0.16
s_{DNMin}	0.00	0.00	0.60	0.45	0.25	0.39	0.40	0.00	0.00
s_{DIAvg}	0.41	0.37	0.69	0.57	0.50	0.65	0.64	0.48	0.46
s_{DNAvg}	0.47	0.47	0.85	0.72	0.58	0.68	0.71	0.48	0.54
s_{DIStd}	0.11	0.12	0.08	0.13	0.14	0.12	0.11	0.12	0.11
s_{DNStd}	0.16	0.17	0.10	0.10	0.18	0.12	0.12	0.16	0.14

[a]Rows give the number of molecules (N) in the set, $n_{min,max,\ avg,\ std}$ rows describe the min., max., and avg. number of heavy atoms per molecule and their deviation. The s rows describe similarity whereby s_C means connected and s_D refers to disconnected computations. Furthermore, the MCS is either induced like s_{CI} or noninduced s_{CN}. Finally, the suffix describes whether it is the min., max., or average similarity or the corresponding standard deviation. On each compound set, we performed $\frac{N(N-1)}{2}$ comparisons. The CAH2 set contains three metallo-organic compounds, and they were omitted because they are not supported within the NAOMI library.

performance, only heavy atoms are covered by default. If required, the adapters can support hydrogen atoms as they are explicitly modeled within NAOMI. For the benchmarks performed, we used the following datasets.

We compiled a small compound set from the first 25 DUD-E[25] targets. This set contains the first and last active and decoy from the selected targets. This set is named EvalSet later on. It should represent similar and dissimilar compounds alike as they are taken from actives and decoys. Furthermore, we use the compound sets used by Englert and Kovács[5] named NCI* and three sets from Kawabata[9] that were used by Englert as well. Finally, we extracted another set from the NCI-Release of 2012,[26] named NCI-Key. For this set, all compounds with IDs ending in 425 or 545 are taken. This set should present a reasonable-sized, reproducable, arbitrary selection of compounds of this dataset. Overall, the datasets represent diverse scenarios from similar to grouped similar to dissimilar collections as they may occur in real-life applications.

An overview of the properties of the compound sets is shown in Table 1. The table provides the minimum and maximum number of heavy atoms as well as several similarity values to distinguish the compound sets.

Test Setup. All calculation were performed on our in-house cluster on Intel Xeon E5-4620 2.20GHz CPUs with 32 cores and 264 GB RAM running a 64 Bit Suse-Linux. Parallelization was implemented by running independent MCS calculations of different molecules in several threads whereby each calculation is single-threaded.

First, we evaluated our upper bound estimation, CT, with four different combinations for the two constraint types "Max. Number of Components" and "Min. Component Size" (MaxComp/MinSize). In the following plots, primarily algorithm runtimes are used for comparison.

The results are shown in Table 2. For the cMCIS, the mean runtime improved by a factor of 2. Even the constrained

Table 2. Mean Runtimes in ms of the Induced Pairwise Comparison of the Compounds from the EvalSet[a]

constraints	mean runtimes in ms	
MaxComp/MinSize	without CT	with CT
connected	7.44	3.63
3/3	2885	447
10/1	29,153	39,862
10/10	21.8	6.9

[a]The cMCIS and the constrained dMCIS profit from component tracking. For the 10/1 dMCS, CT has a worse performance than the naive variant without CT.

disconnected MCIS is improved by this estimation. If only a few components are allowed, the performance can increase beyond a factor of 5. For the unconstrained dMCS, the component tracking cannot improve any upper bound estimation, thus it has no positive impact on the runtimes. In fact, runtimes increased by 1/3. This indicates that, for the unconstrained dMCS, CT as an upper-bound estimation becomes irrelevant.

The Compatible Node Order Strategies. In the following, we will analyze the impact of mapping similar partners first, which is similar to the ECFP strategy proposed by Englert and Kovács.[5] There are some differences between the strategies since we only evaluate ECFP-fingerprint bits up to radius three, while Englert and Kovács use an unlimited radius. Our strategy is implemented such that all similarities are calculated before backtracking starts.

For RIMACS, the SPF and RNP strategy alone have no significant influence on the overall runtimes. Applying them together, however, usually results in performance improvements. Although our SPF strategy and Englert's ECFP strategy significantly differ in radius, we want to point out that heuristic approaches benefit more reliably from using ECFP fingerprint

https://dx.doi.org/10.1021/acs.jcim.0c00741
J. Chem. Inf. Model. 2021, 61, 167−178

bits as an order criterion for compatible atoms. Beyond that, the RIMACS algorithm is an exact MCS approach meaning that any strategy applied must not influence the result size. In the evaluation of heuristic approaches by Englert, the applied strategies influenced the runtime and the result. The comparison of the results was achieved using an "average result size ratio", which is the percentage of the MCES result size compared to an ab initio upper bound estimation of the result size.

The average runtimes for the constrained dMCS with at most three components of minimum size 3 are shown in Table 3. The results show that the gain in performance is significant on all compound sets. Beyond that, the overall runtimes show that the focused sets of similar compounds are especially hard for the exact method. On the other hand, this perfectly demonstrates the effect of efficient state space traversal. Small and simple problem instances are solved efficiently even by a naive algorithm.

Table 3. Mean Runtimes of the Constrained dMCS Calculation for Comparing Pairwise on Selected Compound Sets[a]

	mean runtimes in ms			
	3/3 dMCIS		3/3 dMCES	
set	LVS, MCF	all strategies	LVS, MCF	all strategies
NCI1	17,954	6314	1931	829
NCI2	48,392	34,206	113,326	77,744
NCI-Key	2011	1509	986	875
EvalSet	1495	1109	261	257

[a]The result is constrained to at most three components of minimum size 3 applying the LVS and MCF strategies compared to applying the LVF, MCF, RNP and SPF strategies.

Relative Position Evaluation. The key strategy in the presented algorithm is to favor nodes with the least number of possible mapping partners (seed LVF strategy). Furthermore, the algorithm relies on strategies selecting compatible atoms of the target graph, as evaluated in the previous section. Another issue is the selection of the atom to start with. Even after applying the LVF strategy, there might be several atoms of equal preference. We propose using the relative position of atoms as a secondary decision criterion for the LVF strategy.

To validate the effect of this approach, we evaluated the following relative positions 50%, 75%, and 100% as a preferred mapping start and compared the results to the case in that no selection strategies at all are applied (No S) and relative positions are not considered (No R) in the LVF strategy.

Table 4 shows the evaluation of different relative query molecule starting positions. In principle, the relative starting position has only a minor influence on the runtimes. For the MCIS, the relative position of 75% performs best on three of the five datasets. For the MCES, the relative positions of 100% also performs best in three of five cases.

Neglecting the applied strategies, the actual graph representation of the input can have a major influence on the runtimes as well. To measure the impact of representation dependency and that our strategies help reduce that influence, we randomly permuted the atom and bond positions within the input molecule and performed the same experiment again. Table 5 shows the results. It is important to compare the runtimes to those of Table 4. For the cMCIS on the first three datasets, there is almost no difference. However, for the NCI2 and NCI3 sets, if no selection strategies are applied at all, the performance significantly improves. For those cases where all of the strategies are applied, shuffling the atom order has only a minor influence on the algorithm performance.

For the cMCES, the situation is more complex. Shuffling the input with no strategies applied results twice in the best and twice in the worst performance on a dataset. For the remaining configurations and sets, the mean runtimes are configuration-independent. For the NCI1 and NCI3 sets the runtimes improve significantly, whereas the runtimes do not change much for the first two compound sets.

In order to keep the algorithm consistent for the MCIS and MCES, we will not rely on input permutation for any of the cases and continue with the relative positions of 100 for the MCIS and 75 for the MCES.

Influence of the Compound Sets. In this paper, two kinds of compound sets are used. There are the more diverse ones like the EvalSet or the NCI-Key selection and the more focused NCI1 and NCI2 sets by Englert. The latter show significantly increased runtimes compared to the diverse sets, although the average number of atoms per molecule does not differ that much. However, within the similarity statistics of Table 1, there are still some differences observable. First, the average cMCS and dMCS similarities is more than 50% higher. Another interesting point is the difference between dMCES and dMCIS average similarities. This difference hints that the dMCS contains several small connected components that are adjacent to each other. Compared to the target sets, this seems to be the major difference. The NCI-Key selection has a relatively large difference in dMCS similarity as well but with a much smaller baseline of cMCS similarity. Low cMCS similarity values are the result of only small common components, which dramatically reduce state space for the cMCS.

Influence of Constraining the Number of Connected Components and Their Size. Defining an upper bound for the number of connected components and defining a lower

Table 4. cMCS Performance Evaluation on a Selection of the Datasets[a]

	MCIS mean times (ms)					MCES mean times (ms)				
set	No S	No R	50%	75%	100%	No S	No R	50%	75%	100%
EvalSet	4.90	4.86	4.91	4.94	4.72	3.10	3.34	3.28	3.29	3.21
NCI-Key	3.04	3.10	3.08	3.11	3.06	2.61	2.61	2.65	2.60	2.65
NCI1	129	118	124	130	115	29.30	28.68	27.79	31.17	32.55
NCI2	191	94	108	103	94	819	799	738	633	717
NCI3	12,827	5146	4749	5219	5549	16,356	12,547	12,234	11,796	12,903

[a]Times are provided for the cMCIS and cMCES. For "No S", no strategies are applied at all, and "No R" does not use relative positions to enhance the LVS strategy. The values between 50% and 100% belong to the preferred relative position of the starting node for the LVS strategy on conflict.

https://dx.doi.org/10.1021/acs.jcim.0c00741
J. Chem. Inf. Model. 2021, 61, 167−178

Table 5. Similar Experiment as in Table 4, but This Time, the Internal Molecule Representations Are Permuted, Meaning that the Internal Order of Atoms and Bonds Was Shuffled

	MCIS mean times (ms)					MCES mean times (ms)				
set	No S	No R	50%	75%	100%	No S	No R	50%	75%	100%
EvalSet	5.06	4.93	4.99	5.05	4.92	2.99	3.28	3.15	3.24	3.27
NCI-Key	3.43	3.26	3.30	3.27	3.22	2.78	2.75	2.94	2.86	2.83
NCI1	133	115	115	128	108	18.2	21.6	21.1	18.6	24.3
NCI2	122	102	105	97	95	1094	856	771	820	950
NCI3	6199	4630	4841	4677	4898	15,306	11,280	10,182	9760	11,209

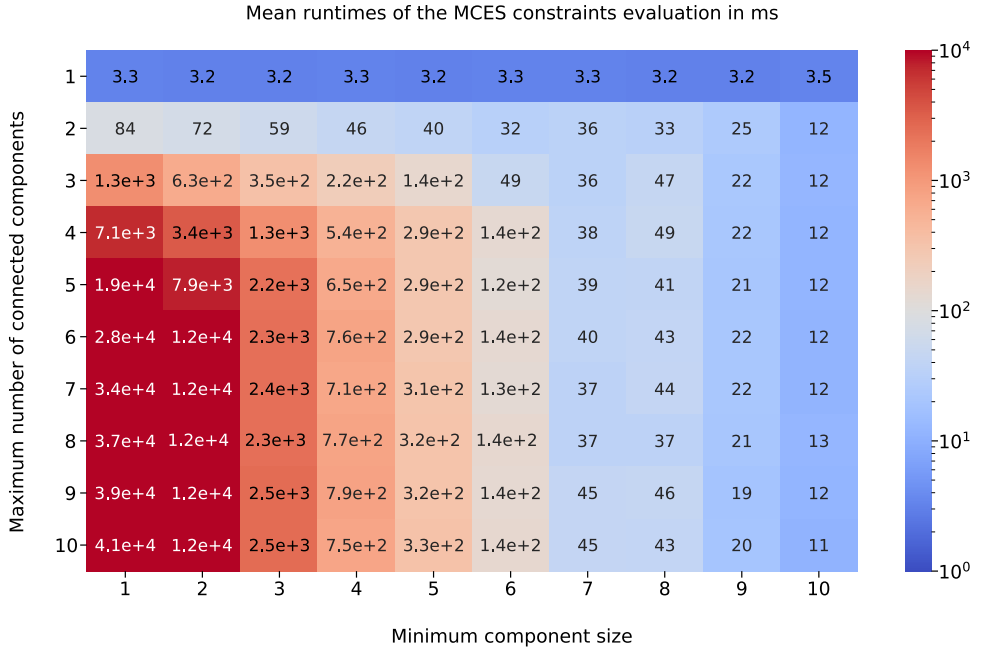


Figure 4. Comparison of the mean runtimes for all parameter combinations on the EvalSet.

bound on the number of atoms in each component have a significant influence on runtimes.

We evaluated the performance for up to 10 connected components with a minimum atom count of 1 to 10 per connected component. The results for the dMCES can be seen in the heatmaps shown in Figures 4 and 5.

Figure 4 shows that, if a connected subgraph is requested (row 1), the minimum size obviously has no impact. This changes as soon as disconnected subgraphs are allowed. For up to two connected components, the mean runtimes are well below 100 ms in all cases analyzed. The same is true if the minimum component size is set to at least 6. Computing times significantly increase if many connected components of small size are allowed. Fortunately, this is the scenario, which is not very meaningful in cheminformatics applications. For reasonable settings with up to three, probably five connected components with at least three to five atoms, the MCS can still be computed in a second on average.

To estimate the importance of dMCES, we evaluated the average number of additionally mapped atoms in comparison to the cMCES. (see Figure 5). As expected, a large minimum component size compensates the opportunity to map multiple connected components (right columns). On average, three to six more atoms can be mapped if a small number of connected components with a reasonable size of at least three atoms is allowed. Although we think that this is a general trend, these results might vary with different datasets used.

In Figures 4 and 5, the most demanding settings are small minimum component sizes with a large number of connected components. Not surprisingly, the algorithm reveals asymptotically exponential runtime behavior. We can see that mean runtimes are heavily influenced by a small portion of outliers (see Figure S1 in the Supporting Information).

Similar to clique-based approaches, the incremental algorithm presented here is designed for MCIS calculations and computes the MCES using line graphs. To accomplish for that, we also measured the algorithm performance for the MCIS and plotted the results in three heatmaps in Figure S2 in the Supporting Information. The overall runtime and result size behavior is the same for the MCES and MCIS, whereas the MCES has on average faster runtimes on the dataset used.

Constrained MCIS on Similar Compounds. To show the effect of constraining the number of connected components and their size on larger, more similar compounds, we iterate the analysis on the NCI2 dataset with more than 40 heavy atoms on average. On larger compounds, it is possible to find

https://dx.doi.org/10.1021/acs.jcim.0c00741
J. Chem. Inf. Model. 2021, 61, 167−178

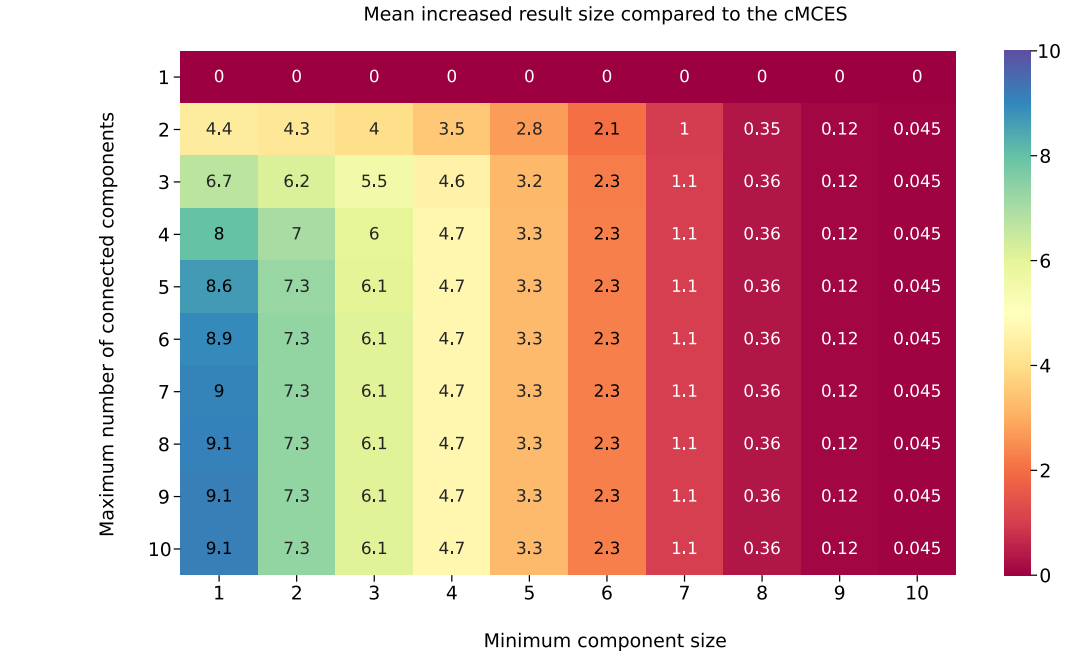


Figure 5. Constrained dMCES result sizes in comparison to the cMCES calculation.

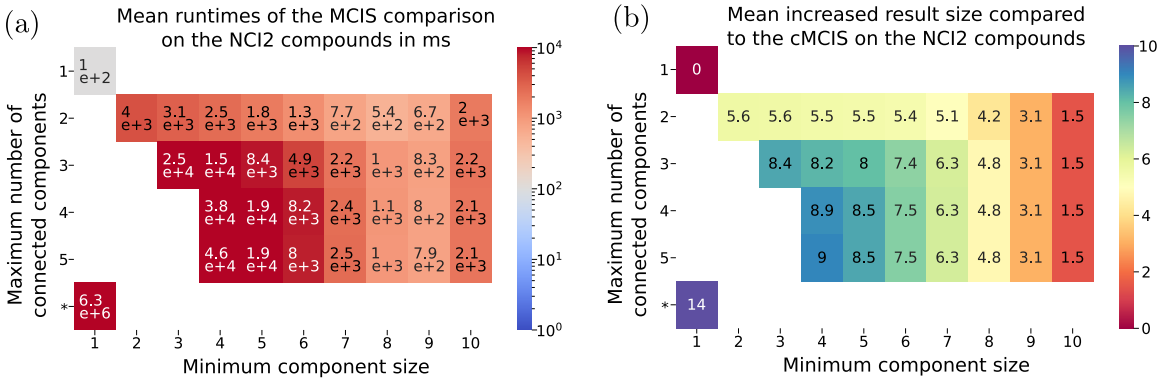


Figure 6. Selected constraint evaluation on the NCI2 dataset.

multiple connected components with larger minimum sizes. Figure 6 shows the runtimes and the gain in result size compared to the cMCIS for up to at least six atoms in each connected component.

.

The first observation in this experiment is that the cMCIS has a higher average runtime of approximately 100 ms. For two connected components, the mean computation time is in the range of a few seconds and falls below 1 s for a minimum component size of 7. Interestingly, the runtimes increase for a minimum size of 10 for all numbers of connected components evaluated. It seems that the upper bound estimation works much worse compared to a component size of 9. Regarding the runtimes for three or more connected components, we observe that all evaluated configurations except a component size of 9 need more than 1 s per MCIS on average. In our opinion, the best configuration in this experiment is 3/5 with an average increase of eight additional mapped atoms compared to the cMCIS and a mean computation time that is still below 10 s. For comparison, the unconstrained dMCIS needs more than an hour on average and achieves an increase of 14 atoms.

Constraint Examples. In order to provide some context how the constraints work we selected three pairs of compounds. For each pair, an MCS between the compounds is shown within a column. Over the different columns, different configurations are displayed.

Figure 7 shows three differently constrained MCS of two compounds of the EvalSet. Independent of the required component size, this pair allows up to three connected components of the mapping of size six. There is no difference between MCIS and MCES. In Figure 8, a slightly different example is shown. Again, the compound pair allows up to three connected components. The cMCIS and cMCES are still identical, but the constrained dMCIS and dMCES differ. This

https://dx.doi.org/10.1021/acs.jcim.0c00741
J. Chem. Inf. Model. 2021, 61, 167−178

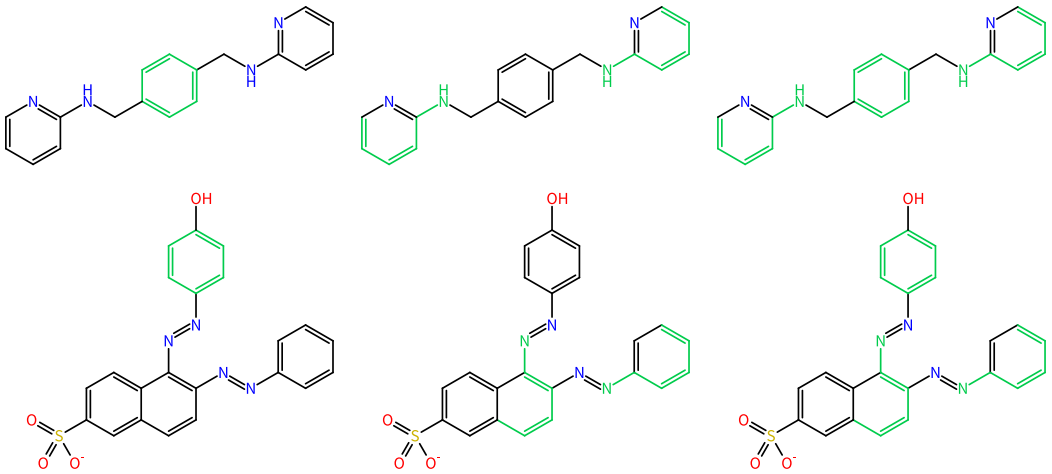


Figure 7. Three variants of an MCS between the compounds "C67488378" and "116" of the EvalSet; the columns, the cMCS, on the left and the constrained dMCS with two respective three components in the middle and on the right. There is no difference in the results for MCIS and MCES.

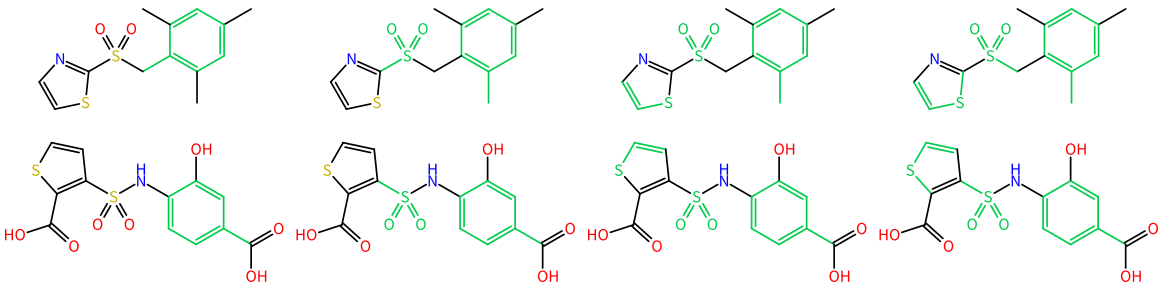


Figure 8. Another example with three variants of an MCS between the compounds "403120 CHEMBL237830" and "C03844583" of the EvalSet. In the first column, a cMCS is shown. The second column shows a constrained dMCS with at most 2 components. The last two columns show the constrained dMCIS and dMCES in the 3/3 configuration.

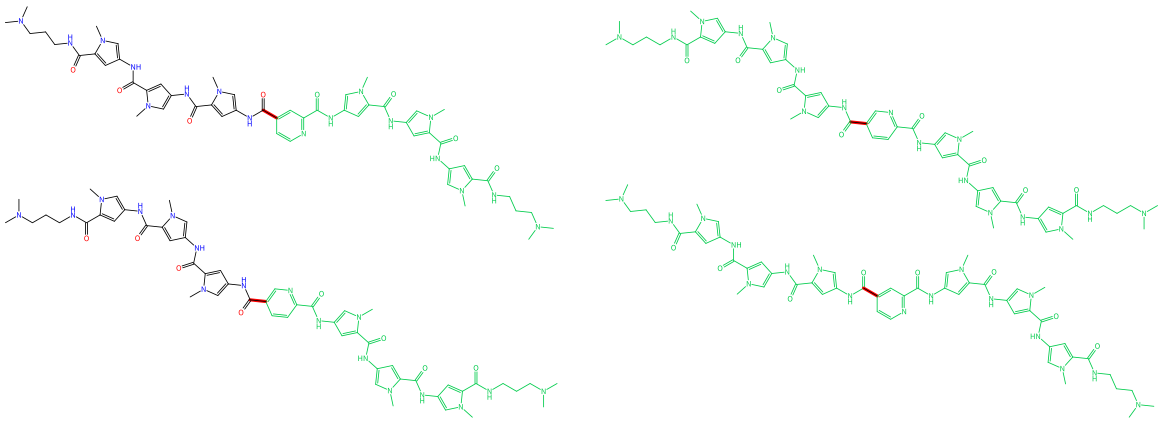


Figure 9. Example with two compounds of the NCI3 set. On the left, the cMCS is shown, and on the right, the dMCES. Both compounds are identical except for the ring substitution in the middle. The outgoing bond of this ring is additionally highlighted in red. Here, the dMCES is computed more than an order of magnitude faster than the cMCS.

is shown in the last two columns. Finally, in Figure 9, an interesting performance example is shown. Both compounds are taken from the NCI3 set and differ only in a central substitution position. This causes the interesting situation in that the dMCS is computed faster than the cMCS. For reasonable constraint selections like 3/3, the dMCS is more than an order of magnitude faster than the cMCS. The bond

https://dx.doi.org/10.1021/acs.jcim.0c00741
J. Chem. Inf. Model. 2021, 61, 167−178

Table 6. Comparison of the cMCES Runtimes to Other Exact Methods and Times of Heuristic Approaches Reported in the Literature on Similar Hardware.

	reported times in ms[5]			measured times in ms			
set	MC	BU	Indigo(H)	RIMACS	Indigo(E)	RIMACS(R)	RDKit
EvalSet	-	-	-	2.3	7.8	10.0	17.9
NCI-KEY	-	-	-	2.0	8.1	15.6	38.7
NCI1	16.6	1.3	10.8	69.5	8934	255.9	878.7
NCI2	18.6	2.7	30.7	605.4	22,858	830.5	47,364
NCI3	38.3	12.2	99.6	-	-	-	-
NCI4	125.5	107.0	190.6	-	-	-	-
NCI-S	9.2	1.5	12.9	5.7	127.4	50.2	267.5
CAH2	4.4	0.4	7.3	0.7	2.0	1.4	4.0
CDK2	10.6	0.8	10.6	1.7	5.0	6.1	15.0
NEU	6.5	1.1	9.0	0.9	3.4	2.5	3.4

[a]Englert and Kovács[5] benchmarked heuristic approaches. MC is their proposed best heuristic MCS algorithm. BU is an authoritative reimplementation of the build-up algorithm by Kawabata,[9] and Indigo(H) is the heuristic MCS approach of the Indigo[27] toolkit. We measured times for the exact variant of Indigo named Indigo(E). The RDKit[28] uses a weaker bond compatibility criterion than Indigo. For a more fair comparison, we report two different times for our MCES. The column RIMACS shows the results and mean runtimes for the default cMCES in RIMACS. The result sizes are comparable with those of the Indigo library. To offer a fair comparison with the RDKit, the RIMACS(R) variant is part of the comparison. For this cMCES variant, all bonds are assumed compatible. Using this relaxation result, sizes become more comparable. Times marked with "-" were not calculated.

belonging to this difference is additionally highlighted for better visualization.

COMPARISON WITH OTHER METHODS

In a last study, we compare our algorithm to two other cMCS approaches, computing a cMCES. We used RDKit[28] and Indigo[27] as a reference. Both are open source libraries with a different approach of computing the MCES. In RDKit, the FMCS[7] algorithm is used. It is based on substructure enumeration and generates a SMARTS string as part of the MCS result. The FMCS is applicable to the pairwise MCS, but it is better suited for the multi MCS problem, finding a common substructure of more than two compounds. In the Indigo library, an exact clique-based algorithm[29] computes the MCS. Additionally, we compare the performance to a study on heuristic approaches by Englert and Kovács.[5] The study is based on their selection of similar compound sets and three other compound sets from the literature. The authors compare their implementation of a clique-based MCES algorithm (MC) and a so-called build-up algorithm (BU) to the heuristic MCS algorithm implemented in Indigo as well. We included the results of the study in Table 6 and extended it with mean runtimes we measured with our algorithm and the RDKit and Indigo MCS implementations.

Overall, we observe that an exact cMCES on the diverse and relatively simple sets is computed efficiently and has similar mean runtimes as the heuristic approaches. This trend of fast computation is supported by the comparison on our diverse sets based on the EvalSet or our NCI-Key selection. In the comparison of the exact approaches, the substructure-based algorithm presented here performs best. Comparing the runtimes on the focused sets of similar compounds with increasing size, the heuristic approaches have shorter runtimes as expected. The first two NCI sets with around 20 or 40 heavy atoms per molecule are at the boundary of acceptable mean runtimes for exact algorithms. In comparison with the other two exact approaches RIMACS is reliably the fastest competitor. Especially on the NCI compounds by Englert, RIMACS outperforms the other exact approaches by a factor of at least 3 and often by an order of magnitude. When analyzed in more detail (see Table S1 in the Supporting Information), this runtime advantage is based on overall improvements but especially on more robust runtimes on outliers. For example, in the third quantile of the comparisons for the NCI2 compounds, Indigo is approximately 15% slower, but on the overall set, average runtimes differ by a factor of ~40. On the more diverse compound sets, the average runtimes are less influenced by outliers. This is indicated by the RIMACS average runtimes that are between the second and third quartile for four of five compound sets.

CONCLUSIONS AND OUTLOOK

We presented our RIMACS algorithm for the computation of exact maximum common substructures with additional constraints. The algorithm follows an established strategy of incremental growing substructures. It handles an upper bound for the number of connected components in combination with a lower bound on the number of atoms in each component. These constraints for dMCS are chemically meaningful and increase the algorithm efficiency substantially without becoming heuristic. We proved correctness of the applied backtracking scheme and the implemented strategies for search space optimization. The algorithm is designed for the induced maximum common substructure search but is also applicable for the chemically more relevant non-induced MCES problem using line graph conversion.

Our evaluation shows the benefit of our applied atom selection strategies for the connected and constrained disconnected MCS. We evaluate runtimes and result sizes compared to the connected MCS. The evaluation shows that the minimum component size has much more influence on runtimes than the actual maximum number of connected components. Constraining the disconnected MCS to at most three connected components with at least three atoms results in chemically reasonable maximum common substructures and can still be computed with high efficiency.

In comparison to heuristic approaches, the focused compound sets show the limits of the exact MCS computation. Substructure-based approaches are advantageous for the exact computation, but the asymptotically exponential runtime

https://dx.doi.org/10.1021/acs.jcim.0c00741
J. Chem. Inf. Model. 2021, 61, 167−178

behavior leads to infeasibility on large, highly similar and pseudo-symmetric compounds.

Heuristic algorithms might be a way to go in these scenarios. This comes, however, with several disadvantages besides the fact that the results are most likely suboptimal. If the algorithm is randomized, it is not even guaranteed that the results of two independent calculations of the same molecules have the same size.

Finally, the mean runtimes within our evaluation are dominated by a small number of outliers. Even for higher quantiles like 75% or 90%, the runtimes per computation are still affordable. As an alternative to a heuristic algorithm, we propose to use a reasonable limit on the number of recursive backtracking steps. If not finished within this bound, the algorithm might go into a heuristic, avoiding infeasible long runtimes.

The RIMACS algorithm became part of several of our tools within the NAOMI ChemBio Suite including SMARTScompare.[30,31] Furthermore, it forms an ideal ground for the MCS calculation and substructure searching in chemical fragment spaces.[32,33]

■ ASSOCIATED CONTENT

Supporting Information

The Supporting Information is available free of charge at https://pubs.acs.org/doi/10.1021/acs.jcim.0c00741.

The proof of correctness, implementation notes of the ΔY exchange, additional runtime heatmaps, and a more detailed runtime comparison table for the external tool comparison (PDF)

■ AUTHOR INFORMATION

Corresponding Author

Matthias Rarey − *Universität Hamburäg, ZBH - Center for Bioinformatics, 20146 Hamburg, Germany;* orcid.org/0000-0002-9553-6531; Phone: +49 (40) 428387351; Email: rarey@zbh.uni-hamburg.de

Authors

Robert Schmidt − *Universität Hamburäg, ZBH - Center for Bioinformatics, 20146 Hamburg, Germany;* orcid.org/0000-0002-7089-4842

Florian Krull − *Universität Hamburäg, ZBH - Center for Bioinformatics, 20146 Hamburg, Germany*

Anna Lina Heinzke − *Universität Hamburäg, ZBH - Center for Bioinformatics, 20146 Hamburg, Germany*

Complete contact information is available at: https://pubs.acs.org/10.1021/acs.jcim.0c00741

Author Contributions

All authors contributed to the algorithmic development of RIMACS. F.K. provided an initial version of the backtracking algorithm 2005/2006. A.L.H performed an initial clique-based evaluation of the disconnection constraints. R.S. implemented the strategies and upper bounds and performed the experiments. R.S. and M.R. wrote the manuscript.

Notes

The authors declare the following competing financial interest(s): M.R. declares a potential financial interest in the event that software tools including the RIMACS algorithm are licensed for fee to non-academic institutions in the future.

The NAOMI ChemBio Suite including tools like SMARTScompare using the RIMACS algorithm are available through https://uhh.de/naomi. Many of our tools can be used on our web servers https://proteins.plus and https://smarts.plus. The whole suite of tools is available free of charge for academic use, otherwise for an annual fee. The source code of the RIMACS algorithm is released on our GitHub repository https://github.com/rareylab.

■ REFERENCES

(1) Raymond, J. W.; Willett, P. Maximum common subgraph isomorphism algorithms for the matching of chemical structures. *J. Comput.-Aided Mol. Des.* **2002**, *16*, 521−533.

(2) Ehrlich, H. C.; Rarey, M. Maximum common subgraph isomorphism algorithms and their applications in molecular science: A review. *Wiley Interdiscip. Rev.: Comput. Mol. Sci.* **2011**, *1*, 68−79.

(3) Duesbury, E.; Holliday, J. D.; Willett, P. Maximum Common Subgraph Isomorphism Algorithms: A Review. *MATCH Commun. Math. Comput. Chem.* **2016**, *77*, 213−232.

(4) Cao, Y.; Jiang, T.; Girke, T. A maximum common substructure-based algorithm for searching and predicting drug-like compounds. *Bioinformatics* **2008**, *24*, 366−374.

(5) Englert, P.; Kovács, P. Efficient heuristics for maximum common substructure search. *J. Chem. Inf. Model.* **2015**, *55*, 941−955.

(6) Hariharan, R.; Janakiraman, A.; Nilakantan, R.; Singh, B.; Varghese, S.; Landrum, G.; Schuffenhauer, A. MultiMCS : A Fast Algorithm for the Maximum Common Substructure Problem on Multiple Molecules. *J. Chem. Inf. Model.* **2011**, *51*, 788−806.

(7) Dalke, A.; Hastings, J. FMCS: a novel algorithm for the multiple MCS problem. *Aust. J. Chem.* **2013**, *5*, O6.

(8) Koch, I. Enumerating all connected maximal common subgraphs in two graphs. *Theor. Comput. Sci.* **2001**, *250*, 1−30.

(9) Kawabata, T. Build-up algorithm for atomic correspondence between chemical structures. *J. Chem. Inf. Model.* **2011**, *51*, 1775−1787.

(10) Kawabata, T.; Nakamura, H. 3D flexible alignment using 2D maximum common substructure: Dependence of prediction accuracy on target-reference chemical similarity. *J. Chem. Inf. Model.* **2014**, *54*, 1850−1863.

(11) Duesbury, E.; Holliday, J.; Willett, P. Comparison of Maximum Common Subgraph Isomorphism Algorithms for the Alignment of 2D Chemical Structures. *ChemMedChem* **2018**, *13*, 588−598.

(12) Stahl, M.; Mauser, H.; Tsui, M.; Taylor, N. R. A robust clustering method for chemical structures. *J. Med. Chem.* **2005**, *48*, 4358−4366.

(13) Cazals, F.; Karande, C. An algorithm for reporting maximal c-cliques. *Theor. Comput. Sci.* **2005**, *349*, 484−490.

(14) Marialke, J.; Körner, R.; Tietze, S.; Apostolakis, J. Graph-based Molecular Alignment (GMA). *J. Chem. Inf. Model.* **2007**, *47*, 591−601.

(15) Mccreesh, C.; Samba Ndojh, N.; Patrick, P.; Christine, S. In *Princ. Pract. Constraint Program. 22nd Int. Conf. CP 2016, Toulouse, Fr. Sept. 5−9, 2016, Proc.*; Rueher, M., Ed.; Springer International Publishing: Cham, 2016; Vol. *9892*; pp. 350−368

(16) Grosso, A.; Locatelli, M.; Pullan, W. Simple ingredients leading to very efficient heuristics for the maximum clique problem. *J. Heuristics* **2008**, *14*, 587−612.

(17) Bron, C.; Kerbosch, J. Algorithm 457: finding all cliques of an undirected graph. *Commun. ACM* **1973**, *16*, 575−577.

(18) Raymond, J. W.; Gardiner, E. J.; Willett, P. RASCAL: Calculation of graph similarity using maximum common edge subgraphs. *Comput. J.* **2002**, *45*, 631−644.

(19) McGregor, J. J. Backtrack search algorithms and the maximal common subgraph problem. *Softw. Pract. Exp.* **1982**, *12*, 23−34.

(20) Hagadone, T. R. Molecular Substructure Similarity Searching: Efficient Retrieval in Two-Dimensional Structure Databases. *J. Chem. Inf. Comput. Sci.* **1992**, *32*, 515−521.

https://dx.doi.org/10.1021/acs.jcim.0c00741
J. Chem. Inf. Model. 2021, 61, 167−178

(21) Whitney, H. Congruent Graphs and the Connectivity of Graphs. *Am. J. Math.* **1932**, *54*, 150.
(22) Nicholson, V.; Tsai, C.-C.; Johnson, M.; Naim, M. A Subgraph Isomorphism Theorem for Molecular Graphs. In *Graph Theory and Topology in Chemistry*; Elsevier: Amsterdam, 1987; pp. 226−230
(23) Cordella, L. P.; Foggia, P.; Sansone, C.; Vento, M. A (sub)graph isomorphism algorithm for matching large graphs. *IEEE Trans. Pattern Anal. Mach. Intell.* **2004**, *26*, 1367−1372.
(24) Urbaczek, S.; Kolodzik, A.; Fischer, J. R.; Lippert, T.; Heuser, S.; Groth, I.; Schulz-Gasch, T.; Rarey, M. NAOMI: On the almost trivial task of reading molecules from different file formats. *J. Chem. Inf. Model.* **2011**, *51*, 3199−3207.
(25) Mysinger, M. M.; Carchia, M.; Irwin, J. J.; Shoichet, B. K. Directory of useful decoys, enhanced (DUD-E): Better ligands and decoys for better benchmarking. *J. Med. Chem.* **2012**, *55*, 6582−6594.
(26) *National Cancer Institute, NCI-Realease 2012*. 2012; https://cactus.nci.nih.gov/download/nci/NCI-Open_2012-05-01.sdf.gz, Accessed on: 2020-02-18
(27) Pavlov, D.; Rybalkin, M.; Karulin, B.; Kozhevnikov, M.; Savelyev, A.; Churinov, A. Indigo: Universal cheminformatics API. *J. Cheminf.* **2011**, *3*, 2011.
(28) Landrum, G. *RDKit: Open Source Cheminformatics*. https://www.rdkit.org/, Accessed on: 2020-02-18
(29) Tonnelier, C.; Jauffret, P.; Hanser, T.; Kaufmann, G. Machine learning of generic reactions: 3. an efficient algorithm for maximal common substructure determination. *Tetrahedron Comput. Methodol.* **1990**, *3*, 351−358.
(30) Schmidt, R.; Ehmki, E. S. R.; Ohm, F.; Ehrlich, H.-C.; Mashychev, A.; Rarey, M. Comparing Molecular Patterns Using the Example of SMARTS: Theory and Algorithms. *J. Chem. Inf. Model.* **2019**, *59*, 2560−2571.
(31) Ehmki, E. S. R.; Schmidt, R.; Ohm, F.; Rarey, M. Comparing Molecular Patterns Using the Example of SMARTS: Applications and Filter Collection Analysis. *J. Chem. Inf. Model.* **2019**, *59*, 2572−2586.
(32) Rarey, M.; Stahl, M. Similarity searching in large combinatorial chemistry spaces. *J. Comput.-Aided Mol. Des.* **2001**, *15*, 497−520.
(33) Boehm, M.; Wu, T.-Y.; Claussen, H.; Lemmen, C. Similarity searching and scaffold hopping in synthetically accessible combinatorial chemistry spaces. *J. Med. Chem.* **2008**, *51*, 2468−2480.

https://dx.doi.org/10.1021/acs.jcim.0c00741
J. Chem. Inf. Model. 2021, 61, 167−178

Supporting Information: Disconnected Maximum Common Substructures Under Constraints

Robert Schmidt,[†] Florian Krull,[†,‡] Anna Lina Heinzke,[†] and Matthias Rarey[*,†]

†*Universität Hamburg, ZBH - Center for Bioinformatics, Bundesstraße 43, 20146 Hamburg, Germany*

‡*Present Address: University of Oslo, Department of Psychology, Center for Lifespan Changes in Brain and Cognition, POB 1094 Blindern, 0317 Oslo, Norway*

E-mail: rarey@zbh.uni-hamburg.de

Phone: +49 (40) 428387351

Proof of Correctness

Let Q, T be molecules (graphs) with their sets of atoms $V(Q), V(T)$ and bonds $E(Q), E(T)$ respectively and let $M \subseteq V(Q) \times V(T)$ be an induced mapping. The mapping is built from pairs of nodes whereby each node of the graphs Q and T occurs at most once. Furthermore M_{max} is the mapping of maximum size.

$$\forall (v_q, v_t), (u_q, u_t) \in M : \{v_q, u_q\} \in E(Q) \Leftrightarrow \{v_t, u_t\} \in E(T)$$

To address all mapped nodes of a graph we define:

$$V(M,G) = \{u|(u,v) \in M \wedge u \in V(G)\} \cup \{v|(u,v) \in M \wedge v \in V(G)\}$$

In order to access mapped nodes efficiently we define the mapping partner function:

$$P(M,u) = v, (u,v) \in M$$

Finally to access adjacent nodes, the neighbor function is defined as follows:

$$N(u,G) = \{v|\{u,v\} \in E(G)\}$$

Theorem 1. *Given a connected mapping M_c, a node $u \in V(M_c, Q)$ and a sufficiently small set of forbidden nodes $F_0 \subseteq V(Q) \setminus V(M_c, Q)$. If* MCS-Incremental *is initially called with a single node candidate list $C_0 = \{u\}$ and not bound by any upper bound estimation, then there exists a number k such that* MCS-Incremental *eventually calls itself with $M_k = M_c$ and $C_k = \emptyset$.*

Proof. Initially the MCS-Incremental function is called with $M_0 = \emptyset, C_0 = \{u\}, F_0, M_{CurrMax}$, whereby $F_0 \cap V(M_c, Q) = \emptyset$. In this context, $M_{CurrMax}$ will be ignored. Here we use the assumption that the UpperBound function will always return an estimation $e > |M_{CurrMax}|$. Consider recursion depth i with $M_i \subseteq M_c, F_i \cap V(M_c, Q) = \emptyset$. No node of the connected target mapping M_c is forbidden and the current mapping M_i is a subset of it. This is especially true for the initial call to MCS-Incremental as mentioned above.

Now the next node v for mapping extension is selected in line 8. In the following, the mapping is extended with any possible compatible node p. By definition of the SelectCompatibleNodes function, the mapping $M_{i+1} = M_i \cup \{(v,p)\}$ is a valid mapping. Furthermore $j = P(M_c, v)$ is part of the set of mapping partners for extension. Thus, eventually the forbidden nodes F_{i+1} will be calculated using M_i, F_i and (v,j). By definition of

the UpdateForbiddenNodes function, we know that $F_{i+1} \cap V(M_c, Q) = \emptyset$, since M_c is a valid mapping and $M_i \cup \{(v, j)\} \subseteq M_c$.

This also implies that any neighbor $n \in N(v, Q)$ that is not yet mapped but part of M_c is guaranteed to be in the candidate list C_{i+1}. Furthermore, if the current candidate list C_i contained any node w that is mapped in M_c, $w \in C_{i+1}$. This implies that the candidate set is not empty if M_c is not enumerated yet. $M_i \neq M_c \Rightarrow C_{i+1} \cap V(M_c, Q) \neq \emptyset$. Now MCS-Incremental is recursively called with $M_{i+1}, C_{i+1}, F_{i+1}, M_{CurrMax}$.

$M_{i+1} \subseteq M_c, F_{i+1} \cap V(M_c, Q) = \emptyset$ and $|M_{i+1}| < |M_c| \Rightarrow C_{i+1} \cap V(M_c, Q) \neq \emptyset$.

It remains to show that if $M_j = M_c$ then MCS-Incremental is eventually called with $C_j = \emptyset$. In line 8 any node of $x \in C_j$ is selected and removed from C_j. In line 17 MCS-Incremental is called with unchanged $M_{j+1} = M_j = M_c$ and $M_{CurrMax}$ but with an updated $C_{j+1} = C_j \setminus \{x\}$ and $F_{j+1} = F_j \cup \{x\}$. This proves the statement.

Lemma 1. *Given a connected mapping M_i with respective candidate list C_i and forbidden node set F_i occurring during enumeration of the mapping M. F_i is extended only with candidate nodes and those violating inducedness of an extended mapping.*

Proof. Within the MCS-Incremental function, there are two lines where F_{i+1} is calculated.

Line 17 The extension where the selected node v is not part of $M_{i+1} = M_i$. This implies that there are no additional nodes that violate inducedness of an extended mapping. And $F_{i+1} = F_i \cup \{v\}$ and $C_{i+1} = C_i \setminus \{v\}$.

Line 12 F_{i+1} is calculated using $M_i, F_i, (v, p)$, whereas v, p as defined in the pseudocode. Following the definition of UpdateForbiddenNodes F_{i+1} contains v. Furthermore all nodes for which mapping extensions with all remaining mapping partners would lead to invalid mappings M_{i+1}.

Lemma 2. *Given a mapping M_i, F_i and the pair for extension (v, p) as used in line 12 of the MCS-Incremental function with $M_{i+1} = M_i \cup \{(v, p)\}$. For all compatible node pairs $(q, t), q \in$*

$V(Q) \setminus F_i, t \in V(T) \setminus V(M_{i+1}, T)$ *for which the extended mapping* $M_{i+2} = M_{i+1} \cup \{(q,t)\}$ *is invalid, either* q *or* t *is adjacent to any node of* M_{i+1}.

Proof. Assume q is not adjacent to any mapped node $\forall u \in V(M_i, Q) : \{q,u\} \notin E(Q)$ and t is not adjacent to any mapped node as well, then $M_{i+2} = M_{i+1} \cup \{(q,t)\}$ is a mapping. This follows from the definition of inducedness. $v_1, v_2 \in V(M,Q) : \{v_1, v_2\} \in E(Q) \Leftrightarrow \{P(v_1, M), P(v_2, M)\} \in E(T)$. This proves the statement.

Lemma 3. *Given a mapping* M_i*, a nonempty candidate list* $C_i \neq \emptyset$ *disjoint to* F_i *and an* F_i *that is a superset of all nodes whose mapping extension would lead to an invalid mapping. Then every mapping* M_{i+1} *computed by* MCS-Incremental *is valid.*

Proof. Initially MCS-Incremental is called by MCS-ProceedWithNextComponent with an empty mapping $M_0 = \emptyset$, a single node candidate list $C = \{v\}$ for any node $v \in V(Q)$, an empty set F of forbidden nodes and an empty current maximum mapping $M_{CurrMax}$.

The initial call fulfills the stated constraints. In line 12 F_{i+1} is calculated. Then C_{i+1} is constructed considering F_{i+1}, correctness following from Lemma 1. Thus, in any subsequent backtracking step where $M_{i+1} \neq M_i$ is a parameter, M_{i+1}, C_{i+1} and F_{i+1} still fulfill the requirement.

Finally in line 17, the same holds true if $M_{i+1} = M_i$ omitting the current node v for the mapping. Thus all computed mappings $M_{CurrMax}$ are valid.

Lemma 4. *The upper bound estimation is correct for the cMCS.*

Proof. The most simple approximation for an upper bound for any state during the algorithm is the size of the current mapping $|M_i|$ plus the remaining mappable nodes $|V(Q) \setminus F_i|$. This follows from Lemma 1. Since the algorithm for the cMCIS is limited to connected expansion of the mapping, each node has to be reachable following a path of mappable nodes starting at any mapped node. Assume there is a node d for which no such path exists and a connected mapping M_j such that $d \in V(M_j, Q) \wedge M_i \subset M_j$. Connectedness of M_j implies that each

mapped node of $M_i \subset M_j$ is reachable using a path consisting of mapped nodes of M_j. This contradicts the assumption.

Theorem 2. *The cMCIS of two graphs Q and T is computed correctly by the* MCS *function.*

Proof. Given the maximum connected mapping M_{max} that represents the cMCIS of Q and T, it is necessary to show that it is eventually enumerated. The MCS function just calls MCS-ProceedWithNextComponent once and the MaxComp parameter prevents any other calls of this function later on. It calls MCS-Incremental with a single node candidate list $C_0 = \{v\},\ \forall v \in V(Q)$. Afterwards v is added to F. Let v_0 be the first node during enumeration that is part of M_{max}. Theorem 1 shows that M_{max} can be enumerated. Since the UpperBound estimation is correct, any time UpperBound is called given $M_i \subseteq M_{max}$, the estimation is at least $|M_{max}|$. Thus either M_{max} is enumerated, or an alternative connected mapping $M'_{max}, |M'_{max}| \geq |M_{max}|$ is found. Lemma 3 implies that M'_{max} is valid. If $|M'_{max}| > |M_{max}|$, then M_{max} is not the cMCIS which contradicts the assumption. Otherwise an alternative valid cMCIS M'_{max} is correctly enumerated.

Lemma 5. *Given a mapping M and an independent connected mapping M_c that have no nodes in common, and none of the nodes of M are adjacent to any node of M_c. Furthermore let the* UpperBound *estimation always return a sufficiently high estimation e, such that no backtrack branch is bound and let the $MaxComp$ parameter be sufficiently high and let $MinSize$ be sufficiently low. Let the* MCS-Incremental *initially be called with $M_0 = \emptyset, C_0 = \{q\}$ for any node in $V(Q)$, F_0 disjoint to $V(M_u, Q)$ and any $M_{CurrMax}$. Then the* MCS-Incremental *function will eventually enumerate $M_u = M \cup M_c$ and $C_i = \emptyset$.*

Proof. Let M be connected as well. The MCS-Incremental function will eventually enumerate $M_k = M$ whereas $C_k = \emptyset$. From Theorem 1 and Lemma 1,2 we know that all nodes $f \in F_k$ are either adjacent to any mapped node in M, part of M or unsuited to construct any mapping M_{k+1}.

Thus no node of M_c is forbidden. $F_k \cap V(M_c, Q) = \emptyset$. When MCS-INCREMENTAL is called with M_k and an empty candidate list, MCS-PROCEEDWITHNEXTCOMPONENT is called in line 4 which then continues with MCS-INCREMENTAL and a single node candidate list. Following Theorem 1, we can conclude MCS-INCREMENTAL will enumerate the component M_c as well resulting in the enumeration of M_u and an empty candidate list.

Now let M be a mapping represented by the union of several connected mappings M_c^i fulfilling the property that any two of them have no nodes in common and none of their nodes are adjacent to any node of the other mapping. Recursively applying the construction above, M will be eventually enumerated.

Lemma 6. *The upper bound estimation is valid for the dMCS.*

Proof. The upper bound estimation for the disconnected case is an extension of the upper bound estimation for the cMCS. The trivial upper bound estimation $e = |M_i| + |V(Q) \setminus F_i|$ remains a valid estimation. If there are no further constraints provided by $MaxComp$ and $MinSize$, then this estimation can be considered as tight, especially if there are enough mapping partners available for all remaining unmapped nodes of Q. $MaxComp$ and $MinSize$ constrain the problem such that the dMCIS of two graphs Q and T usually contains more nodes than the constrained dMCIS. Nonetheless the upper bound estimation remains valid. If $MaxComp$ is provided, then the upper bound considers the connected estimation and the remaining number of largest connected components of mappable nodes. If $MinSize$ is provided, the connected estimation plus the number of nodes in connected mappable components of at least $MinSize$ nodes are considered. If both constraints are provided, the criteria are applied both. This restriction of the upper bound calculation follows the restrictions of the constrained dMCIS calculation.

Theorem 3. *The dMCIS of two graphs Q and T is computed correctly by the* MCS *function.*

Proof. Let the MaxComp and MinSize parameters be set to reasonable values. Given the maximum mapping M_{max} that represents the dMCIS of Q and T whereby M_{max} can contain

several disjoint connected mappings M^i_{max}.

Similar to the cMCIS of Theorem 2 initially MCS calls MCS-ProceedWithNextComponent. This function then calls MCS-Incremental with $C = \{n\}$ for all nodes $n \in V(Q)$. Let v be the first node, that is also mapped in M_{max}. Here, $F_0 \cap V(M_{max}, Q) = \emptyset$. Following Lemma 5 M_{max} will be eventually enumerated, if the calculation is not bound by any upper bound estimation. During all relevant backtracking steps, the UpperBound estimation will at least estimate $|M_{max}|$. As long as not other current maximal mapping M'_{max}, $|M'_{max}| \geq |M_{max}|$ is found, M_{max} will be enumerated. If $|M'_{max}| = |M_{max}|$, then an alternative mapping of equal size was found which is a valid result as well. If $|M'_{max}| > |M_{max}|$ then M_{max} could not have been a dMCIS, since M'_{max} is a mapping.

Lemma 7. *The constrained dMCIS is correct.*

Proof. The MCS-Incremental function enumerates any mapping M component-wise. If a new component of M is started, the constraints for the components M^i of M are verified in line 2. The component-wise build-up of the solution allows to test the constraints before any extension with nodes of a new component. Given the correctness of the UpperBound function, correctness of the constrained dMCIS follows from correctness of the dMCIS.

Theorem 4. *The backtracking scheme is correct for the MCES.*

Proof. Based on the proof of Whitney[S1] an edge isomorphism of a graph corresponds to a node isomorphism with a few exceptions. The most prominent is the triangle triod exchange, which is commonly named Δ-Y-Exchange. Nicholson et. al.[S2] list three more examples explicitly where the line graph has more symmetry than the graph. This can lead to valid edge isomorphism that do not correspond to a node isomorphism.

If those listed cases are handled, it remains to proof that the MCES is correct. Based on the partial edge isomorphism of the mapping, the molecules Q and T can be separated into disjoint subgraphs Q^i and T^i with node sets $V(Q^i)$ and $V(T^i)$ and analogously defined edges whereas $i \neq j \Rightarrow V(Q^i) \cap V(Q^j) = \emptyset$. The same holds true for $E(Q^i), V(T^i)$ and $E(T^i)$

respectively. Now the disconnected MCES of Q and T is reduced to edge isomorphism of Q^i and T^i which is proven by Whitney.

Implementation Notes of the Δ-Y-Exchange Handling

The algorithm is separated into phases of connected extensions of the current mapping followed by an occasional disconnected extension. After each disconnected extension, the new component is extended until the next disconnected extension happens. This behavior allows for a component-wise test for triangle-triod exchanges. Based on Nicholson[S2] it is sufficient to handle a few cases for each connected component to avoid triangle-triod exchanges. Most prominent, the exchange of a triangle with a triod is detected counting the number of unique incident nodes for mappings with exactly three edges. The remaining cases can be described as a triangle with an additional link, a diamond and a tetrahedron. They are handled as follows.

1. Collection of all unique incident nodes to the mapped edges.
2. Collection of all node mappings that are induced by mappings of incident edges.
3. Test for uniqueness of the mapped nodes in the query and target graph.

There are two early abort criteria: If the number of incident nodes differs or there are more node mappings than nodes available a triangle triod exchange is detected.
An evaluation of our implementation detecting triangle triod exchanges on the examples provided by Nicholson[S2] revealed that such an exchange leads to several (invalid) node mappings finally exceeding the number of available nodes. Thus, the final test for uniqueness of the mapped nodes as not used but will be kept it for safety.

More Detailed Constrained dMCS Evaluation

Extended Evaluation of the dMCES Constraints on the EvalSet

This evaluation follows on the heatmaps in Figure 4 and 5. In general the MCS can have an exponential runtime behavior, such that the mean computation time can be heavily influenced by extreme outliers. We therefore plotted the times of the 90% quantile (see Figure S1). In comparison to Figure 4 many more dMCS configurations have runtimes well below one second, e.g. dMCS 3/3 is below 300 ms in 90% of the considered comparisons.

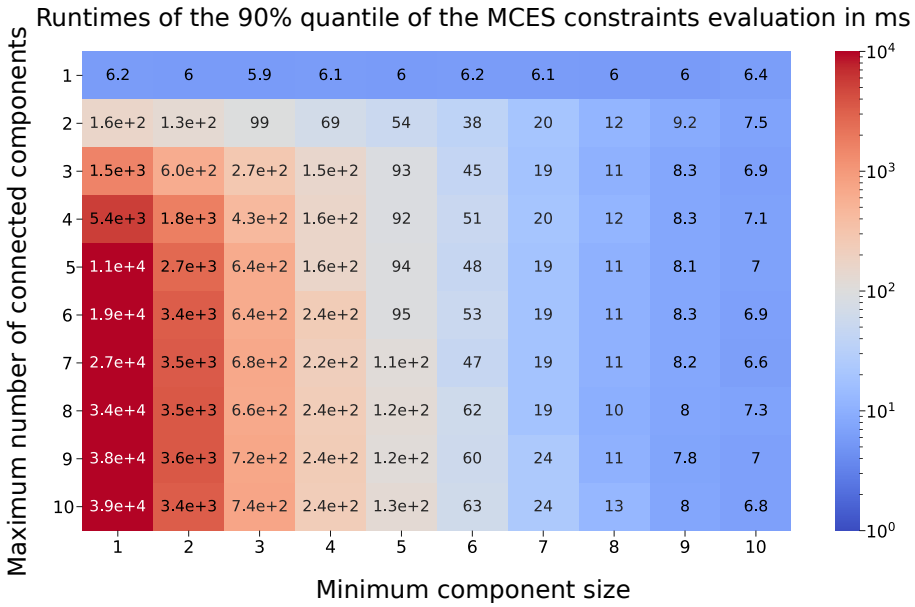


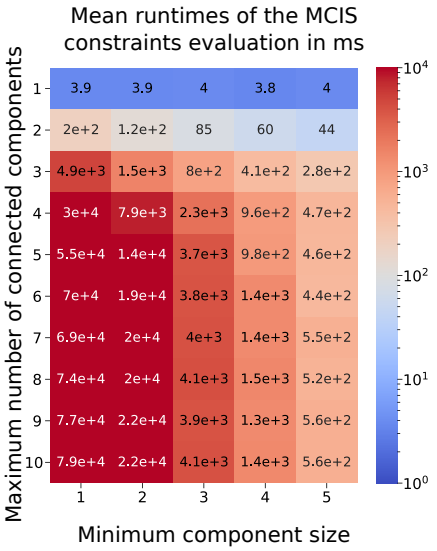
Figure S1: Times of the 90% quantile of the same experiment as shown in Figure 4. This plot shows that the mean times are heavily influenced by a relatively small number of outliers.

Evaluation of the dMCIS Constraints on the EvalSet

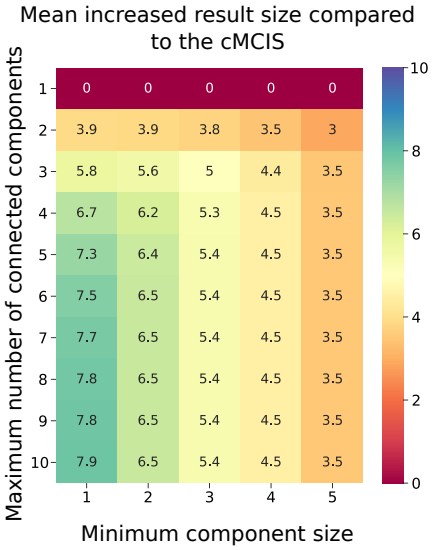
The influence of the upper and lower bounds of the connected component constraints is not only significant for the computing times of dMCES. We evaluated the performance of the dMCIS for up to ten connected components with a minimum atom count of one to five per connected component. As learned from the dMCES results in Figure 5 the mean increase in result size compared to the cMCS becomes less relevant beyond a minimum component size of five. The result of the dMCIS evaluation is shown in the subplots of Figure S2.

Figure S2a shows the mean computation time for all parameter combinations. The computing times follow the same trends as analyzed for Figure 4. If three or more components are allowed, the mean computing times are approximately twice as high as the times for the MCES. To estimate importance of the dMCIS it is necessary to evaluate the average number of additionally mapped atoms in comparison to the cMCIS, see Figure S2b. Similar to the average increase in result size for the dMCES of Figure 5 highest values are achieved for small minimum component sizes and many connected components. The comparison shows that for minimum component sizes up to four atoms the dMCES has a higher increase in number of mapped atoms. This changes for at least five atoms per components. Similar to the dMCES three to five more atoms can be mapped on average if a small number of connected components with a reasonable size of at least three atoms is allowed.

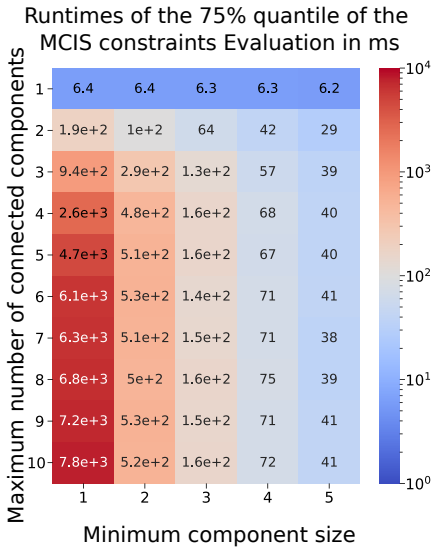
Finally the plot of the 75% quantile showing that except for the cMCIS the average computation times are above the 75% quantile, see Figure S2c. They are also dominated by outliers that need much more time. We see that the runtimes for minimum component sizes of at least four are well below 100ms and the favored configuration of three connected components with at least three atoms also about 130ms or less in 75% of the comparisons on the EvalSet.



(a) Mean computing times for the pairwise comparison on the EvalSet applying the constrained induced dMCIS.



(b) Result size comparison over the different constrained dMCIS comparisons on the EvalSet.



(c) The 75% runtime quantile of the constrained dMCIS comparison.

Figure S2: Exhaustive performance and result evaluation of the induced MCS on the EvalSet. The differentiation between mean runtimes and the 75% quantile shows the influence of outliers for the disconnected computation which is also relatively fast in the median case. The result size evaluation on the right reveals that significant improvements can be achieved for constraints that still have affordable runtimes.

Additional Tables for the Comparison of the Exact Algorithms

Table S1: Additional analysis of the average runtimes of Table 6. Here, two variants of RIMACS and the RDKit and exact cMCES algorithm of Indigo are compared. The RIMACS column represents RIMACS in its cMCES default configuration, RIMACS(R) uses a weaker bond compatibility criterion in order to produce results that are comparable in size to those of the RDKit. For each of the compound sets and algorithms the runtimes of the second and third quartile are listed.

	Measured Times in ms							
	RIMACS		Indigo(E)		RIMACS(R)		RDKit	
Quartile	q2	q3	q2	q3	q2	q3	q2	q3
EvalSet	1.31	2.44	4.46	8.86	2.91	7.88	4.53	11.73
NCI-KEY	0.53	1.20	1.41	3.80	1.10	3.46	0.91	3.29
NCI1	7.04	15.00	42.47	69.03	29.00	158.50	38.10	419.75
NCI2	29	55	52	64	440	668	933	10,669
NCI-S	2.23	4.11	17.91	39.20	4.80	14.00	5.80	36.00
CAH2	0.48	0.79	1.17	1.95	0.67	1.37	1.35	3.38
CDK2	1.14	1.84	3.69	6.22	3.38	6.87	5.16	12.65
NEU	0.78	1.23	3.13	3.79	1.50	2.00	2.57	5.09

Table S1 provides additional in depth information for the comparison of RIMACS and the other exact methods. Most important, the runtime difference between RIMACS and the external tools is less significant regarding the second and third quartiles. In Table 6, on average RIMACS is more than one order of magnitude faster than the RDKit or Indigo toolkit on the NCI1 and NCI2 compounds. On the NCI-S compounds this was also the case for indigo and RDKit was five times slower than the adapted RIMACS variant. In Table S1, one can observe that the average runtimes for Indigo and RDKit are more heavily influenced by extreme outliers than average runtimes for RIMACS. For the EvalSet, CAH2, CDK2 and NEU compounds, the runtimes of the third quartile of RIMACS are above the average runtimes indicating that outliers do not have much influence on the average runtime. For Indigo the same holds true on the EvalSet and the CDK2 and NEU compounds. For the RDKit solely the runtimes of the third quartile of the NEU compounds are above the average runtimes. In comparison to Indigo and RDKit RIMACS offers the fastest average and

quartile runtimes on the compound sets evaluated including an improved outlier behavior.

References

(S1) Whitney, H. Congruent Graphs and the Connectivity of Graphs. *Am. J. Math.* **1932**, *54*, 150.

(S2) Nicholson, V.; Tsai, C.-C.; Johnson, M.; Naim, M. A Subgraph Isomorphism Theorem for Molecular Graphs. In *Graph Theory and Topology in Chemistry*; Elsevier: Amsterdam, 1987; pp 226–230.

Maximum Common Substructure Searching in Combinatorial make-on-Demand Compound Spaces

[D2] Schmidt, Robert u. a.: „Maximum Common Substructure Searching in Combinatorial Make-on-Demand Compound Spaces“. In: *J. Chem. Inf. Model.* 61 (2021). DOI: 10.1021/acs.jcim.1c00640

http://pubs.acs.org/articlesonrequest/AOR-B2JVKD6UTVYGEDSPVJDK

JCIM JOURNAL OF CHEMICAL INFORMATION AND MODELING

pubs.acs.org/jcim Article

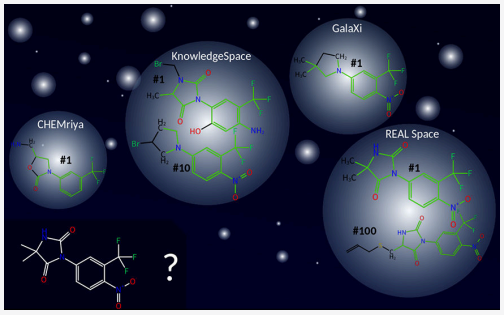
Maximum Common Substructure Searching in Combinatorial Make-on-Demand Compound Spaces

Robert Schmidt, Raphael Klein, and Matthias Rarey*

Cite This: J. Chem. Inf. Model. 2022, 62, 2133−2150 Read Online

ACCESS | Metrics & More | Article Recommendations | Supporting Information

ABSTRACT: Commercial make-on-demand compound spaces have become increasingly popular within the past few years. Since these libraries are too large for enumeration, they are usually accessed using combinatorial fragment space technologies like FTrees-FS and SpaceLight. Although both search types are of high practical impact, they lack the ability to search for precise structural features on the atomic level. To address this important use case, we developed SpaceMACS enabling efficient and precise maximum common induced substructure (MCIS) similarity and substructure searches within chemical fragment spaces. SpaceMACS enumerates a user-defined number of compounds in a multistep procedure. First, substructures of the query are extracted and matched to all fragments of the space. Then partial results are combined to actual compounds of the space. In this way, SpaceMACS identifies common substructures even if they cross fragment borders. We applied SpaceMACS on three commercial fragment spaces searching for the 150 000 most similar analogs to a glucosyltransferase binder from literature. We were able to find almost all building blocks used for the synthesis of the 90 listed analogs and a plethora of additional results. SpaceMACS is the missing link to enable rational drug discovery on make-on-demand combinatorial catalogs. No matter whether initial compound suggestions come from a de novo design, an AI-based compound generation, or a medicinal chemist's drawing board, the method gives access to the structurally closest chemically available analogs in seconds to at most minutes.



INTRODUCTION

Applications of the maximum common substructure (MCS) problem[1,2] in chemistry comprise a variety of use cases but are generally focused on small databases. Most prominently, the MCS can be used as a chemical similarity measure for screening compound catalogs for analogs. As special cases, the more general MCS algorithm enables also compound lookup and substructure searching. The queries are not necessarily limited to molecular fragments. Using generic descriptions like SMARTS,[3] also chemical pattern search can be established. MCS calculations on large compound collections often introduce heuristics providing upper bounds for MCS-based similarity. While this can lead to a substantial speedup, it comes at the expense of loosing precision.[4] The generic MCS problem usually comprises four variants. The common substructure can be either connected or disconnected, and further on, it can be induced or noninduced. For the induced MCS (MCIS) all adjacency relations between matched nodes must be identical. Regarding approaches solving the MCS problem, there are a few different types of algorithms calculating exact[4−9] or heuristic[10−13] results.

With the ongoing trend of fast increasing compound library sizes and the rise of combinatorial chemistry, the MCS has lost its attraction as a similarity measure. Since the exact calculation is computationally demanding, fingerprints are most often used instead.[14−16]

Nowadays screening compound collections with millions or even billions of compounds is a common use case in drug development endeavors. Especially large make-on-demand compound catalogs like REAL Space,[17] CHEMriya,[18] and GalaXi[19] are highly attractive as a starting point for drug design. In many applications, these collections are partially enumerated and screened sequentially for similar compounds, required substructures, or even with three-dimensional approaches like molecular docking.[20,21] Already in their current form, make-on-demand compound spaces are too large for enumeration requiring combinatorial fragment representations (Fragment Spaces) and custom-made search algorithms.[16,22] The FTrees-FS algorithm was the first of its kind enabling the exact search with a reduced graph similarity measure.[22] Recently, SpaceLight[16] was introduced as the first combina-

Special Issue: From Reaction Informatics to Chemical Space

Received: June 4, 2021
Published: September 3, 2021

torial search approach based on connected substructure fingerprints (CSFPs)[16] and Morgan fingerprints highly similar to the well-established ECFP[14] fingerprints.

Regarding future developments in accessible library size, the enumeration of whole fragment spaces becomes unfeasible and sublinear scaling methods that mimic existing similarity measures enabling searching on the atom level are an urgent need.[21] Commercial available compound libraries exceeding 19 billion compounds are already available growing several billion compounds per year by increasing growth rate.[23−26]

In comparison to the estimation of accessible chemical space[27] of certainly over 10^{60} even the largest public and commercial spaces are several orders of magnitude smaller.[28] Nevertheless, hardware requirements and scaling behavior of the search methods are key factors for future application scenarios. The impressive computation results for queries on large collections within seconds as provided by ZINC20[21] are achieved using hardware worth several hundred thousands of dollars. It is obvious that searching enumerated libraries will fail taking the growth rates into account; and in fact, they fail already today on the largest available make-on-demand catalogs. For comparison, combinatorial approaches like FTrees-FS[22,29,30] widely adopted in pharmaceutical research achieve results within minutes on affordable standard computers. The recently introduced SpaceLight approach[16] enables topological fingerprint similarity calculations within seconds as well, using a standard desktop computer. Even more important, the algorithms have the potential to handle the compound catalogs of the near future with probably 10^{14}−10^{20} make-on-demand molecules.

Recently, machine learning (ML) got a lot of impetus in early phase drug discovery. Substantial progress has been made in the development of generative approaches creating molecules with trained properties. Even if the chances are high that the resulting molecules are synthetically available, it remains questionable whether ML-based methods could or even should produce compounds ready-to-purchase from a make-on-demand catalog. A more reasonable approach seems to let ML-based methods focus on physicochemical and biological compound properties and use fast cheminformatics search engines as a final step to determine the closest purchasable analogs. A key technology to link machine learning to make-on-demand catalogs is therefore a fast, sublinear-scaling (maximum common) substructure search algorithm.

Here we present a novel approach named SpaceMACS, enabling efficient MCS-similarity searches in large combinatorial spaces. The basic algorithmic strategy follows a dynamic programming approach on trees as was applied for FTrees-FS as well (see ref 22). To increase variability, SpaceMACS allows many SMARTS features within the query structure. SpaceMACS provides results within minutes on standard computers with reasonable scaling behavior and has the ability to improve query computation time on larger multicore machines. The SpaceMACS algorithm is designed for the connected induced MCS problem restricting to cyclic/acyclic bond matching. Beyond that, it is capable of atomic-level molecule comparisons including substructure searches in combinatorial fragment spaces.

METHODS

Commercial fragment spaces[17−19,31,32] as provided e.g. by BioSolveIT[26] for Enamine,[23] OTAVAchemicals,[24] or WuXi LabNetwork[25] are built from molecular fragments with explicit linker positions. Typically hundreds of linker types model reaction chemistry details thus enabling combinatorial search. For algorithmic and model purposes, fragments are combined to treelike structures. While ring forming reactions can be modeled with sophisticated linker substitution strategies, by design, fragments of those spaces do not form macrocycles. Within the presented approach, acyclic bonds in the query molecule are only allowed to match acyclic bonds in the fragments. Ring bonds are handled accordingly to this restriction. Beyond that, it is not required for rings to be matched as a complete unit. The presented approach computes a connected maximum common induced substructure (cMCIS) using the RIMACS algorithm.[5] RIMACS is a substructure-based MCS algorithm, directly building up the actual mapping between compared molecules. It is designed to be a fast and lightweight algorithm offering simple adoptions to all sorts of graphs. Within the NAOMI[33] framework the performance is further improved relying on atom similarity estimations. Within the SpaceMACS approach, RIMACS is used to calculate a weighted cMCS. In this application, there are two crucial features of RIMACS. First RIMACS allows specification an initial mapping improving relevance of the calculated results and overall performance. Second, after calculation of a weighted maximal result, it is possible to update the weight. This feature is exploited to calculate the actual MCS variant used within SpaceMACS as described in the Fragment Matching section.

We illustrate the basic concept of substructure search in combinatorial fragment spaces within the following example searching for analogs of nilutamide[34] in REAL Space[17] (see Figure 1). Regarding the target compound and its decomposition into fragments, in the bottom of Figure 1, it is obvious that bonds to linkers are acyclic in any target structure. In order to guarantee independent calculations on different fragments, it is necessary to enforce that those bonds to linker

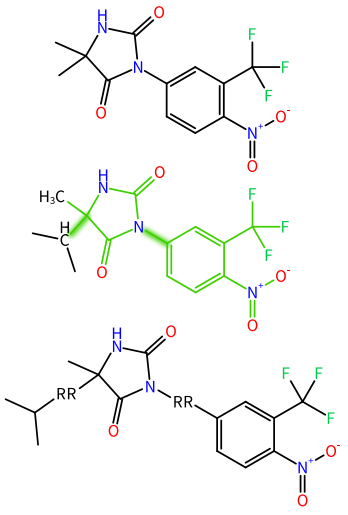


Figure 1. Nilutamide[34] as an example query (top) and one of its most similar analogs in the REAL Space[17] (middle) including its decomposition into fragments (bottom). The linker atoms in the fragments are labeled *R*, and bonds in the target molecule connecting the fragments are highlighted.

https://doi.org/10.1021/acs.jcim.1c00640
J. Chem. Inf. Model. 2022, 62, 2133−2150

atoms are mapped to acyclic bonds in the query as well. Keeping the model consistent results in strict distinction between ring- and chain-bonds. Additional distinctions between ring- and chain-atoms are applicable to improve performance, but do not affect the model. The decomposition yields the key observation resulting in the FTrees-FS and SpaceMACS algorithms: For a MCS expanding over more than a fragment, it is sufficient to consider only those partial MCS-results containing any linker of the fragment.

Besides the preparation of the input data, the MCS search in fragment spaces is divided into three sequential phases. First, parts of the molecule are matched to the fragments of the fragment space. Second, the matches are combined to construct all possible results. Finally, the results are enriched by in-fragment MCS-matches where no fragment-linker is involved. This workflow including the user interaction is shown in Figure 2.

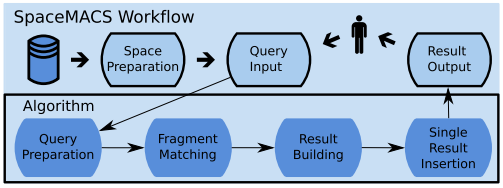


Figure 2. Overview of the SpaceMACS algorithm and the user interaction within the SpaceMACS tool.

DEFINITIONS

Within the SpaceMACS approach two MCS similarity measures are supported that are defined as follows. Given a query molecule Q with qa heavy atoms and a target compound T from the fragment space with ta heavy atoms and an cMCIS of size m. MCS-Size is represented by the pair (m, ta) and is sorted using the following ordering relation:

$$(m_1, ta_1) > (m_2, ta_2) \Leftrightarrow m_1 > m_2 \vee (m_1 = m_2 \wedge ta_1 < ta_2)$$

There is no standard way of defining a similarity value based on MCS. For the MCS-Similarity measure we decided on a definition which is close to the Tanimoto score (or Jaccard[35] index). For identical molecules s_{MCS} reaches the maximum value of 1.0. The larger the molecules get with respect to their MCS, the smaller s_{MCS} is with 0.0 as the minimum. MCS-Similarity is represented by the pair $(s_{MCS}, (m, ta))$, the similarity value and MCS-Size. $s_{MCS} = \frac{m}{qa + ta - m}$. MCS-Similarity is sorted hierarchically by its first and second member.

In the following, the SpaceMACS approach is illustrated using the MCS-Size measure which is supported inherently by the algorithm and structure of the space. Supporting the MCS-Similarity measure describing a more intuitive chemical similarity is handled later on in a dedicated section focusing on the differences in comparison to MCS-Size.

Fragment Space Preparation. Fragments can have several linker atoms also named linkers representing bonds to be formed between fragments. Within the approach, some of the linkers are matched and others are not. When handling partial results, the algorithm has to know the minimum size of any target compound containing the given fragment. This information is generated beforehand for an input fragment space in the following way.

For each linker type, the minimum number of heavy atoms added by any fragment having such a linker is collected. Initially, we consider only fragments with exactly one linker. Then, all fragments are taken into account and for each of its linkers, upper bounds estimating the minimum number of heavy atoms are applied. These calculations are performed for each linker and repeated on all fragments until convergence. Finally, this information is used to generate lists of the smallest compatible fragments, so-called *termination suggestion*. In this context, we define the term to *saturate* a fragment, meaning that links not matched by the MCS are extended with fragments from the corresponding list. This way, fragments are combined to valid molecules forming a *target* in the fragment space.

Fragment Matching. In general, an MCS match of a molecule to a target in a fragment space covers multiple fragments. The situation that the MCS is fully contained in one fragment is an almost trivial to handle a special case. These matches will be calculated later in the overall workflow such that the size of common substructures spanning over multiple fragments can be used as a lower bound on the expected result.

The MCS maximizes the number of mapped atoms between the query and an arbitrary combination of fragments from the space with no further restrictions. Within a single fragment, any linker atom is one further atom to match but could represent several more matched atoms in the final result. To handle this fact accordingly, we apply a weighted cMCS approach. Heavy atoms receive a weight of one and linkers are assigned a weight representing the maximum number of atoms matched in any molecule of the space respecting the current mapping location. This definition exploits the restrictions that acyclic bonds are forced to be matched to acyclic bonds and implies that linkers are incident to exactly one acyclic bond. For weighted MCS calculations, the RIMACS algorithm published earlier can be applied.[5]

The mentioned considerations lead to the following two-step procedure:

1. Generation of the bond processing order list: For this step the assumption that query molecules Q are connected directed graphs $Q = (A, B)$ with the atoms A and bond B whereas $B \subseteq A \times A$ is used. All edges have an antiparallel counterpart. As stated above, only those bonds that are acyclic are of interest.

   At first, we define a disjunct bond-based path whereas bonds are used only in one way:

   $P_B = ((a_1, a_2), (a_2, a_3), ..., (a_{n-1}, a_n)), a_i \in V$

   For a given bond $(a_0, a_1) \in B$ the set of reachable bonds is defined as

   $B_R(a_0, a_1) = \{(a_i, a_j) \in B | \exists P_B = ((a_1, a_2), ..., (a_i, a_j))\}$

   B_R is the set of reachable directed bonds starting at a_1.

   Cyclic bonds are not processed later on, but they might appear on paths. For the sake of simplicity those bonds are assumed to be processed already. For the following matching step, a bond $(a_u, a_v) \in B$ is to be processed, when all of its reachable bonds $B_R(a_u, a_v)$ are processed. This results in a dynamically adapting processing order starting at bonds pointing to terminal atoms toward the inner regions of the query molecule. Note that this order is well suited for parallelization.

https://doi.org/10.1021/acs.jcim.1c00640
J. Chem. Inf. Model. 2022, 62, 2133−2150

Figure 3 depicts four independent bonds for parallel processing of nilutamide, and one dependent chain bond

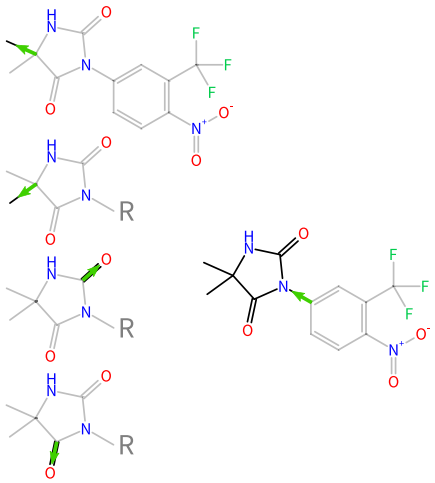


Figure 3. Depiction of the bond processing order dependencies for the chain bond connecting the two rings of nilutamide. The highlighted bonds in the (partial) depictions of nilutamide on the left side have to be processed before the highlighted bond in the right depiction is processed. Highlighting uses green arrows pointing in the direction of the dependent parts.

connecting the two rings. This bond is highlighted in the structure on the right side of Figure 3 and depends on all four bonds shown on the left.

2. Calculation of partial weighted cMCIS results on all fragments: For each directed bond (a_u, a_v) of the processing order list, the atom a_u is mapped to a linker in the fragment matched and the atom a_v is mapped onto the (single) adjacent atom of the linker in the fragment. Given this initial mapping the cMCIS is computed for each of the fragment linkers. The initially mapped weighted cMCIS results calculated by this procedure are stored under a so-called *bond-linktype key* = $((a_p, a_q), type)$ defined by the (atom-)pair of the directed bond and the linker type used for the initial mapping. For each *bond-linktype key*, results are sorted in descending order by the MCS-Size.

The weights for the MCS are calculated as follows: Any mapping of an atom a_q of the query onto any heavy atom of the fragment receives a weight of 1. Any mapping of a_q onto a linker atom a_l with linker l is weighted with the number of mapped atoms of the best possible extension. The weight calculation relies on the processing order, requiring that the MCS calculations of all reachable bonds $(a_i, a_j) \in B_R(a_p, a_q)$ are finished. Note that a_p is unique for each atom a_q mappable onto the linker atom a_l given the actual initial mapping. First, the keys of potential follow up results are enumerated using the acyclic bond (a_p, a_q), and all compatible linker types to l. The largest MCS-Size of any follow-up result defines the weight of the mapping of a_q onto a_l.

In order to store the results, their weight of the cMCIS calculation is extended to an MCS-Size. The MCS-Size of this fragment contains the MCS-Size of extension results and the sizes of all fragments used for linker saturation for all linkers except the initial one.

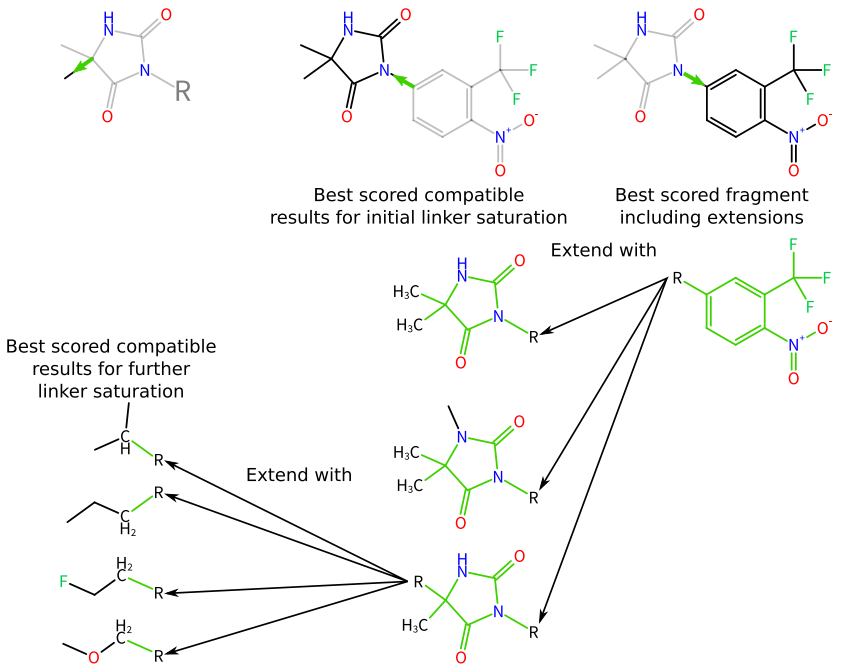


Figure 4. Result building procedure for nilutamide on REAL Space. The top row represents selected elements from the processing order defining the relative location of each result. Below that, top scored results for fragments are selected and used for further result extension until all linkers are saturated. Selection follows according to descending MCS-Size.

https://doi.org/10.1021/acs.jcim.1c00640
J. Chem. Inf. Model. 2022, 62, 2133−2150

This procedure allows using the MCS-Size as an additive measure for partial results.

If the initial acyclic bond (a_u, a_v) is not compatible to the bond in the fragment between linker and its neighbor, the MCS calculation is skipped. If any MCS result cannot be extended over the initial linker, then it represents either an MCS within the fragment which is calculated later on, or there is another linker that produces the MCS result.

As an optimization the restriction of acyclic bond matching can be exploited such that compatible atoms are required to have the same distance measured in acyclic bonds to the initially mapped atoms in the query and the fragments.

Usually also target compounds having an MCS to the query of suboptimal MCS-Size are of interest. In order to find these target compounds as well, we modified the algorithm to solve the following MCS problem variant.

First, we calculate the so-called local weight as the maximum MCS without any linker atoms mapped. All maximal weighted MCS results whose weights are greater or equal to the local weight are clustered by mapped links of the current fragment $\{(a_q, a_l), ..., (a_p, a_k)\}$. From each cluster, the maximum result is selected and stored for the respective *bond-linktype key*.

In order to remain within reasonable runtime and memory constraints the number of results stored per *bond-linktype key* is limited and depends on user preferences.

RESULT BUILDING

For each acyclic bond (a, b) of the query, two lists of intermediate fragment results with descending MCS-Size have been computed, one list considering the bond in direction from a to b, the other in the opposite direction from b to a. Following link compatibility, the results are grouped in pairs extending them with the best result of their opposite bond and any compatible linker type. Those list pairs are then processed in descending order by the combined MCS-Size. The enumeration is limited by a user-defined expected number of results. According to this parameter, the resulting target molecules are created by an exhaustive recursive enumeration scheme. Figure 4 depicts the result-building procedure on the example of nilutamide on the REAL Space.

After the initial combination of two fragments, the generation handles first the matched and second the unmatched links of the fragments. Finally, the fragment combination is canonized and stored. The whole generation process is implemented as a depth first search backtracking procedure on the partial results and their result extensions. The single steps are as follows:

Matched Link Extension. Each matched link is considered. The result extension considers all compatible fragments in descending order by MCS-Size. This procedure also includes link extension for all alternative results with different link extensions for the same fragment.

Unmatched Link Extension. Each unmatched link is extended using fragments of the *termination suggestion* for the respective linker.

Result Building. If there are no more unprocessed links, the resulting target compounds are canonized by sorting the fragments by their internal IDs and internal position of the linker atoms.

Early Abort Criteria. If any partial result is worse than the currently last accepted result, the extension is stopped. Since fragment combinations can in theory result in identical targets, the result list exceeds the requested number by a factor of 5. In order to avoid nondeterministic behavior, even more results are stored as long as the MCS-Size is identical.

Finally, the target compounds are converted and sorted lexicographically by unique SMILES.[36,37] After deduplication, all final target compounds are sorted by MCS-Size and unique SMILES as secondary criterion. Deduplication is necessary since the same target compound might be enumerated with different MCS-Size values resulting from different fragment combinations. The specific enumeration of results is explained in more detail in the Supporting Information chapter 1.

In-Fragment Result Insertion. The initial assumption that a common substructure extends over at least two fragments, might not hold true in all cases. To accommodate for this, the algorithm offers a postprocessing step inserting single fragment results. All results generated in the following are limited by the constraint that linkers are not considered as part of the MCS. Since the achievable MCS-Size is limited by fragment size, single fragment matching is performed as a follow-up step using the MCS-Size of already achieved multifragment results as a lower bound. Before matching a fragment, a simple element and bond counting heuristic results in an upper bound of the MCS-Size. The cMCIS is computed for fragments as long as the estimated MCS-Size is larger or equal to the last ranked result. For the resulting fragments, all links are saturated and the final target compounds are inserted.

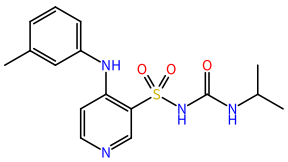
MCS-Similarity. The presented algorithm is designed to find the target compounds with maximum MCS-Size as defined above. In the following, we extend the algorithm to optimize the more intuitive MCS-Similarity measure. Using MCS-Similarity does not change the structure of the SpaceMACS algorithm. Fragment matching is handled identically for MCS-Size and MCS-Similarity. The enumeration procedure for the result generation requires adjustments and during single-fragment MCS-Result insertion, MCS-Similarity is used instead of MCS-Size as the fragment and result ordering criterion.

The most similar compounds can vary significantly when searching the top results using MCS-Similarity instead of MCS-Size. The enumeration employs abort criteria exploiting the additive nature of MCS-Size. In order to keep promising combinations of partial results within the enumeration procedure, an upper bound for MCS-Similarity values based on the number of mapped atoms and the total number of heavy atoms is computed. The upper bound approximation results from the fact that matched linkers occur for which alternative results with slightly lower number of mapped atoms and significantly lower number of heavy atoms might exist.

Calculating Upper Bounds for MCS-Similarity. Each partial result is represented as an (implicitly) single-linked list of *ResultNodes*, each storing a fragment with the respective MCS matching, the previous ResultNode and additional size information. Most important members of those nodes are the actual MCS-Size and the MCS-Size after saturation of all linkers. For the approximation of the MCS-Similarity two additional sizes, the base estimation and the final estimation are stored within the ResultNodes. The base estimation is the sum of the current MCS-Size, local maximum similarity estimations e_{link} for the matched links and the MCS-Size of the saturated open links. For a matched link with an extension of

https://doi.org/10.1021/acs.jcim.1c00640
J. Chem. Inf. Model. 2022, 62, 2133−2150

Query:



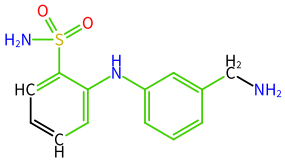
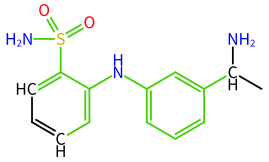
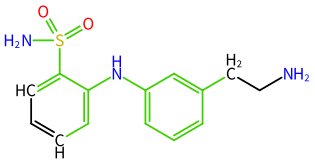
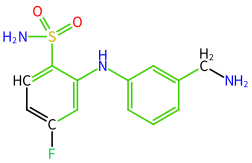
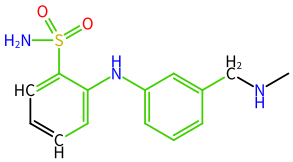
M	TS	Sim	Result Pos (Similarity)	Compound	Result Pos (MCS-Size)
17	19	0.65	1		8
17	20	0.63	2		9
17	20	0.63	3		10
17	20	0.63	4		11
17	20	0.63	5		12

Figure 5. Top five results for MCS-Similarity for the query Q1 on GalaXi[19] including the result positions of the results when using MCS-Size. The result position of the shown molecules using MCS-Size are displayed to the right of the molecule depictions. The number of mapped atoms is shown in column M, and the number of heavy atoms of the displayed molecules is shown in column TS.

MCS-Size (m_l, ta_l) and a *termination suggestion* for the link with $ta_{\min}$ heavy atoms, e_{link} is calculated as follows:

$$e_{\text{link}} = \begin{cases} (m_l,\ ta_l) & ta = ta_{\min} \\ (m_l - 1,\ ta_{\min}) & \text{otherwise} \end{cases}$$

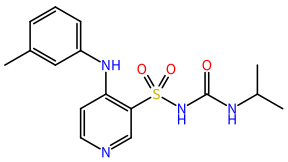
For case that $ta_l \neq ta_{\min}$, the difference $ta_l - ta_{\min}$ is stored in a extension delta value list L_Δ. The list L_Δ is finally used to calculate the similarity estimation. Initially the similarity estimation sim_{est} is set to the base estimation. Then, for each delta value $d \in L_\Delta$ in ascending order, the similarity estimation is maximized: $\text{sim}_{\text{est}} = \max\left(\frac{m}{t+q-m}, \frac{m+1}{t+d+q-(m+1)}\right)$.

Beyond the similarity estimation, the remaining parts of the enumeration procedure are the same as for MCS-Size. Using the estimation emphasizes that MCS-Similarity depends on the computation of nonoptimal results during fragment matching. The parameter results stored per *bond-linktype key* is crucial here as well.

Supporting Non-Molecular Queries. In many applications, a higher level of flexibility than searching with a query molecule is required. Often, bioisosteric replacements are applied for heterocycles or functional groups. To optimally support such queries within SpaceMACS, SMARTS-like queries are supported. Due to the nature of fragments being partial molecules SpaceMACS cannot support the full SMARTS[3] language. For example, SMARTS recursion ("[$(<SMARTS>)]") and disconnected SMARTS patterns require knowledge about the molecule structure beyond fragment borders. These SMARTS features therefore violate locality of the results when matching fragments. Beyond that, for SpaceMACS SMARTS expressions have to be rewritten such that each edge is either explicitly cyclic or acyclic. In order to come as close as possible to user intensions, Kekulé forms

https://doi.org/10.1021/acs.jcim.1c00640
J. Chem. Inf. Model. 2022, 62, 2133−2150

Query:



M	TS	Sim	Result Pos (Similarity)	Compound	Result Pos (MCS-Size)
18	36	0.43	3405		1
18	37	0.42	5556		2
18	37	0.42	5557		3
18	37	0.42	5558		4
18	38	0.41	6022		5

Figure 6. Top five results for MCS-Size for the query Q1 on GalaXi[19] including the result positions found when using MCS-Similarity. The result positions of the shown molecules using MCS-Similarity are displayed to the left of the molecule depictions. The number of mapped atoms is shown in column M, and the number of heavy atoms of the displayed molecules is shown in column TS.

are detected and the aromaticity of the affected nodes and edges is adjusted within the SMARTS patterns. To do so, SpaceMACS relies on the SMARTS inspection abilities of the SMARTScompare approach.[38,39] In conclusion the following SMARTS features are forbidden within the SpaceMACS approach:

- SMARTS-recursion
- Disconnected SMARTS
- Chirality
- Specifying explicit hydrogen atoms "[H]" or "[#1]". Note that hydrogen counts are allowed, e.g., "[CH3]"

Note that edges within SMARTS patterns are explicitly made cyclic or acyclic based on the SMARTScompare analysis[38] of the edge environment in the pattern. Any other aspect of the SMARTS language is supported by the SpaceMACS algorithm.

Although queries are in summary only SMARTS-like, a substantial degree of flexibility is achieved. For example, various substitution patterns in heterocycles, different halogenation patterns, or carbonyl−sulfinyl exchange can be easily modeled in individual queries. In order to avoid confusion with patterns that are valid SMILES and SMARTS, the SpaceMACS tool accepts "SMILES" and "SMARTS" followed by a space as prefixes for queries supplied on the command line or during interactive sessions.

SMARTS-like queries within SpaceMACS are handled similar to molecular queries, meaning MCS-Size and MCS-Similarity calculations are supported. In order to fulfill the notion of matching SMARTS patterns, we introduce an additional matching mode named *substructure*. The substructure mode in general behaves as MCS-Size, but applies the additional constraint that the query must be fully contained within the target compound of the fragment space.

https://doi.org/10.1021/acs.jcim.1c00640
J. Chem. Inf. Model. 2022, 62, 2133−2150

Online Result Filtering. In many use cases, results are filtered down by simple properties in postprocessing steps. Within the algorithm, filters for the minimum number of mapped atoms, minimum similarity values and minimum and maximum number of target compound atoms can be applied during enumeration of the results and in-fragment MCS mappings.

Result Storage. The overall number of results SpaceMACS produces is user-controlled, requiring knowledge about the acceptance bound, i.e. the least scoring target molecule still contained in the result list. Fortunately, MCS-Size and MCS-Similarity usually have only a limited number of different, discrete values. Assuming a query size of 30 and target size up to 50 atoms, there are at most 1500 different values for MCS-Size and MCS-Similarity. In practice, less than 100 different score values are expected for almost any query and a reasonable number of results. This observation allows counting the number of results per score. By storing the occurring scores in an ordered container, efficient result position estimations are possible without the necessity to store all results in an ordered datastructure.

Parallelization. When implementing the SpaceMACS algorithm, several steps can be performed in parallel. All preparation or conversion tasks can be easily parallelized. In the fragment matching step, all partial results are ordered by score. To achieve overall low runtimes, it is advantageous to compute several results without sorting and sort and merge results at defined synchronization points. Some directed bonds of the query are independent of each other. They are processed in parallel without the need of synchronization. A simple scheduling mechanism suffices since completion of some directed bonds enables computation of other ones if all their dependencies B_R are met.

The result enumeration using the partial mapping results is difficult to run in parallel. For small numbers of results, this step usually takes a few milliseconds. When computing thousands of results, the enumeration can easily cause more than 50% of the overall runtime (See Figure S6 in the Supporting Material). An effective parallelization scheme performs result enumeration for each pair of partial results in a single thread. Synchronization happens whenever new results get stored. Up to this point the result lists are sorted by score thread-local. Then, the overall list of results get updated.

The final task of single fragment result insertion performs the prescoring and sorting of the fragments as well as the enumeration of results in parallel, whereas a thread handles all results belonging to a specific fragment.

RESULTS AND DISCUSSION

Before entering into extensive statistical validation experiments, we want to demonstrate the practical use of MCS-Size and MCS-Similarity search on chemical fragment spaces with a first example. The aim is to find molecules structurally similar to Torasemide (Query Q1) in the commercial make-on-demand GalaXi space. After about one second on a standard computer, the molecules as shown in Figures 5 and 6 were retrieved.

Both figures display the size of the mapping (M), the number of heavy atoms of the target molecules (TS), and the respective similarity values. The top 5 results using MCS-Similarity feature smaller numbers of common heavy atoms but show greater overall chemical similarity due to overall smaller numbers of unmatched heavy atoms. MCS-Similarity has the advantage over MCS-Size that the number of unmatched heavy atoms also has a substantial influence on the similarity value. As can be seen in Figure 6, maximizing the number of mapped atoms alone can result in finding substructures within much larger target molecules featuring numerous unmatched side chains. Especially when requesting small numbers of results, we deem molecules retrieved using MCS-Similarity more relevant than those retrieved using MCS-Size. As will be shown later on, MCS-Similarity searches require more resources. Therefore, if large numbers of molecules are requested, the maximum number of heavy atoms in the resulting molecules can be limited. This way results are fairly similar to MCS-Similarity without the computation and approximation overhead described in the Methods sections and shown within the benchmarks.

For validation we consider three different scenarios. First, we compare the results of the combinatorial SpaceMACS search to an MCS search in an enumerated variant of the space. Second, we perform a sequential search to measure the hit list consistency. Finally, we evaluate the divergence of results with varying lengths of result lists.

Enumerated Space Result Comparison. For the first experiment, a fragment space is created which is small enough to be fully enumerated. We evaluated the best scored N results when our method is applied on the small fragment space and compared to the results achieved by searching linearly in the enumerated list of compounds. We enumerated two (arbitrarily extracted) subspaces of the KnowledgeSpace,[31,32] a public fragment space comprising more than 100 literature reactions describing hundreds of trillions of virtual compounds. Then we enumerated all compounds using a customized version of FSees.[40] The extracted spaces as well as their enumerations are part of the validation_spaces.zip file in the SI. As queries, we used the 100 query molecules selected by Lessel and Lemmen[41] for a novel fragment space similarity measure. We compared the 5000 most similar analogs of each query in the fragment space and the enumerated space. For each query, space and the three mapping modes we performed 10 independent runs. We compared the results for the ring-chain-atom compatibility set to "on" and "off" in five different filter scenarios. For the scenarios, the minimum number of mapped atoms is abbreviated MinM, the minimum number of heavy atoms in the target compound is minT and the maximum number of heavy atoms in the target compound is maxT.

1. No filter
2. MinM 10 (not for MCS-Similarity); minT 15; min similarity value 0.25 (only for MCS-Similarity)
3. MinM 12 (not for MCS-Similarity); min similarity value 0.3 (only for MCS-Similarity)
4. MinT 15; maxT 25
5. MinM 15 (not for MCS-Similarity); min similarity value 0.35 (only for MCS-Similarity)

Our default parameter set detected no differences on the 500-fragment (~200 000 compound) space. On the 1000-fragment space (~400 000 compounds), we observed different results for query 39 in MCS-Size mode and ring chain atom compatibility set to "true". For this query, the best MCS-Size score is (12, 36), and overall 4494 results have an MCS-Size greater than or equal to (12, 55). Here 90 suboptimal compounds below rank 4500 with MCS-Size values of (11, 34), (11, 35), and (11, 36) were not generated. Instead,

https://doi.org/10.1021/acs.jcim.1c00640
J. Chem. Inf. Model. 2022, 62, 2133−2150

SpaceMACS listed 90 additional results of MCS-Size (11, 37) meaning that those 90 results got lost during enumeration. Those missing results occur in the filter settings one and two. The reason is that the list of temporary results in the algorithm is limited. In summary, the experiment shows that SpaceMACS is exact on high ranks, but differences might occur on low ranks of the result list. The influence of temporary list lengths will therefore be further analyzed below.

Enumerated Space Validation. To further validate self-consistency, we queried the 101 most similar compounds of the space for each compound of the space and validated that the query itself was best scored and all remaining results are also part of the enumerated space. This validation was performed for all three modes on both enumerated spaces for both settings of ring- and chain-atom compatibility. This experiment shows that if a compound is part of a fragment space, then SpaceMACS will find it independently of the mode parameter.

Result Divergence with Varying List Lengths. While SpaceMACS produces exact results on high ranks, molecules on lower ranks might diverge between runs with different settings with respect to storage limits. We used four fragment spaces[17−19,31,32] and requested 100, 1000, and 10 000 results for 100 queries storing 1, 10, 100, 1000, and 10 000 intermediate results for matching. As queries, we used the same 100 queries as for the enumerated space comparison experiment.

In total, we performed two types of analysis: first, consistency of the top ranked results within the applied storage limit and, second, analysis of the smallest rank of different results between adjacent storage limit settings.

First, we evaluate the results retrieved using MCS-Size. Table 1 lists the number of queries for which any result difference was detected applying the listed result-storage-limit requesting 1000 or 10 000 results. The entry pairs show the number of queries with different results and additional average number of different results within the top 1000 ranks requesting 1000 and 10 000 results. With KnowledgeSpace and REAL Space, the results are consistent for all queries and all storage limit values analyzed. On CHEMriya there are five to seven queries where deviating results are observed. One could relate this divergence to the structure of the CHEMriya space that contains approximately twice as many fragments as the GalaXi space but describes more than five times as many compounds. This implies that for each fragment of CHEMriya, there are far more results that could be enumerated than for GalaXi. On GalaXi there are four and two queries with deviating results applying storage limits deemed to be too small.

Table 1. Overview of the Number of Queries with Different Results within a Selected Storage Limit for the Four Spaces Using MCS-Size Enumerations[a]

limit	CHEMriya[18]	GalaXi[19]	KnowledgeSpace[31,32]	REAL Space[17]
1	5/158.8	–	–	–
10	7/164.3	4/16.5	–	–
100	5/191.0	2/28.5	–	–
1000	5/191.0	–	–	–
10 000	5/191.0	–	–	–

[a]Each entry contains the number of queries with any difference and the average number of different results when requesting 1000 and 10 000 results, or – if no difference was observed.

Table 2 shows the retrieved results using MCS-Similarity. For the REAL Space, almost a third of the queries have minor differences. But the total number of different result molecules is small with less than 10 single-point differences on average for any of the queries. This shows that the approximation within the enumeration for MCS-Similarity is more sensitive to temporary list length limitations than the straight MCS-Size enumeration. This trend continues on all of the other three spaces and all the storage limits tested.

Table 2. Overview of the Number of Queries with Different Results within a Selected Storage Limit for the Four Spaces Using MCS-Similarity Enumerations[a]

limit	CHEMriya[18]	GalaXi[19]	KnowledgeSpace[31,32]	REAL Space[17]
1	3/77.7	4/92.3	1/2.0	29/4.3
10	9/43.4	6/21.3	4/9.3	36/7.4
100	12/37.3	6/7.7	7/20.9	35/8.5
1000	8/53.4	4/21.0	8/17.9	31/9.4
10 000	8/59	4/21.0	8/17.8	31/9.6

[a]Each entry contains the number of queries with any difference and the average number of different results when requesting 1000 and 10 000 results, or – if no difference was observed.

Finally Figure 7 points out that the result-storage-limit parameter should be chosen with care. Figure 7a plots the average rank of the first difference in the result lists using the storage limits 100 and 1000 and for the storage limits 1000 and 10 000 requesting 10 000 results. For MCS-Similarity, the first difference often occurs well before rank 500 for the CHEMriya space. On the opposite, for the REAL Space on average about the first ∼2000 ranks are identical. The situation is quite similar for MCS-Size but with less variance. This plot emphasizes the need to store a reasonable number of intermediate results when requesting large numbers of results. Regarding Figure 7b we observe similar trends between MCS-Similarity and MCS-Size. For MCS-Similarity, the first difference occurs on average beyond rank 1000 for all four spaces, but there are still outliers with much smaller ranks going even below rank 500. For MCS-Size instead, there are almost no differences below rank 2000.

Based on our experience, we recommend using a limit fitting roughly the number of requested results. For MCS-Similarity, the limit should be slightly higher, e.g. factor 2, and for MCS-Size it is affordable if the limit is slightly lower than the number of requested results.

Overall our validation experiments show that the "native" MCS-Size measure works reliably producing consistent results on the commercial spaces too large for enumeration and finds the same results on small enumerated spaces compared to a linear search. Furthermore even the MCS-Similarity measure proves to be fairly consistent on the spaces and performed without any difference on the comparison to the enumerated spaces. But on the commercial spaces MCS-Similarity starts to become heuristic in the case that temporary list lengths are chosen too small. With temporary list lengths close to the number of retrieved results, deviations occur only on very high ranks.

External Comparison. Inspired by the enumerated space comparison validation experiment we performed an experiment comparing SpaceMACS results on the 500 fragments

https://doi.org/10.1021/acs.jcim.1c00640
J. Chem. Inf. Model. 2022, 62, 2133−2150

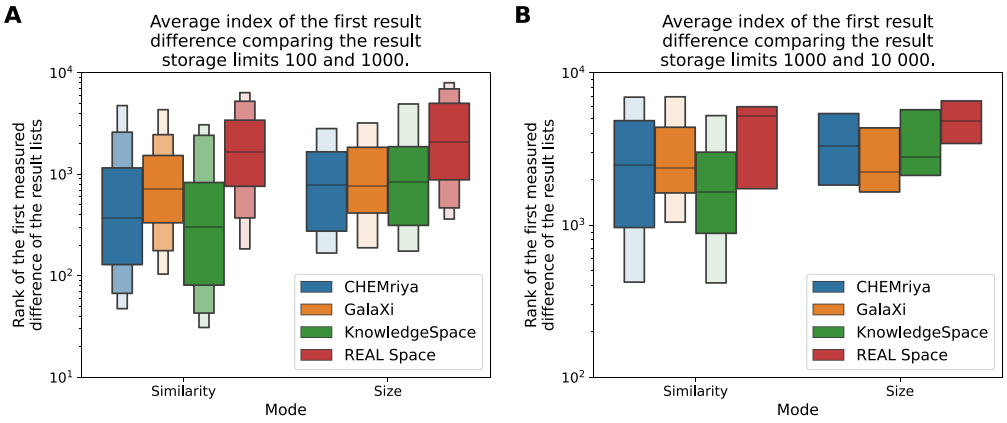


Figure 7. Log-scale plots showing the average rank of the first occurrence of different results increasing the result-storage limit from 100 to 1000 (A) and from 1000 to 10 000 (B) when requesting 10 000 results from different fragment spaces using MCS-Similarity and MCS-Size.

space to results retrieved using the RDKit FMCS[13,42] algorithm on the ∼200 000 enumerated counterpart of the fragment space. SpaceMACS mimics an exact approach calculating a constrained cMCIS, and FMCS calculates a cMCES represented as SMARTS. Applying the RDKit FMCS algorithm on all ∼200 000 compounds required more than 750 s of computation time for the queries on average restricting ring- and chain-bond compatibility. Extracting the best 5000 results using SpaceMACS took 21.9 s on average and 16.5 s when ring- and chain-atoms are allowed to be matched. Comparing the result lists there are five queries with identical result lists between RDKit and SpaceMACS allowing ring- and chain-atoms to be matched. On average there are 1803 common results between RDKit and SpaceMACS and 2134 when ring- and chain- atoms are allowed to be matched. When calculating the cMCIS using RIMACS within the NAOMI library there are on average 2206 common results when ring- and chain- atoms are allowed to match. A detailed analysis for selected cases can be found in the Supporting Information (see Figure S7).

Hardware Requirements and Parallelization. The SpaceMACS software implementing the algorithm described above loads a fragment space and then matches a number of queries. In order to reduce waiting times for loading the fragment space the tool supports interactive query processing. Therefore it accepts input from standard input and tries to parse it as SMILES[36] or SMARTS[3] or load it as a molecule file. The interactive mode supports manipulation of all important parameters as well. To work with spaces like the REAL Space[17] the machine should be equipped with at least 32 GB of main memory. The parallelization benchmarks were performed using two Intel(R) Xeon(R) Gold 6142 CPU @ 2.60 GHz processors with 32 cores and 64 threads in total and more than 200 GB of memory running a 64-Bit Suse Linux.

As a demonstration for the scaling abilities of the algorithm, we performed experiments on all four fragment spaces provided by BioSolveIT, meaning the commercial CHEMriya[18] space (∼52 000 frag, 1.1×10^{10} comp), the commercial GalaXi[19] space (∼22 000 frag, 2.1×10^{9} comp), the public KnowledgeSpace[31,32] (∼320 000 frag, 2.9×10^{14} comp), and the commercial REAL Space[17] (∼890 000 frag, 1.9×10^{10} comp). Each of the spaces were searched for the most similar 1000 and 10 000 results for the computation modes MCS-Similarity, MCS-Size, and substructure for the same 100 queries as used for our validation. The results are presented in Figure 8 whereas we selected one case for each fragment space.

In all selected test cases, SpaceMACS shows a reasonable scaling behavior using one to eight threads in parallel. Using 16 or even 32 threads does not gain much runtime performance any more. On small spaces, runtimes are in the area of seconds for small numbers of results independent of the score being MCS-Size or MCS-Similarity. Although KnowledgeSpace contains far more compounds than REAL Space, its fragments representation is more compact. Requesting 1000 results on this space provides a measurable distinction between the MCS-Size and MCS-Similarity measures. Additionally the error bars show that there are a few outliers and one extreme outlier requiring about ten times the mean runtime for all numbers of threads analyzed. The two most significant outliers are shown in Table 3. Runtimes show that those two queries are not outliers when using MCS-Size. On other spaces like the REAL Space, they are no runtime outliers at all. Runtimes of those outlier cases have in common that the enumeration of the results takes a significant amount of time, getting even worse requesting 10 000 results. Most probably, this phenomenon is related to an overestimation of MCS-Similarity values for intermediate results.

Finally the commercial REAL Space (Figure 8c) shows the most desired scaling behavior retrieving 10 000 results with runtimes going down to almost using 32 threads. Results for the CHEMriya space are interesting in another point. In comparison to REAL Space, the number of fragments is more than an order of magnitude smaller, but the space contains more than half the number of compounds as REAL Space. On CHEMriya, we observe a negative scaling behavior selecting larger numbers of results using more then 16 threads using MCS-Similarity. For MCS-Size average runtimes of ∼10 s are reached for 16 and 32 threads. Overall, computation of result applying the MCS-Similarity measure proofs more complex as MCS-Size. Beyond the overall scaling behavior, runtime partitions of the three major steps of the algorithm are analyzed in Figure 9.

In the first experiment, a single thread is used running the algorithm in substructure mode. The computing time is dominated by fragment matching taking only 1−2 s for the calculation independent of the requested number of results.

https://doi.org/10.1021/acs.jcim.1c00640
J. Chem. Inf. Model. 2022, 62, 2133−2150

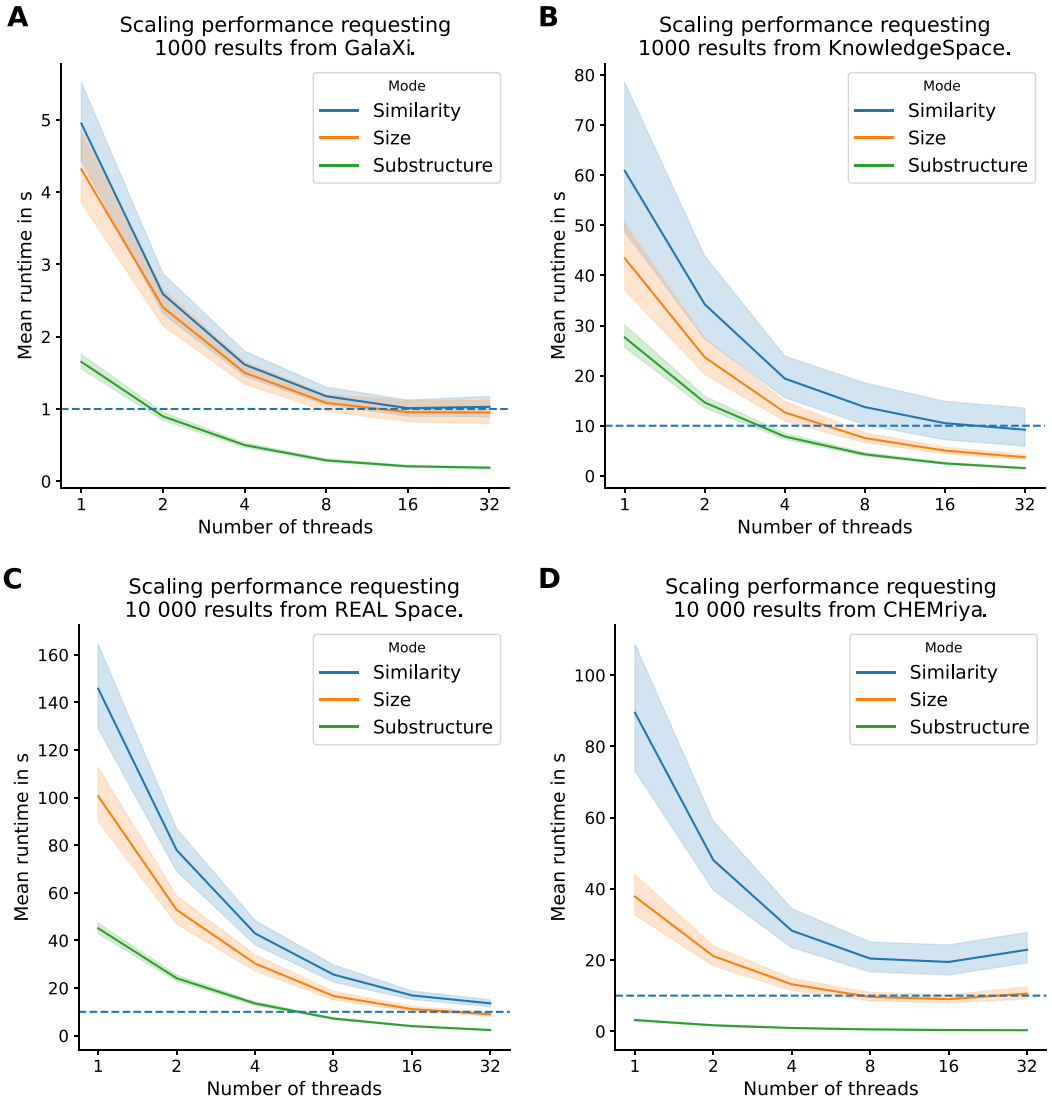


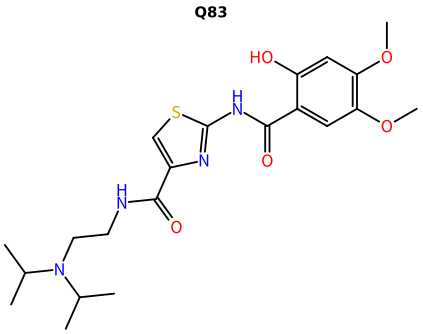
Figure 8. Scaling overview of SpaceMACS requesting 1000 results from commercial GalaXi (A) and public KnowledgeSpace (B) and retrieving 10 000 results from commercial REAL Space (C) and commercial CHEMriya (D) running 1 to 32 threads in parallel. For GalaXi (A) runtime of 1 s is shown for reference. For KnowledgeSpace (B), REAL Space (C) and CHEMriya (D) runtime of 10 s are shown for reference.

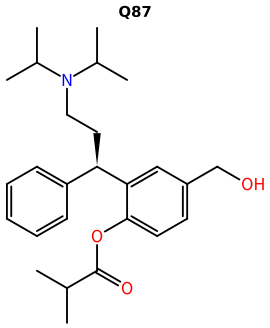
One has to keep in mind though, that several query molecules are not contained in GalaXi resulting in empty result lists (Figure 9a). When running in MCS-Size mode, here on KnowledgeSpace using four threads, we observe that the computing of fragment matching is nearly constant and independent from the number of requested results as expected. The structure of this space leads to increasing runtime portions for result enumeration when increasing the number of requested results (Figure 9b). On REAL Space using MCS-Similarity the distinction of the runtime partitions looks a bit different. Beyond the near-constant time required for matching, runtimes required for result enumeration and single result insertion seem to scale proportionally to each other (Figure 9c). Finally the plot of the CHEMriya space is focused on the limits of the scaling abilities of SpaceMACS. In this case, we requested 50 000 results and plotted runtimes for different numbers of threads. The plot shows that runtimes are finally dominated by result enumeration or in this case single fragment matches. The observed stagnation on runtimes hints that finally internal synchronization of the results becomes the bottleneck (Figure 9d). For more information about the scaling behavior requesting 50 000 results, consider Figures S5 and S6 in the Supporting Information.

Memory Requirements. As another important point, we measured the amount of memory consumed by SpaceMACS during benchmarking of the 100 queries. The results are shown in Table 4. This table shows the minimum required memory preparing the space and the maximum observed memory consumption requesting 10 000 results for the 100 queries used for validation. Due to differences within the enumerations for MCS-Size and MCS-Similarity, the table summarizes memory consumption for both modes.

https://doi.org/10.1021/acs.jcim.1c00640
J. Chem. Inf. Model. 2022, 62, 2133−2150

Table 3. Detailed Runtimes of the Two Most Significant Outliers of the Similarity Calculation on KnowledgeSpace[a]





KnowledgeSpace				
no of threads	4	16	4	16
MCS-Size	12.0s/3.3s/0.0s	4.41s/3.18s/0.01s	12.0s/15.0s/0.3s	4.4s/15.0s/0.1s
MCS-Similarity	12s/120s/0s	4s/120s/0s	13s/164s/1s	4s/157s/0s
REAL Space				
no of threads	4	16	4	16
MCS-Size	23.0s/1.0s/0.7s	8.10s/0.95s/0.22s	27.0s/2.7s/4.7s	9.3s/2.6s/1.5s
MCS-Similarity	24.0s/1.6s/1.4s	8.1s/1.6s/0.5s	27.0s/8.9s/9.3s	9.2s/8.3s/3.1s

[a]Runtimes are measured requesting 1000 results. Each runtime block is structured as follows: fragment matching/result building/single fragment MCS-Results.

First, we noticed that the amount of memory required for loading the space clearly shows a dependence on the number of fragments of the space. The additional amount of memory used for MCS-Size result enumeration of 10 000 results varies in between the spaces. For the GalaXi space it achieves its smallest value and for the CHEMriya space the additional amount of memory is largest. This is unexpected, because the REAL Space contains far more fragments and describes a virtual library which is ∼1.7 times larger. KnowledgeSpace requires the second largest amount of memory for enumeration of the results. Regarding MCS-Similarity, the amount of memory required rises about half the additional amount of memory required for MCS-Size. Interestingly the public KnowledgeSpace is an outlier in this relation with the smallest overall increase of required memory.

PAINS Study. PAINS structural alerts[43] are frequently used in medicinal chemistry applications, therefore we were interested in the ability of SpaceMACS to search with the respective patterns. To do so, we used the collection of PAINS[43] patterns from Ochem[44,45] and performed a substructure search requesting up to 100 000 results. 438 of the SMARTS patterns retrieved are written without SMARTS recursion and are usable as queries for SpaceMACS. As further parameters we used a result-storage limit of 100 000, and the number of open-link extensions was set to 100 000 as well. The latter parameter was chosen based on the observation of increasing numbers of results on REAL Space. On the CHEMriya and GalaXi space 10 000 open-link extensions are sufficient.

The results of the PAINS experiment are summarized in Table 5. While the majority of PAINS patterns do not match fragment space compounds, some receive a significant number of hits.

Beginning with the smallest space, GalaXi, we found 29 patterns matching, only 10 of which with significant numbers. Going into detail of the six structural alerts flagging at least 100 000 compounds, three of them "Anil_no_alk", "Catechol_A", and "Pyrrole_A" have at least 100 000 matches in all four spaces. The remaining three alerts are significantly represented in REAL Space and CHEMriya as well. In CHEMriya, 13 structural alerts match at least 100 000 compounds. But below them, none of the remaining alerts matches more than 10 000 molecules. Interestingly more than half of the alerts match at most 100 compounds. Regarding the 10 alerts matching exactly one molecule, nine of them are within the building blocks of the space. The last of those results, the pattern Anil_NH_alk_C matches exactly one compound due to restrictive hydrogen properties.

On REAL Space the situation is similar to the CHEMriya but with more hits in the intermediate ranges. There are 20 structural alerts matching at least 100 000 molecules and further 21 alerts match 10 000 to 100 000 compounds. The most populated bin covers the range of 100 to 10 000 matches with 41 alerts. Regarding those patterns with exactly 1 hit, most of them hit building blocks of the space. Finally, on KnowledgeSpace there are the most alerts matching at least 100 000 molecules. This might relate to the fact that KnowledgeSpace is by far the largest fragment space analyzed. Compared to CHEMriya and REAL Space KnowledgeSpace does not provide dedicated building blocks that could be matched by any structural alert.

The majority of PAINS structural alerts do not produce any hits and can be neglected for compounds extracted from the space. For the remaining ones, it depends on the use case whether postprocessing is required for extracted molecules. Given the shear number of molecules represented in the chemical spaces, PAINS alerts are rather seldom. This study also shows how SpaceMACS can be used to further optimize chemical fragment spaces by avoiding specific unwanted structural elements.

https://doi.org/10.1021/acs.jcim.1c00640
J. Chem. Inf. Model. 2022, 62, 2133−2150

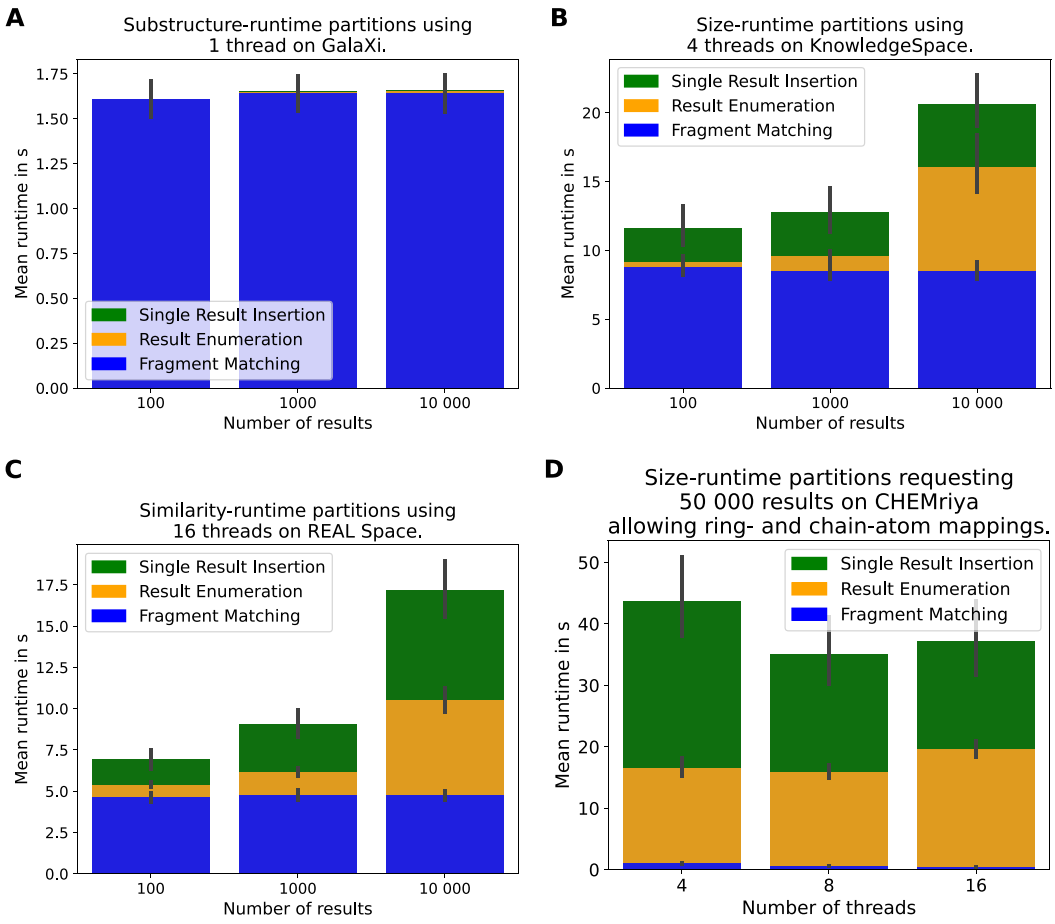


Figure 9. Runtime partitions of the three steps of the SpaceMACS algorithm requesting 100 up to 10 000 results (A−C) and 50 000 results (D). In A, 100, 1000, and 10 000 results are retrived from GalaXi running one thread using substructure mode. Distinction between the algorithm steps is not available for queries without any result. In B, 100, 1000, and 10 000 results are retrieved from KnowledgeSpace running four threads in parallel using MCS-Size. In C, 100, 1000, and 10 000 results are retrieved from REAL Space running 16 threads in parallel using MCS-Similarity. In D, 50 000 results are retrieved from CHEMriya running four, eight, and 16 threads in parallel using MCS-Similarity. Additionally allowing ring- and chain- atom compatibility.

Table 4. Overview of the Memory Used by the SpaceMACS Tool for Each of the Four Spaces[a]

	occupied memory in GB		
space	space preparation	MCS-Size	MCS-Similarity
CHEMriya	0.7	8.5	12.4
GalaXi	0.3	2.5	3.5
KnowledgeSpace	4.3	10.8	11.4
REAL Space	11.1	14.9	16.5

[a]The first column displays the amount of memory required loading and preparing the space. The last to columns show the maximum observed memory requirements searching for 10 000 results for 100 queries using the specified mode.

Table 5. Overview of the Number of Patterns Matching Compounds within the Fragment Spaces[a]

	patterns matching no. compounds					
space	any	≥100 000	[10 000, 100 000)	[100, 10 000)	[2, 100)	1
CHEMriya	61	13	0	4	34	10
GalaXi	29	6	4	13	6	0
KnowledgeSpace	89	52	4	15	18	0
REAL Space	130	20	21	41	29	19

[a]For each space, the total number of pattern matching for any compound is shown. The remaining columns classify the number of matches in bins covering roughly two orders of magnitude of potential hits.

A detailed list of all matching PAINS structural alerts can be found in the Supporting Information (see SI File PAINS-Hits.csv).

A SAR-by-Space Application on Glucosyltransferase (GtfC). A substructure search in fragment spaces of billions of molecules is a useful tool to evolve fragments (e.g., based on a crystallographic fragment screening), search for molecules with bioactive substructures (e.g., to design target focused libraries), finding close neighbors of a hit molecule (e.g., to perform SAR studies), finding analogs with improved ADMET properties or to search for the maximum common substructure match of a designed compound (e.g., AI-based, denovo designed or a medicinal chemist's idea).

https://doi.org/10.1021/acs.jcim.1c00640
J. Chem. Inf. Model. 2022, 62, 2133−2150

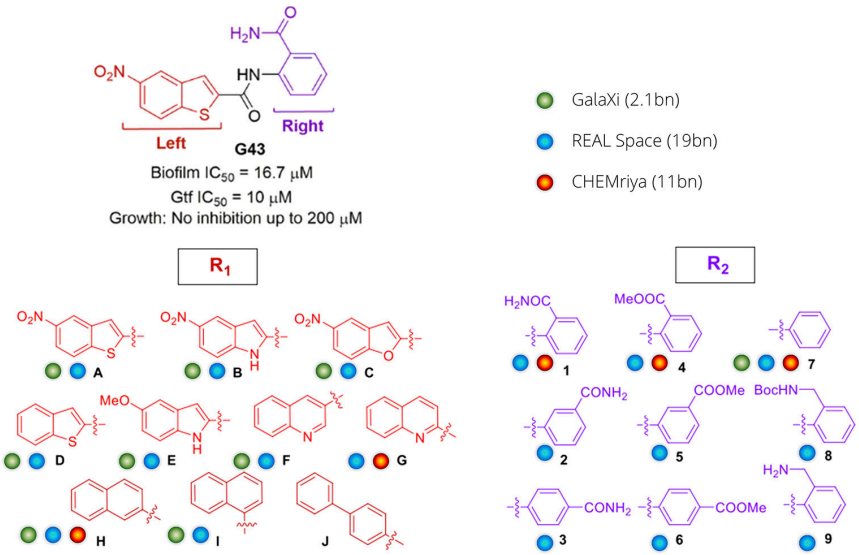


Figure 10. Lead compound G43 used as a query for SpaceMACS (upper left), building blocks R_1 and R_2 used for the traditional SAR study and their occurrence in the commercial fragment spaces indicated by green (GalaXi), blue (REAL Space), and red (CHEMriya) spots (bottom) (Adapted from the work of Nijampatnam et al.[46] Copyright 2020 American Chemical Society).

SAR-by-space as introduced by Klingler et al.[30] is a method to screen fragment spaces for close analogs. Structure−activity relationships can be done as a fast follow-up with commercially available make-on-demand compounds. Here, we applied SpaceMACS on a traditional wet lab SAR approach. To highlight the complementarity of search methods, the so far available algorithms to screen fragment spaces FTrees-FS[22,29] and SpaceLight[16] were applied as well.

The chosen application example deals with the inhibition of glysoyltransferases (GtfBCD) that are responsible for the formation of cariogenic biofilms in dental caries.[46] In a previous work a virtual screening with a library of 500 000 small molecules from the ZINC database[47] on the glucosyltransferase GtfC was performed.[48] Lead compound G43 (see Figure 10) underwent extensive SAR studies by the synthesis and testing of 90 analogs in a follow-up work.[46] These molecules are based on a simple amide formation reaction, for which ten different carboxylic acid building blocks (A−J) were coupled with nine aromatic amine-containing fragments (1−9) (see Figure 10).

We used the compound G43 in SMILES notation as input for SpaceMACS (similarity mode) and screened commercial fragment spaces. These spaces comprise a chemical space of 32 billion make-on-demand molecules. A SpaceMACS search took about 2 min retrieving the first 50 000 from REAL Space (19bn make-on-demand molecules), GalaXi (2.1bn make-on-demand molecules), and CHEMriya (11bn make-on-demand molecules).

Analysis of the occurrence of all 90 compounds synthesized by Nijampatnam et al.[46] in its traditional approach revealed that 50% of the molecules (45/90) could be found within one of the commercial fragment spaces, accessible through an analog amide formation reaction. In addition, we found 90% of the carboxylic acide building blocks (9/10; only J was not found) and 100% of the amine building blocks in at least one of the fragment spaces indicating a plethora of additional combinations.

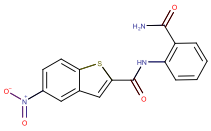
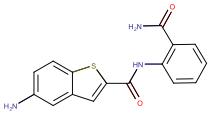
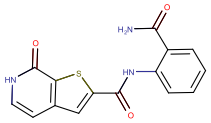
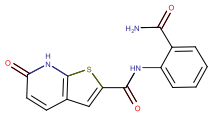
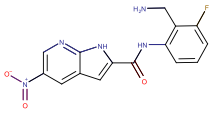
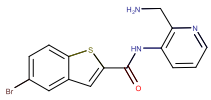
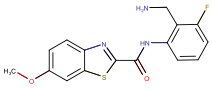
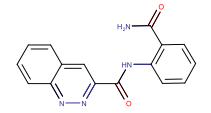
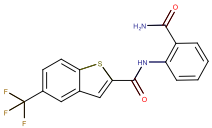
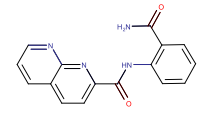
In order to select new molecules the data set of 150 000 molecules was filtered for unwanted substructures, PAINS, and toxic substructures as implemented in BioSolveIT's MedChemWizard workflow.[49] A similarity threshold of 0.64 was set, and compounds lacking the central amide were filtered out. The remaining set of 73 847 molecules was clustered by an atom-based Bermis Murcko scaffold using the RDKit KNIME nodes (version 4.3.0, KNIME GmbH Zurich, knime.org) ending up with 1902 clusters. Only seven of these clusters have been covered by the traditional SAR study.

We aimed to find new molecules in this exhaustive pool of derivatives (see Table 6). As a guideline we used the assumption that the *ortho* amide of the amine building block is mandatory for binding. Also the disadvantages of the nitro group has been discussed by the authors. Simple replacements of the nitro group in lead compound G43 has been found in CHEMriya (trifluor methyl group) or REAL Space (primary amine). Both groups were used as replacement of nitro groups with an increase of activity as summarized on the swissbioisostere server.[50] GalaXi has been found to offer a vast number of molecules similar to amine building block 9 of the traditional SAR approach, albeit the exact match of building block 9 was not found. All three spaces contain different atom based Bermis Murcko alternatives to the lead compound. Docking into the binding site of GtfC as performed by Zhang et al.[48] and Nijampatnam et al.[46] may help to filter out the best candidates.

In order to demonstrate the added value of SpaceMACS, the same query was executed with SpaceLight[16] (fcsfp2.5) and FTrees-FS[22,29] (see Figure 11). SpaceLight is a method to find close analogs with high Tanimoto similarity based on connected subgraph fingerprints, whereas FTrees-FS matches feature trees representing a topological pharmacophore profile. The latter is also known for scaffold hopping and therefore finds molecules with low Tanimoto similarity. The maximum overlap was found between the output of SpaceMACS and SpaceLight (16 241 molecules, 11%), the lowest between

https://doi.org/10.1021/acs.jcim.1c00640
J. Chem. Inf. Model. 2022, 62, 2133−2150

Table 6. Example Set of G43 Derivatives Found by SpaceMACS

Rank	MCS-Similarity	Molecule	Source
1	1		REAL Space
31	0.92		REAL Space
2614	0.77		REAL Space
2615	0.77		REAL Space
9	0.78		GalaXi
43	0.73		GalaXi
328	0.68		GalaXi
103	0.77		CHEMriya
381	0.75		CHEMriya
1852	0.7		CHEMriya

FTrees-FS and SpaceLight (4059 molecules, 2.7%). The overall overlap between all methods was 0.9% (1334 molecules). SpaceMACS is therefore enriching the toolbox to find again different molecules in fragment spaces. Overall, the three search methods complement each other quite well spanning a range from precise chemical matching via chemical to pharmacophore-like similarity. Taking into account a set of commercial fragment spaces is a good choice. Almost no overlap between these spaces has been calculated by Lessel and Lemmen.[41] We were able to confirm this estimation with an

https://doi.org/10.1021/acs.jcim.1c00640
J. Chem. Inf. Model. 2022, 62, 2133−2150

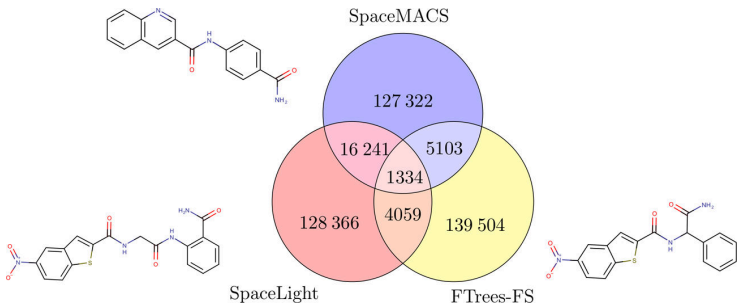


Figure 11. Overlap of the search results between SpaceMACS (blue; similarity mode), SpaceLight (red; fcsfp2.5), and FTrees-FS (yellow). In total 150 000 molecules were written out of three commercial fragment spaces: GalaXi, REAL Space, and CHEMriya. MCS-based similarity (SpaceMACS) enforce exact substructure matching of a core region and ignores the rest of the molecule. Fingerprint-based similarity (SpaceLight) is structure-driven as well but allows variances like repeated patterns. Finally, reduced-graph similarity (FTrees-FS) has the lowest dependency on structure and finds molecules with a similar arrangement of functional groups.

overlap between REAL Space, GalaXi, and CHEMriya of five molecules only (0.01%). The biggest overlap between REAL Space and CHEMriya of 1415 compounds (2.83%) is also a chance to find the same compound with possibly other synthetic routes or prices (see Figure S8).

CONCLUSION

In this paper we presented a novel approach to retrieve valuable compounds from combinatorial chemical spaces too large for enumeration. Our approach, SpaceMACS, transfers the well-known MCS methodology into fragment spaces and virtual make-on-demand accessible compound libraries. Combined with two similarity measures, MCS-Size and MCS-Similarity, SpaceMACS addresses three prominent cheminformatics problems on fragment spaces simultaneously. Given a query compound, the approach enables identity search, substructure search and substructure-based similarity search.

SpaceMACS heavily exploits the fragment-based nature of those spaces allowing calculation of most similar compounds within seconds to minutes running on standard desktop computers. So far, the algorithmic approach comes with two limitations. First, the combinatorial structure of the fragment space has to be tree-like. Although ring formation reactions can be modeled, reactions creating macrocycles from fragments can be not. Such spaces which sometimes appear in DNA-encoded libraries can be handled by the SpaceLight algorithm only. Second, the algorithm remains formally correct only if ring bonds are not allowed to match on acyclic bonds formed in the reactions. We have not seen a single application case were such a matching behavior was required, however this obviously depends on a users preferences. These limitations come with an enormous gain in compute performance making MCS searching in chemical fragment spaces possible for the first time. The performance of SpaceMACS depends on the number of compounds retrieved and the internal structure of the fragment space, but not of the total number of molecules within such libraries. So with respect to the library size SpaceMACS is a sublinear scaling method in all aspects meaning runtimes and memory consumption.

Beyond the algorithm, the outmost potential for performance gains of SpaceMACS are advances in the creation of reaction-based fragment spaces. Encoding multicomponent reactions combining three or more building blocks offers the largest potential to further improve SpaceMACS performance. This fact is observable on KnowledgeSpace representing at least 3 orders of magnitude more compounds than three commercial spaces together without requiring significantly more runtime.

The SpaceMACS tool accepts queries from all standard molecule file formats and SMARTS-like queries extending query possibilities to specify exit vectors in fragment growing scenarios or respecting bioisosteric replacements within a single query. Fragment growing is additionally supported by the possibility to specify desired minimum and maximum heavy atom counts for result compounds and minimum numbers of mapped heavy atoms or minimum similarity values. Several success stories of fragment-based drug discovery can be found in the literature.[51−53] However, a promising fragment needs its follow-up strategy. Here, SpaceMACS is the method of choice to enumerate diverse sets of compounds from combinatorial libraries, having in common the fragments substructure only.

Our case study describes the first step of a fast follow-up on a virtual screening hit. Within seconds and minutes giant combinatorial libraries could be screened without the need of high computational effort. A plethora of analogs have been found that can be synthesized on-demand in a few weeks. SpaceMACS can be seen as a brick at different stages of the whole drug discovery process. At a later point it can be used for bioisosteric replacement or to find analogs that may have better ADME properties. No matter what design method is used by medicinal chemists and modelers, SpaceMACS enables to close the gap to the large make-on-demand compound catalogs emerged in the last years. With very low computational resources, the early phase drug design process can be decoupled from the need of expensive in-house synthesis of potential leads and follow-up compounds. In this way, the algorithm has the potential to substantially shorten the design-make-test cycle of pharmaceutical research.

SOFTWARE AND DATA AVAILABILITY

SpaceMACS is available for Linux, MacOS, and Windows as part of the NAOMI ChemBio Suite at https://uhh.de/naomi and is free for academic use and evaluation purposes. Furthermore, SpaceMACS will soon be integrated into BioSolveIT's infiniSee platform available at https://www.biosolveit.de/infiniSee/. KnowledgeSpace is freely available from BioSolveIT, see https://www.biosolveit.de/infiniSee/#knowledgespace. Several make-on-demand commercial com-

https://doi.org/10.1021/acs.jcim.1c00640
J. Chem. Inf. Model. 2022, 62, 2133−2150

pound spaces, including REAL Space, GalaXi, and CHEMriya, are available to licensed users through BioSolveIT.

■ ASSOCIATED CONTENT

Supporting Information

The Supporting Information is available free of charge at https://pubs.acs.org/doi/10.1021/acs.jcim.1c00640.

Additional manuscript chapters including the following chapters: a more detailed description of the result enumeration procedure, additional plot showing more details of the scaling behavior, additional plots describing scaling limitations requesting 50 000 results from the three commercial fragment spaces, additional information about the external comparison experiment, and a more detailed table showing the selected compounds of our case-study. Zip archive containing the subspaces of KnowledgeSpace used for validation, as *.space and as enumerated *.smiles files. The overview of PAINS-pattern matching any compound in the four fragment spaces (ZIP)

■ AUTHOR INFORMATION

Corresponding Author

Matthias Rarey − *Universität Hamburg, ZBH−Center for Bioinformatics, 20146 Hamburg, Germany;* orcid.org/0000-0002-9553-6531; Phone: +49 (40) 428387351; Email: matthias.rarey@uni-hamburg.de

Authors

Robert Schmidt − *Universität Hamburg, ZBH−Center for Bioinformatics, 20146 Hamburg, Germany;* orcid.org/0000-0002-7089-4842

Raphael Klein − *BioSolveIT GmbH, 53757 Sankt Augustin, Germany;* orcid.org/0000-0002-5087-8730

Complete contact information is available at: https://pubs.acs.org/10.1021/acs.jcim.1c00640

Author Contributions

R.S. and M.R. developed the SpaceMACS concept. R.S. implemented the SpaceMACS approach. R.K. provided the application scenario for SpaceMACS. M.R. supervised the project. All authors participated in manuscript writing.

Notes

The authors declare the following competing financial interest(s): M.R., as a shareholder of BioSolveIT GmbH, declares a potential financial interest in the event that the SpaceMACS software is licensed for a fee to non-academic institutions in the future.

■ REFERENCES

(1) Raymond, J. W.; Willett, P. Maximum common subgraph isomorphism algorithms for the matching of chemical structures. *J. Comput.-Aided Mol. Des.* **2002**, *16*, 521−533.

(2) Ehrlich, H. C.; Rarey, M. Maximum common subgraph isomorphism algorithms and their applications in molecular science: A review. *Wiley Interdiscip. Rev.: Comput. Mol. Sci.* **2011**, *1*, 68−79.

(3) *Daylight Chemical Information Systems Inc.* https://www.daylight.com/dayhtml/doc/theory/theory.smarts.html (accessed on 2021-05-28).

(4) Raymond, J. W.; Gardiner, E. J.; Willett, P. RASCAL: Calculation of graph similarity using maximum common edge subgraphs. *Comput. J.* **2002**, *45*, 631−644.

(5) Schmidt, R.; Krull, F.; Heinzke, A. L.; Rarey, M. Disconnected Maximum Common Substructures under Constraints. *J. Chem. Inf. Model.* **2021**, *61*, 167−178.

(6) Bron, C.; Kerbosch, J. Algorithm 457: finding all cliques of an undirected graph. *Commun. ACM* **1973**, *16*, 575−577.

(7) Stahl, M.; Mauser, H.; Tsui, M.; Taylor, N. R. A robust clustering method for chemical structures. *J. Med. Chem.* **2005**, *48*, 4358−4366.

(8) Cao, Y.; Jiang, T.; Girke, T. A maximum common substructure-based algorithm for searching and predicting drug-like compounds. *Bioinformatics* **2008**, *24*, 366−374.

(9) Koch, I. Enumerating all connected maximal common subgraphs in two graphs. *Theor. Comput. Sci.* **2001**, *250*, 1−30.

(10) Sheridan, R. P.; Miller, M. D. A method for visualizing recurrent topological substructures in sets of active molecules. *J. Chem. Inf. Comput. Sci.* **1998**, *38*, 915−924.

(11) Kawabata, T. Build-up algorithm for atomic correspondence between chemical structures. *J. Chem. Inf. Model.* **2011**, *51*, 1775−1782.

(12) Englert, P.; Kovács, P. Efficient heuristics for maximum common substructure search. *J. Chem. Inf. Model.* **2015**, *55*, 941−955.

(13) Dalke, A.; Hastings, J. FMCS: a novel algorithm for the multiple MCS problem. *J. Cheminf.* **2013**, *5*, O6.

(14) Rogers, D.; Hahn, M. Extended-Connectivity Fingerprints. *J. Chem. Inf. Model.* **2010**, *50*, 742−754.

(15) Cereto-Massagué, A.; Ojeda, M. J.; Valls, C.; Mulero, M.; Garcia-Vallvé, S.; Pujadas, G. Molecular fingerprint similarity search in virtual screening. *Methods* **2015**, *71*, 58−63.

(16) Bellmann, L.; Penner, P.; Rarey, M. Topological Similarity Search in Large Combinatorial Fragment Spaces. *J. Chem. Inf. Model.* **2021**, *61*, 238.

(17) Enamine Ltd. *REALSpace_19bn-2021-04.space.* https://www.biosolveit.de/infiniSee/#realspace (accessed on 2021-05-28).

(18) OTAVA CHEMICALS MB *CHEMriya_11bn-2021-05.space.* https://www.biosolveit.de/infiniSee/#chemriya (accessed on 2021-05-28).

(19) WuXi LabNetwork, GalaXi_2.1bn-2020-11.space. https://www.biosolveit.de/infiniSee/#galaxi (accessed on 2021-05-28).

(20) Lyu, J.; Wang, S.; Balius, T. E.; Singh, I.; Levit, A.; Moroz, Y. S.; O'Meara, M. J.; Che, T.; Algaa, E.; Tolmachova, K.; Tolmachev, A. A.; Shoichet, B. K.; Roth, B. L.; Irwin, J. J. Ultra-large library docking for discovering new chemotypes. *Nature* **2019**, *566*, 224−229.

(21) Irwin, J. J.; Tang, K. G.; Young, J.; Dandarchuluun, C.; Wong, B. R.; Khurelbaatar, M.; Moroz, Y. S.; Mayfield, J.; Sayle, R. A. ZINC20 - A Free Ultralarge-Scale Chemical Database for Ligand Discovery. *J. Chem. Inf. Model.* **2020**, *60*, 6065−6073.

(22) Rarey, M.; Stahl, M. Similarity searching in large combinatorial chemistry spaces. *J. Comput.-Aided Mol. Des.* **2001**, *15*, 497−520.

(23) *Enamine Ltd.* https://enamine.net/ (accessed on 2021-05-28).

(24) *OTAVA CHEMICALS MB.* https://www.otavachemicals.com/ (accessed on 2021-05-28).

(25) *WuXi LabNetwork.* https://www.labnetwork.com/ (accessed on 2021-05-28).

(26) *BioSolveIT GmbH.* https://biosolveit.de (accessed on 2021-05-28).

(27) Bohacek, R. S.; McMartin, C.; Guida, W. C. The art and practice of structure-based drug design: A molecular modeling perspective. *Med. Res. Rev.* **1996**, *16*, 3−50.

(28) Hoffmann, T.; Gastreich, M. The next level in chemical space navigation: going far beyond enumerable compound libraries. *Drug Discovery Today* **2019**, *24*, 1148−1156.

(29) Rarey, M.; Dixon, J. S. Feature trees: A new molecular similarity measure based on tree matching. *J. Comput.-Aided Mol. Des.* **1998**, *12*, 471−490.

(30) Klingler, F. M.; Gastreich, M.; Grygorenko, O. O.; Savych, O.; Borysko, P.; Griniukova, A.; Gubina, K. E.; Lemmen, C.; Moroz, Y. S. SAR by space: Enriching hit sets from the chemical space. *Molecules* **2019**, *24*, 3096.

https://doi.org/10.1021/acs.jcim.1c00640
J. Chem. Inf. Model. 2022, 62, 2133−2150

(31) Detering, C.; Claussen, H.; Gastreich, M.; Lemmen, C. KnowledgeSpace - a publicly available virtual chemistry space. *J. Cheminf.* **2010**, *2*, 53757.
(32) BioSolveIT GmbH *KnowledgeSpace_290tr_2019-05.space*. https://www.biosolveit.de/infiniSee/#knowledgespace (accessed on 2021-05-28).
(33) Urbaczek, S.; Kolodzik, A.; Fischer, J. R.; Lippert, T.; Heuser, S.; Groth, I.; Schulz-Gasch, T.; Rarey, M. NAOMI: On the almost trivial task of reading molecules from different file formats. *J. Chem. Inf. Model.* **2011**, *51*, 3199−3207.
(34) *Drugbank, Nilutamide*. https://go.drugbank.com/drugs/DB00665 (accessed on 2021-05-28).
(35) Levandowsky, M.; Winter, D. Distance between sets [5]. *Nature* **1971**, *234*, 34−35.
(36) Weininger, D. SMILES, a chemical language and information system. 1. Introduction to methodology and encoding rules. *J. Chem. Inf. Model.* **1988**, *28*, 31−36.
(37) Weininger, D.; Weininger, A.; Weininger, J. L. SMILES. 2. Algorithm for Generation of Unique SMILES Notation. *J. Chem. Inf. Comput. Sci.* **1989**, *29*, 97−101.
(38) Schmidt, R.; Ehmki, E. S.; Ohm, F.; Ehrlich, H. C.; Mashychev, A.; Rarey, M. Comparing Molecular Patterns Using the Example of SMARTS: Theory and Algorithms. *J. Chem. Inf. Model.* **2019**, *59*, 2560−2571.
(39) Ehmki, E. S.; Schmidt, R.; Ohm, F.; Rarey, M. Comparing Molecular Patterns Using the Example of SMARTS: Applications and Filter Collection Analysis. *J. Chem. Inf. Model.* **2019**, *59*, 2572−2586.
(40) Lauck, F.; Rarey, M. Customized Enumeration of Chemical Subspaces with Limited Main Memory Consumption. *J. Chem. Inf. Model.* **2016**, *56*, 1641−1653.
(41) Lessel, U.; Lemmen, C. Comparison of Large Chemical Spaces. *ACS Med. Chem. Lett.* **2019**, *10*, 1504−1510.
(42) Landrum, G. A. *RDKit: Open Source Cheminformatics*. https://www.rdkit.org/ (accessed on 2021-05-28).
(43) Baell, J. B.; Holloway, G. A. New substructure filters for removal of pan assay interference compounds (PAINS) from screening libraries and for their exclusion in bioassays. *J. Med. Chem.* **2010**, *53*, 2719−2740.
(44) Tetko, I. V. *OCHEM*. https://ochem.eu/ (accessed on 2021-05-28).
(45) Sushko, I.; Salmina, E.; Potemkin, V. A.; Poda, G.; Tetko, I. V. ToxAlerts: A web server of structural alerts for toxic chemicals and compounds with potential adverse reactions. *J. Chem. Inf. Model.* **2012**, *52*, 2310−2316.
(46) Nijampatnam, B.; Ahirwar, P.; Pukkanasut, P.; Womack, H.; Casals, L.; Zhang, H.; Cai, X.; Michalek, S. M.; Wu, H.; Velu, S. E. Discovery of Potent Inhibitors of Streptococcus mutans Biofilm with Antivirulence Activity. *ACS Med. Chem. Lett.* **2021**, *12*, 48−55.
(47) Sterling, T.; Irwin, J. J. ZINC 15 - Ligand Discovery for Everyone. *J. Chem. Inf. Model.* **2015**, *55*, 2324−2337.
(48) Zhang, Q.; Nijampatnam, B.; Hua, Z.; Nguyen, T.; Zou, J.; Cai, X.; Michalek, S. M.; Velu, S. E.; Wu, H. Structure-Based Discovery of Small Molecule Inhibitors of Cariogenic Virulence. *Sci. Rep.* **2017**, *7*, 1−10.
(49) BioSolveIT GmbH *MedChemWizard*. https://www.biosolveit.de/knime#MedChemWizard (accessed on 2021-05-28).
(50) Wirth, M.; Zoete, V.; Michielin, O.; Sauer, W. H. SwissBioisostere: A database of molecular replacements for ligand design. *Nucleic Acids Res.* **2013**, *41*, 1137−1143.
(51) Mortenson, P. N.; Erlanson, D. A.; De Esch, I. J.; Jahnke, W.; Johnson, C. N. Fragment-to-Lead Medicinal Chemistry Publications in 2017. *J. Med. Chem.* **2019**, *62*, 3857−3872.
(52) Erlanson, D. A.; De Esch, I. J.; Jahnke, W.; Johnson, C. N.; Mortenson, P. N. Fragment-to-lead medicinal chemistry publications in 2018. *J. Med. Chem.* **2020**, *63*, 4430−4444.
(53) Jahnke, W.; Erlanson, D. A.; De Esch, I. J.; Johnson, C. N.; Mortenson, P. N.; Ochi, Y.; Urushima, T. Fragment-to-Lead Medicinal Chemistry Publications in 2019. *J. Med. Chem.* **2020**, *63*, 15494−15507.

https://doi.org/10.1021/acs.jcim.1c00640
J. Chem. Inf. Model. 2022, 62, 2133−2150

Supporting Information: Maximum Common Substructure Searching in Combinatorial Make-on-Demand Compound Spaces

Robert Schmidt,[†] Raphael Klein,[‡] and Matthias Rarey*,[†]

†*Universität Hamburg, ZBH - Center for Bioinformatics, Bundesstraße 43, 20146 Hamburg, Germany*

‡*BioSolveIT GmbH, An der Ziegelei 79, 53757 Sankt Augustin, Germany*

E-mail: rarey@zbh.uni-hamburg.de

Phone: +49 (40) 428387351

1 Detailed Result-Enumeration Scheme

The enumeration of the results follows simple ideas but has to handle various special cases. The main goal is the correct enumeration of the top scored results. Beyond that, runtime and the total number of results considered are other targets for optimization.

The result enumeration is not focused on fragments especially, but on the acyclic bonds in the query and pairs of compatible fragment space linker types resulting in a enumeration process based on pairs of compatible *bond-linktype keys*, meaning their linker types are compatible and they represent the same bond in opposite directions.

During matching, results are stored based on their *bond-linktype key*. The results to be

considered for a *bond-linktype key* usually comprise several fragments and even several (different) MCS mappings within a single fragment. All of them have in common that they map a linker with linker type of the *bond-linktype key* to a distinct atom within the query. Respecting this, each extension of partial results has to deal with many different branches in search space.

The enumeration process employs four important datastructures. Those are three types of nodes, representing the final fragment tree and an result overview handler. All node types know their parent ResultNode including the linker of the fragment they are extending in their parent node and the previous node of their kind.

ResultNode ResultNodes represent the actual selection of fragments and the connections between them building the final target compound. ResultNodes store actual size information about the current result and the size of the final compound. For the MCS-Similarity based enumeration, the estimation of the best possible similarity value respecting any further extension is also crucial. Finally ResultNodes track a result *id* enabling efficient branch and bound procedures during enumeration.

MatchedLinkNode MatchedLink nodes represent future extensions of matched links on result nodes. Each extension is represented by a *bond-linktype key* and a delta value compared to the final MCS-Size of the best result.

OpenLinkNode OpenLinkNodes are used to saturate fragments of the ResultNodes. They only store the saturated linker.

OverviewHandler The OverviewHandler is responsible for the progress of the branch and bound procedure. It stores the final size for MCS-Size based enumerations of the MCS-Similarity estimations for MCS-Similarity based result enumerations for the results based on their *id*. The overview handler is initialized with the target number of results enumerated. During enumeration the handler differentiates between results enumerated and stored already and results actually in enumeration extending the cur-

rent *bond-linktype key*. In total, each result is assigned an absolute rank for the current state of the enumeration.

The matching process has to consider acyclic bonds in both directions. For correct enumeration this is no longer necessary, since results are extended over all their linkers. Due to this, only *bond-linktype keys* with increasing internal atom positions of the affected atoms are considered on the outmost result enumeration. Nonetheless all *bond-linktype keys* are necessary for correct result extensions.

Result Enumeration for a Distinct *bond-linktype key*

First of all, the best result is assigned a ResultNode and the list of current results of the OverviewHandler is cleared. All alternative results within the current fragment, represented by the ResultNode and all results of other fragments sharing the same *bond-linktype key* are enumerated. This results in a list of *sibling* results. As they represent the initial fragments, their initial link has not been considered for any extension so far. Based on compatible linker types of the fragment space, all possible MatchedLinks for the initial link are enumerated. Then, regular extensions of the matched links for the result including their alternatives are enumerated. Each combination of results including the initial link is sorted by descending MCS-Size of the extension. Finally OpenLinkNodes are added saturating the current ResultNode. This enters the main recursion of the enumeration process handling the first ResultNode a MatchedLinkNode and an optional OpenLinkNode. If there are several MatchedLinkNodes or OpenLinkNodes, they are accessible based on the previous relation within all nodes.

The Main Recursion

Based on the result position of the current ResultNode, the OverviewHandler aborts the current result extension. As long as there is a MatchedLinkNode to extend the result, it is

processed.

MatchedLinkNodeExtension:

Extending a matched link adds a new ResultNode with updated current size and final size values, based on the delta stored in the MatchedLinkNode updating the final size or estimation within the OverviewHandler. The extension determines whether and which result continues the *id* of the current ResultNode. Then all *sibling* results of the new ResultNode are enumerated and extended with all combinations of further MatchedLinkNodes and final saturation of OpenLinkNodes. This continues the main recursion. After completion of the recursion, each MatchedLinkNode is replaced by an OpenLinkNode catching all remaining fragment extension.

OpenLinkNode extension:

OpenLinkNodes are extended, whenever there is no MatchedLinkNode any more. Based on the *termination suggestion* all alternative fragment extensions are enumerated. The first fragment of the *termination suggestion* usually continues the *id* of the current result. The remaining alternatives of the list create new ResultNodes with updated current and final size values. The estimation for MCS-Similarity enumeration is updated using the same delta values as for the final size update. If the fragment, selected for extension has more than one linker, remaining OpenLinkNodes are added. Each of the enumerated results continues the main recursion.

Result Building: When there are no other nodes, the (implicit) list of ResultNodes represents a fragment tree. Nodes are canonized based on internal ids of the fragments and positions of the links connecting the fragments. Afterwards they are stored.

Updating Final Sizes

Unfortunately final result sizes vary significantly during all stages of the result enumeration. This is most obvious for the MCS-Similarity enumeration assigning upper bound MCS-Similarity values to the ResultNodes. More subtle, this might happen during MCS-Size

enumeration as well. Fortunately this happens only on expansion of MatchedLinkNodes or during sibling enumeration of ResultNodes such that no actual update of the estimated score is needed during MCS-Size enumeration. To accommodate for those updates, results are added to the overview handler as late as possible for MCS-Similarity, whereas results are added as early as possible for MCS-Size enumerations. This way, almost no result gets lost due to too optimistic MCS-Similarity or MCS-Size estimations.

2 Scaling Behavior

As an extension to the selection of runtime plots in Figure 8 we provide the full scaling behavior of SpaceMACS retrieving 1000 and 10 000 results from the four fragment spaces analyzed. The analysis is provided in the Figures S1-S4. Each figure is focused on a specific fragment space. As in Figure 8 performance is measured using the 100 queries from Lessel and Lemmen.[S1]

In Figure S1 scaling behavior for CHEMriya is shown. Searching for up to 1000 analogs of the queries, runtimes are below 10s seconds on average using 4 to 8 threads, depending on the ring- and chain-atom compatibility parameter. Retrieving 10 000 results, runtimes for MCS-Size are as low as 10s using 16 threads. On the other hand, enumeration and synchronization of the results substantially influence the performance. The results show that using 16 or more threads in parallel does not gain much performance. The relatively small number of fragments of the space suggest an upper limit of 8 threads for SpaceMACS searches of the 11 billion compounds represented in CHEMriya.

Figure S2 is focused on GalaXi. On GalaXi, there is almost no difference observable between the ring- and chain-atom compatibility settings. Enumeration of 1000 results takes less than 2s on average using 8 threads and might even be as low as 1s. Enumeration on 10 000 results is achieved within 10s using 4 threads. Since the GalaXi space is relatively small, 4 to 8 threads are a reasonable upper limit to be used on this space.

Figure S3 handles the KnowledgeSpace. As discussed in the manuscript, the standard deviation for MCS-Similarity is relatively high. The influence of extreme outliers becomes more serious when requesting 10 000 results. Compared to the commercial spaces, KnowledgeSpace contains far more products containing more than two fragments. But even including those outliers, average runtimes are within at most minutes retrieving most similar compounds from over 290 trillion compounds of the KnowledgeSpace. KnowledgeSpace represents almost four orders of magnitude more compounds than the tree commercial spaces combined.

Finally Figure S4 shows the runtime performance on the REAL Space. Searching for 1000 analogs, runtimes are as small as 10s on average using 32 threads. In contrast to the other spaces, using 16 threads results in an acceptable scaling. In comparison to the other spaces, the REAL Space contains by far the most fragments. This seems to be one of the reasons for the best scaling behavior observed on the fragment spaces analyzed.

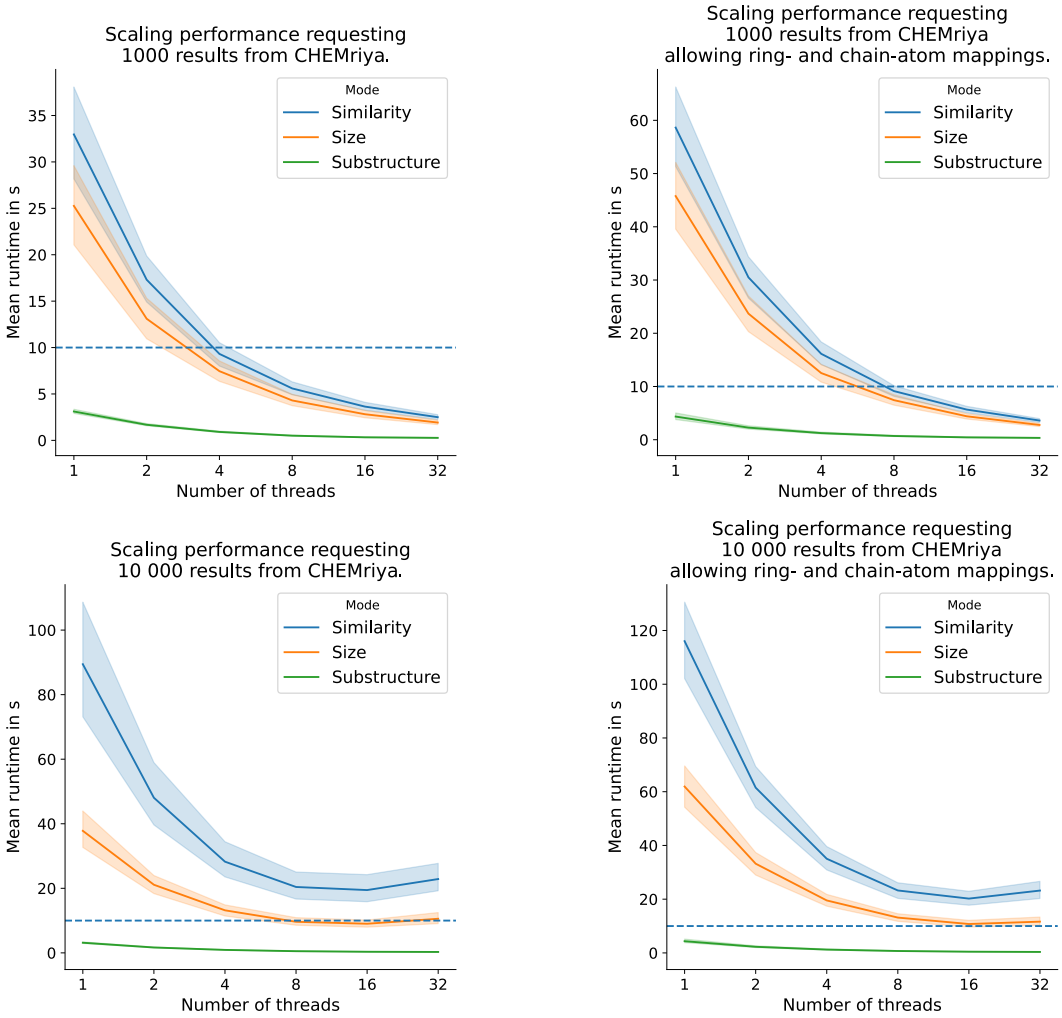


Figure S1: SpaceMACS scaling behavior using up to 32 threads retrieving 1000 and 10 000 results from CHEMriya. Runtime of 10s is marked for reference. In the plots on the left, ring- and chain-atom mapped onto each other. On the right side, such mappings are allowed resulting in slightly different runtimes.

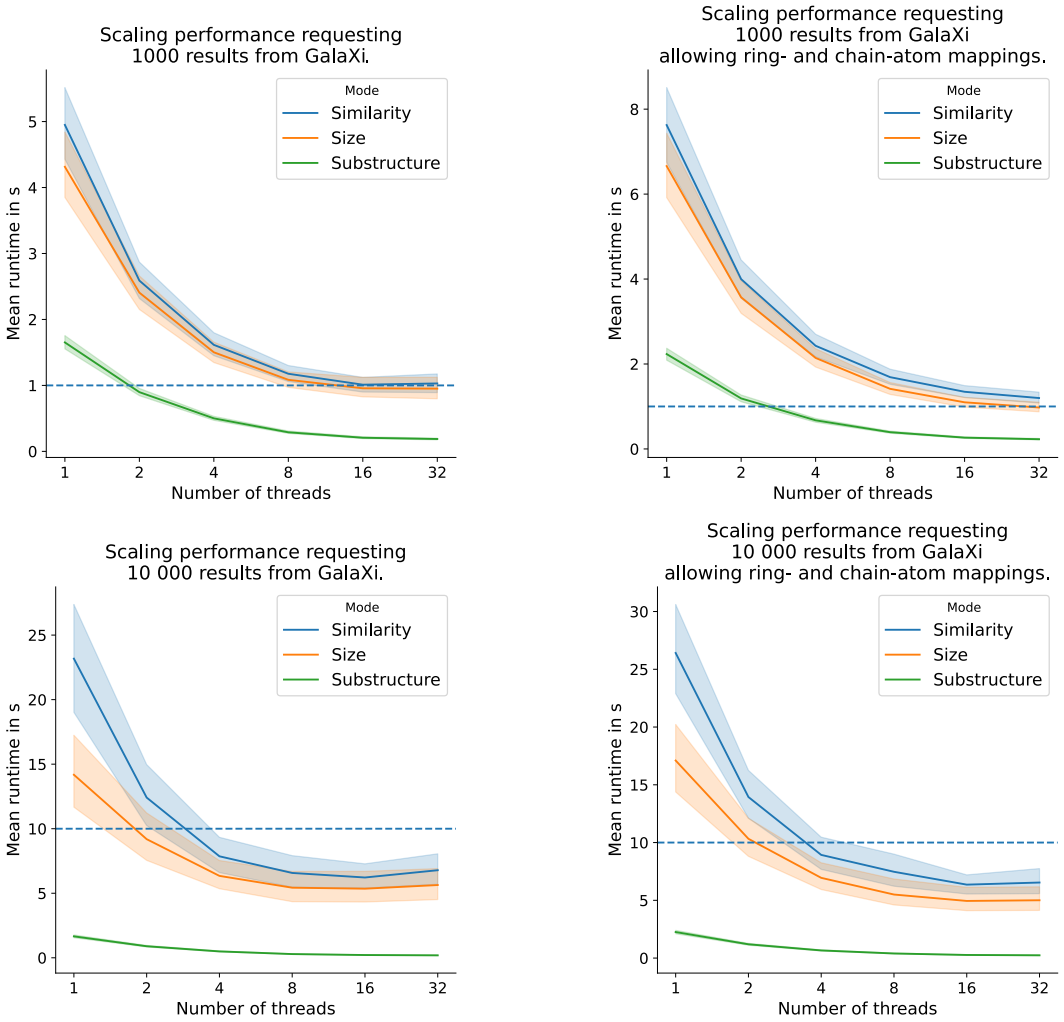


Figure S2: SpaceMACS scaling behavior using up to 32 threads retrieving 1000 and 10 000 results from GalaXi. Runtimes of 1s (top row) and 10s (bottom row) are marked for reference. In the plots on the left, ring- and chain-atom mapped onto each other. On the right side, such mappings are allowed resulting in slightly different runtimes.

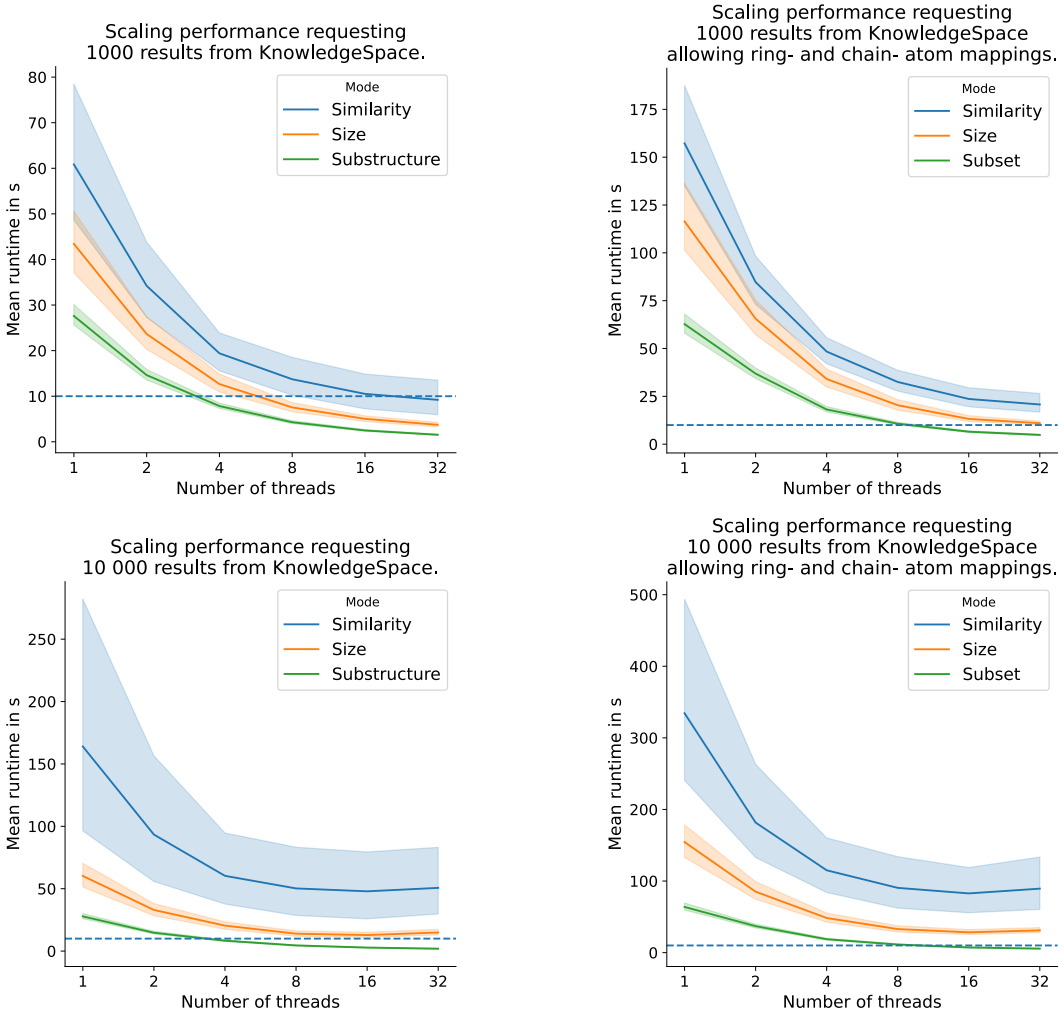


Figure S3: SpaceMACS scaling behavior using up to 32 threads retrieving 1000 and 10 000 results from KnowledgeSpace. Runtime of 10s is marked for reference. In the plots on the left, ring- and chain-atom mapped onto each other. On the right side, such mappings are allowed resulting in slightly different runtimes.

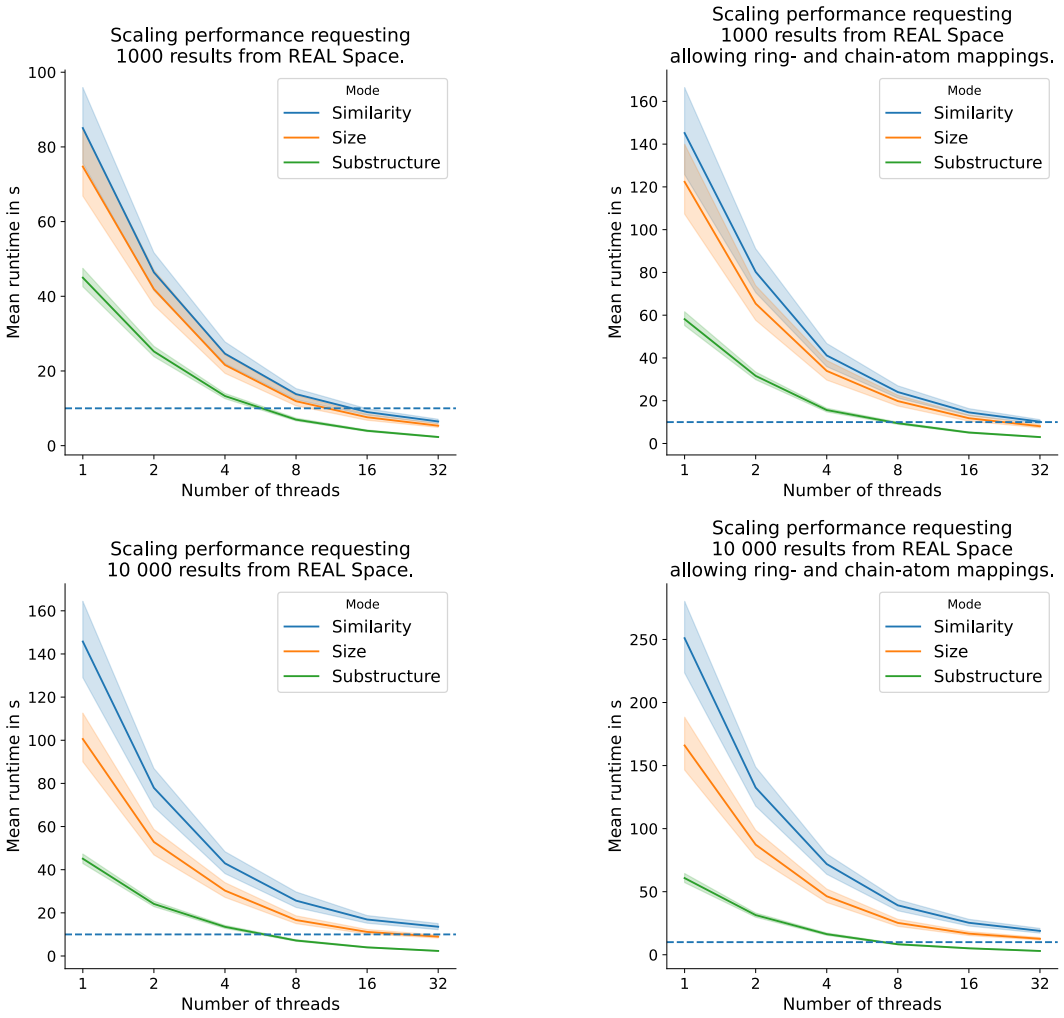


Figure S4: SpaceMACS scaling behavior using up to 32 threads retrieving 1000 and 10 000 results from REAL Space. Runtime of 10s is marked for reference. In the plots on the left, ring- and chain-atom mapped onto each other. On the right side, such mappings are allowed resulting in slightly different runtimes.

3 Scaling limits

Additional to the runtime partition plots in Figure 9 we provide the full overview about the scaling behavior requesting 50 000 results on the three commercial spaces. Figure S5 focuses on MCS-Size, whereas Figure S6 shows the same plots, but for MCS-Similarity. Two of the three spaces analyzed contain only a relatively small number of fragments, meaning the CHEMriya[S2,S3] and the GalaXi.[S4,S5] On both spaces, runtimes are completely dominated by result building or in-fragment MCS-result computation. Only for the third space, the REAL Space[S6,S7] matching accounts significantly for the overall runtimes. According to the scaling behavior, on CHEMriya there is no gain in performance using more than eight threads. On the GalaXi, runtimes slightly decrease using 16 instead of eight threads, but the gain in performance is far from any linear scaling. Regarding the number of results requested, the REAL Space shows the best scaling behavior. Here, using 16 instead of four threads decreases runtimes by a factor of 2 whereas the number of threads is increased by a factor of 4. Even if this behavior might be acceptable, it clearly shows the scaling limitations on all available commercial spaces analyzed.

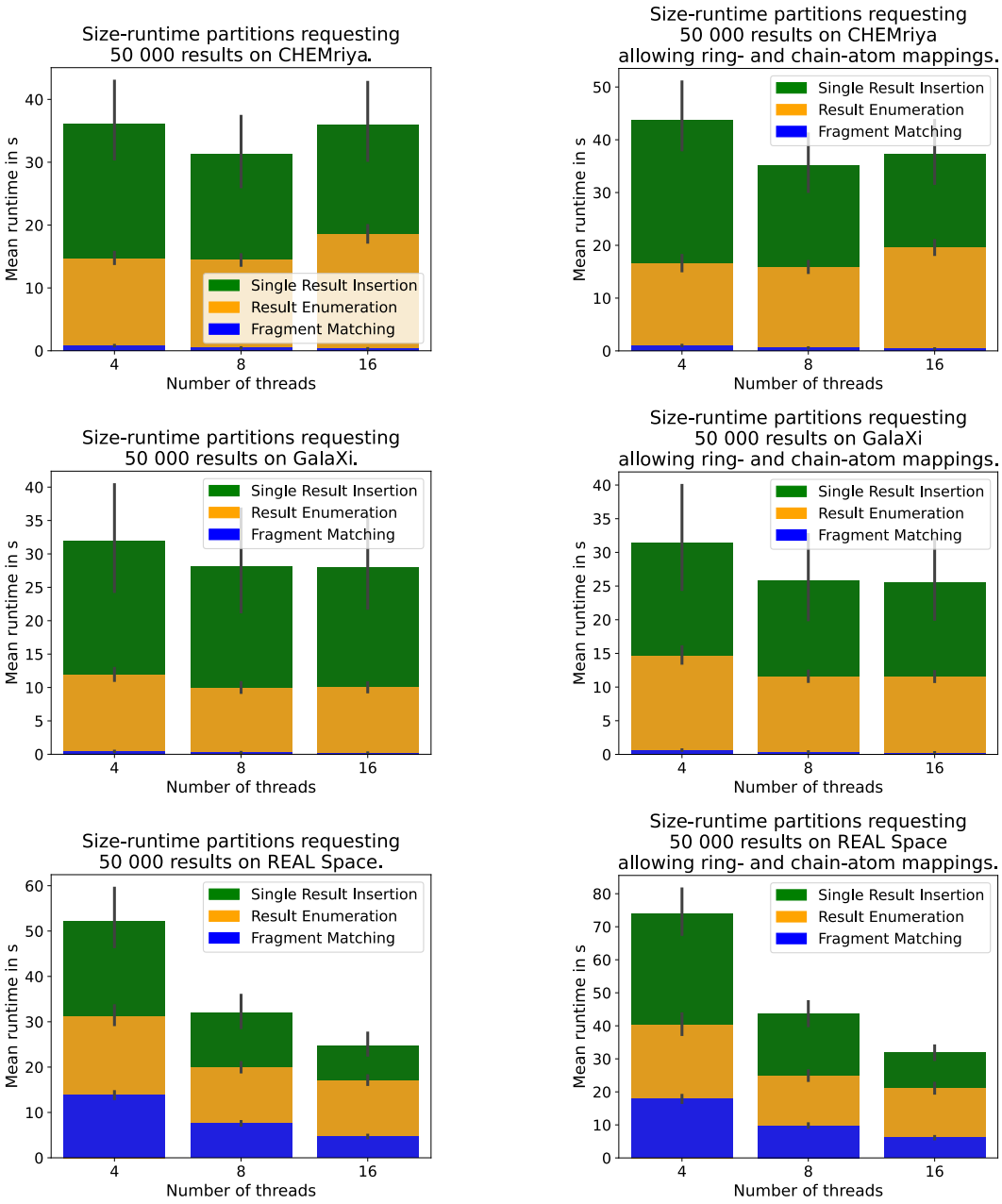


Figure S5: Runtime partitions and scaling behavior requesting 50 000 results on the three commercial spaces using MCS-Size. Each subplot shows the runtime partitions using four, eight and 16 threads in parallel. In the plots on the left, ring- and chain- atoms are not mapped onto each other. On the right side, such mappings are allowed resulting in slightly different runtimes.

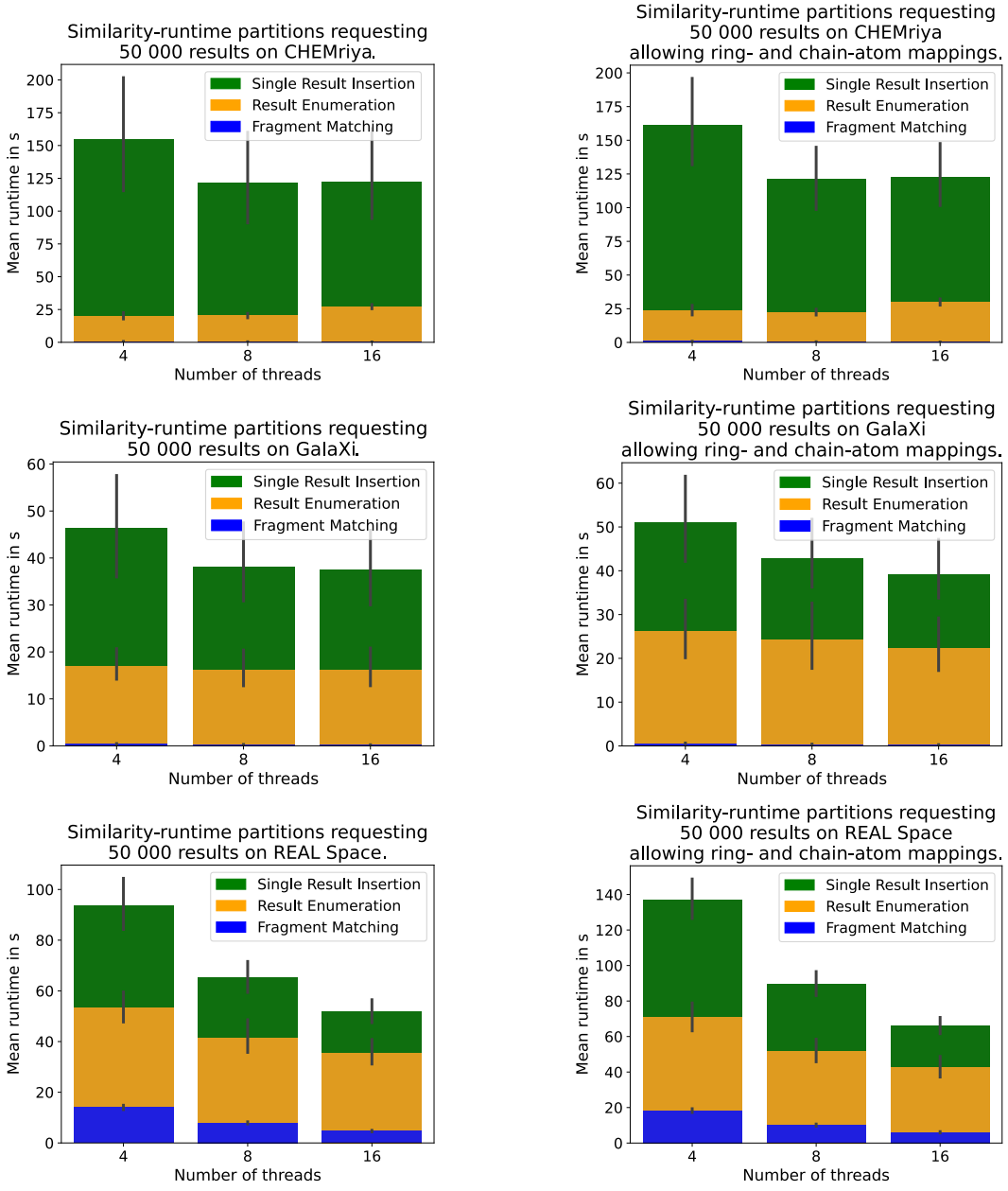


Figure S6: Runtime partitions and scaling behavior requesting 50 000 results on the three commercial spaces using MCS-Similarity. Each subplot shows the runtime partitions using four, eight and 16 threads in parallel. In the plots on the left, ring- and chain-atoms are not mapped onto each other. On the right side, such mappings are allowed, resulting in slightly different runtimes.

4 External Comparison

Two example queries from the comparison of SpaceMACS on the 500 fragments representation and RDKit on the enumerated molecules. Query Q81 was chosen because all results except the topmost 5 are identical and query Q60 was selected for the smallest overlap measured of four compounds.

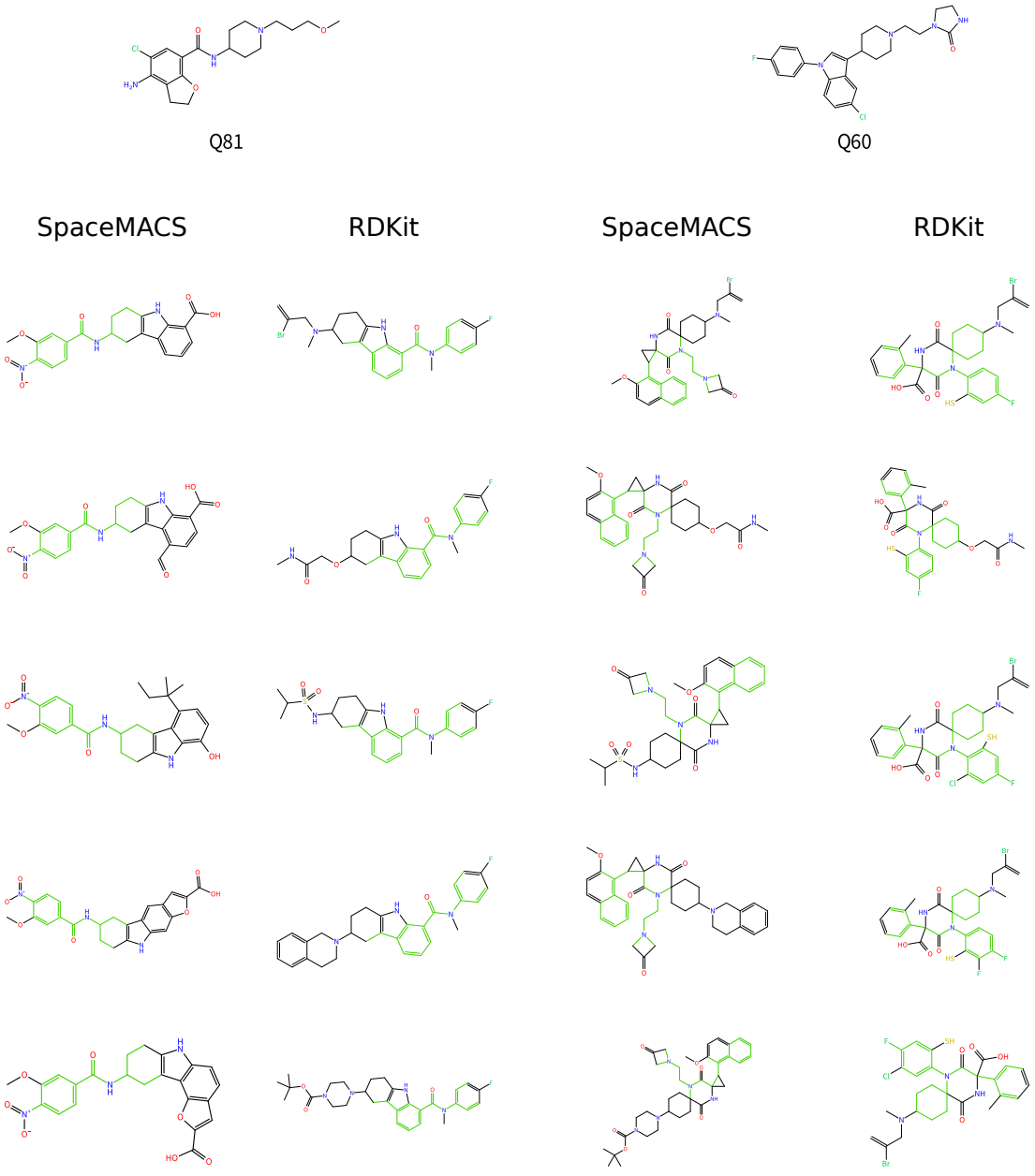
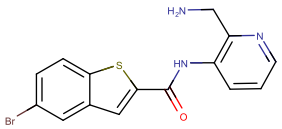
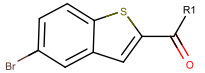
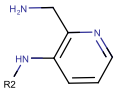
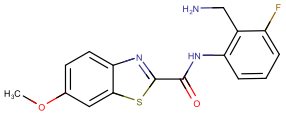
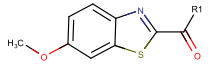
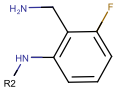
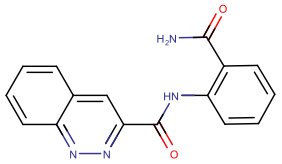
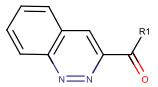
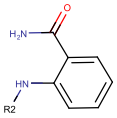
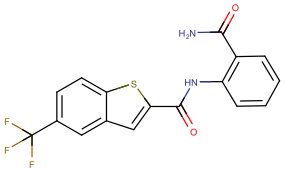
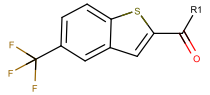
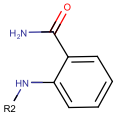
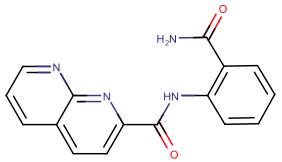
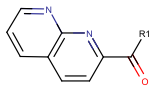


Figure S7: The top 5 results for the queries Q81 and Q60 on the 500 fragments space calculated by SpaceMACS and RDKit. Within each of the results, the calculated MCS is highlighted in green.

Table S1: **Selection of G43 derivatives found by SpaceMACS.**

Rank	MCS-Similarity	MCS-Size	Molecule	Source	Compound ID	Building Block 1	Building Block 2
1	1	24/24		REAL Space	m_22bca__138522__138468		
31	0.92	22 / 22		REAL Space	m_240690bcc__7350998__13902326		
2614	0.77	20 / 22		REAL Space	m_22bca__138522__11620124		
2615	0.77	20 / 22		REAL Space	m_22bca__138522__15414758		
9	0.78	21 / 24		GalaXi	WXVL001__AK0951__SE1311		

Table S1: continued from previous page

Rank	MCS-Similarity	MCS-Size	Molecule	Source	Compound ID	Building Block 1	Building Block 2
43	0.73	19 / 21		GalaXi	WXVL001__AK4061__SE0385		
328	0.68	19 / 23		GalaXi	WXVL001__AK0951__SE1052		
103	0.77	20 / 22		CHEMriya	YASR001__OTV1780792__OTV0125330235		
381	0.75	21 / 25		CHEMriya	YASR001__OTV7131291__OTV0125330235		
1852	0.7	19 / 22		CHEMriya	YASR001__OTV7017520065__OTV125330235		

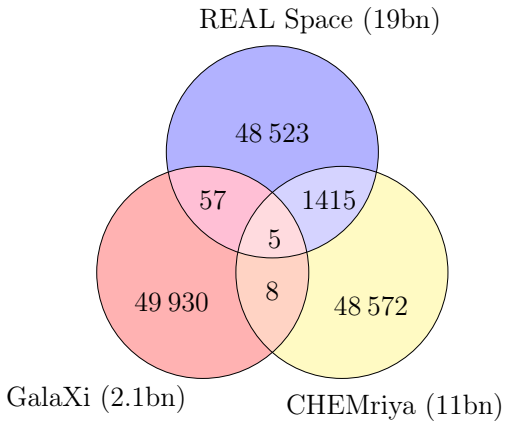


Figure S8: Overlap of the search results between REAL Space (blue), GalaXi (red) and CHEMriya (yellow). 50 000 molecules were written out of each of the three commercial fragment spaces.

References

(S1) Lessel, U.; Lemmen, C. Comparison of Large Chemical Spaces. *ACS Med. Chem. Lett.* **2019**, *10*, 1504–1510.

(S2) OTAVA CHEMICALS MB, CHEMriya_11bn_2021-05.space. `https://www.biosolveit.de/infiniSee/#chemriya`, (accessed on: 2021-05-28).

(S3) OTAVA CHEMICALS MB. `https://www.otavachemicals.com/`, (accessed on: 2021-05-28).

(S4) WuXi LabNetwork, GalaXi_2.1bn_2020-11.space. `https://www.biosolveit.de/infiniSee/#galaxi`, (accessed on: 2021-05-28).

(S5) WuXi LabNetwork. `https://www.labnetwork.com/`, (accessed on: 2021-05-28).

(S6) Enamine Ltd., REALSpace_19bn_2021-04.space. 2021; `https://www.biosolveit.de/infiniSee/#realspace`, (accessed on: 2021-05-28).

(S7) Enamine Ltd. `https://enamine.net/`, (accessed on: 2021-05-28).

Comparing Molecular Patterns Using the Example of SMARTS: Theory and Algorithms

[D3] Schmidt, Robert u. a.: „Comparing Molecular Patterns Using the Example of SMARTS: Theory and Algorithms“. In: *J. Chem. Inf. Model.* 59.6 (2019), S. 2560–2571. DOI: 10.1021/acs.jcim.9b00250

http://pubs.acs.org/articlesonrequest/AOR-sXvxJXHkFzQuctjywxaz

JCIM JOURNAL OF CHEMICAL INFORMATION AND MODELING
Cite This: J. Chem. Inf. Model. 2019, 59, 2560−2571
Article
pubs.acs.org/jcim

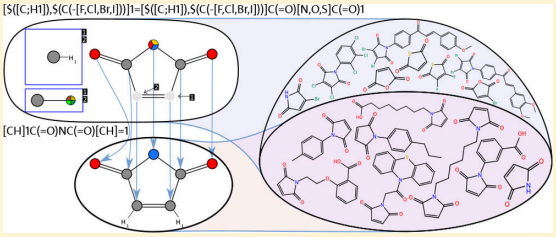
Comparing Molecular Patterns Using the Example of SMARTS: Theory and Algorithms

Robert Schmidt, Emanuel S. R. Ehmki, Farina Ohm,† Hans-Christian Ehrlich,‡ Andriy Mashychev,§ and Matthias Rarey*

ZBH - Center for Bioinformatics, Bundesstraße 43, 20146 Hamburg, Germany

Supporting Information

ABSTRACT: Molecular patterns are widely used for compound filtering in molecular design endeavors. They describe structural properties that are connected with unwanted physical or chemical properties like reactivity or toxicity. With filter sets comprising hundreds of structural filters, an analytic approach to compare those patterns is needed. Here we present a novel approach to solve the generic pattern comparison problem. We introduce chemically inspired fingerprints for pattern nodes and edges to derive an easy-to-compare pattern representation. On two annotated pattern graphs we apply a maximum common subgraph algorithm enabling the calculation of pattern inclusion and similarity. The resulting algorithm can be used in many different ways. We can automatically derive pattern hierarchies or search in large pattern collections for more general or more specific patterns. To the best of our knowledge, the presented algorithm is the first of its kind enabling these types of chemical pattern analytics. Our new tool named SMARTScompare is an implementation of the approach for the SMARTS language, which is the quasi-standard for structural filters. We demonstrate the capabilities of SMARTScompare on a large collection of SMARTS patterns from real applications.



INTRODUCTION

Chemical patterns are widely used during filtering steps of drug design endeavors. They are associated with the common task of substructure searching for filtering and tagging. There are a few published chemical pattern languages, such as SMARTS,[1] SLN,[2] and Molecular Query Language.[3] All of those languages have in common that they describe molecular substructures with many features for atoms and bonds. Several collections comprising hundreds of patterns have been published for different languages.[4−12] The primary focus of these collections is compound identification in a high-throughput manner.[5,8−12]

When using chemical patterns to filter molecules, the focus lies more on matching the right class of chemistry than identifying the specific atoms that are matched by pattern nodes. Patterns are furthermore frequently used to match functional groups[13−15] and identify and classify rotatable bonds[16,17] or single atoms with specific chemical environments.[7] In this context, the set of atoms that are matched by the pattern are of specific interest. Their environment can be important for the pattern match, although it is not part of the expected result.

For the SMARTS language, a series of tools have been developed for visualization[18,19] and editing.[20] Furthermore, some existing methods are able to automatically construct molecular patterns separating two compound sets,[21,22] the latter automatically creating SMARTS expressions. However, beyond visual inspection, there is no validated algorithmic approach available to date to compare two SMARTS patterns. To improve, compare, or extend pattern collections in a meaningful way, a pattern comparison method is highly desirable.

From a purely semantic point of view, molecular patterns are similar to regular expressions. Regular expressions are used extensively for text search in computer science. In theoretical computer science, inclusion and equality of regular expressions are computed, transforming them into equivalent deterministic final automatons (DFAs) and testing whether their accepted languages are inclusive or identical. However, there might be an exponential blowup while constructing the DFA.[23] For subsets of regular languages, there are polynomial time DFA constructions[24] and polynomial time comparison approaches for direct regular expression comparisons.[25] However, the structural differences between regular expressions and chemical patterns are too large to transfer these findings.

In cheminformatics, the RDKit[26] library has some functionality implemented to compare two molecular patterns in the form of SMARTS. Subset relations of simple patterns that are written as pure conjunctions without any negation can be determined if no property transfer is needed. For the more general pattern comparison problem, allowing disjunctions or negation of simple properties, the results do not guarantee to find more specific patterns anymore (for more details, see section SI6 in the Supporting Information).

Received: March 22, 2019
Published: May 23, 2019

DOI: 10.1021/acs.jcim.9b00250
J. Chem. Inf. Model. 2019, 59, 2560−2571

To be able to conduct comprehensive analysis of molecular patterns, we have developed a novel analytic approach. In the following, we will model the chemical pattern comparison problem and present an outline of our algorithm. Several technical details of the method can be found in the Supporting Information. Finally, we validate the detection of more specific patterns and conclude with an outline of upcoming applications for the SMARTScompare approach.[27]

METHODS

The Chemical Pattern Comparison Problem. In this section, an abstract chemical pattern is defined. This follows the goal to provide definitions for a generic pattern comparison algorithm that is applicable to several pattern languages. On the basis of the concept of abstract chemical patterns, we can define relevant terms like pattern isomorphism, pattern inclusion, and pattern similarity. The following definitions focus on the chemical space of existing molecules (CS) and do not consider differences introduced by different implementations of pattern languages or chemistry models.

For simplicity, we decided to focus our theoretical approach on the problem of identifying molecule sets matched by a certain pattern. Describing the individual atoms matched in a molecule by a pattern will be considered later when the theory is mapped to the SMARTS language.

Isomorphic Patterns. A chemical pattern P, such as a SMARTS expression, can be understood as a subset $CS(P)$ of the space of all theoretically existing molecules, CS. Some of the molecules from CS match the pattern and therefore are part of the subset, while others do not and are not. The subset $CS(P)$ is infinite in size for many patterns, but nevertheless, it provides a first glimpse of the notion of pattern equality: we consider two patterns P_1 and P_2 to be equal if and only if $CS(P_1) = CS(P_2)$. It becomes immediately clear that if $CS(P_1) = CS(P_2)$, then P_1 and P_2 have to match the same set of atoms in each molecule of $CS(P_1)$.

Since two patterns can match different parts of a molecule, further relationships like mutual exclusion are not particularly useful if defined this way. To capture the chemical intuition of a term saying "two patterns have something or nothing in common", we have to take the mappings between the nodes of the patterns and the atoms within the matching molecules into account. Since a pattern node always matches a single atom in a molecule, we can associate node n with the subset $AS(n)$ of all matching atomtypes from a whole atomtype space, AS. As above, we can define the equality relationship for a pair of pattern nodes n_1 and n_2. It should be noted that in contrast to CS, the space AS is of finite size, such that the node relationships can be computed easily. Once equality between nodes is defined, many concepts from graph theory, most importantly isomorphism and subgraph isomorphism, can be transferred to chemical patterns. For completeness, we can consider bondtypes analogously. For a pattern edge e, $BS(e)$ is the subset of all matching bondtypes from the whole bondtype space, BS. Two pattern edges e_1 and e_2 are considered equal if and only if $BS(e_1) = BS(e_2)$.

In summary, we suggest that pattern isomorphism be defined as follows: Given two chemical patterns $P_1 = (N_1, E_1)$ and $P_2 = (N_2, E_2)$, where the N_i are the sets of nodes and the E_i are the sets of edges between nodes, P_1 and P_2 are isomorphic if and only if there exists a bijective node-mapping function f: $N_1 \rightarrow N_2$ such that

- $\forall\, n \in N_1$: $AS(n) = AS(f(n))$
- $\forall\, m, n \in N_1$: $\{m, n\} \in E_1 \Leftrightarrow \{f(m), f(n)\} \in E_2 \wedge BS(\{m, n\}) = BS(\{f(m), f(n)\})$

Furthermore, we need the following restrictions:

1. The atomtype space AS and bondtype space BS cannot contain any chemically infeasible or redundant states.
2. Some properties (e.g., aromaticity) have a strong influence on the environment of a node or an edge, e.g., if a pattern describes exactly two generic atoms that are part of a generic aromatic ring, then its second aromatic node does not provide any information to the pattern. Because of this, a generic chemical pattern must be nonredundant in several ways:
   (a) Each node of a chemical pattern must be relevant for its representation in the chemical space CS.
   (b) Chemical patterns must be nonredundant in the sense that for each node n, its whole subset of atomtype space, $AS(n)$, is necessary to describe $CS(P)$. The same must hold true for all of the pattern edges e and their subsets of the bondtype space, $BS(e)$. Otherwise different patterns P_1 and P_2 ($P_1 \neq P_2$) could describe the same subset of chemical space, i.e., $CS(P_1) = CS(P_2)$. This implies that for each atomtype and bondtype that is possible for any pattern node or edge, there exists a molecule $m \in CS(P)$ such that $m \notin CS(P')$ if a certain atomtype or bondtype is not present at the node or edge:

$$P = (N_1, E_1) \tag{1}$$

$$\forall\, n \in N_1\colon \forall\, s_a \in AS(n)\colon \exists\, m \in CS(P)\colon \exists\, P'\colon P' = (N_1 \backslash \{n\} \cup \{n \backslash \{s_a\}\}, E_1) \Rightarrow m \notin CS(P') \tag{2}$$

$$\forall\, \{m, n\} \in E_1\colon \forall\, s_e \in BS(\{m, n\})\colon \exists\, m \in CS(P)\colon \exists\, P'\colon P' = (N_1, E_1 \backslash \{\{m, n\}\} \cup \{\{m, n\} \backslash \{s_e\}\}) \Rightarrow m \notin CS(P') \tag{3}$$

The Subpattern Relationship. From the application point of view, it would be very interesting to detect subset relationships. In practice, a subset relationship $P_1 \subseteq P_2$ means that P_2 is a more generic pattern than P_1, that is, P_2 matches all molecules that P_1 hits and potentially some more (in which case P_1 is a true subset). For a pair of pattern nodes n_1 and n_2, we define $n_1 \subseteq n_2$ if and only if $AS(n_1) \subseteq AS(n_2)$. Following the definition of subgraph isomorphism, we are able to define the subset relationship between chemical patterns as follows: We consider a chemical pattern $P_1 = (N_1, E_1)$ to be a subpattern of $P_2 = (N_2, E_2)$ if and only if there exists a subset $N_1' \subseteq N_1$ and a bijective node-mapping function f: $N_2 \rightarrow N_1'$ such that

- $\forall\, n \in N_2$: $AS(f(n)) \subseteq AS(n)$
- $\forall\, n_1, n_2 \in N_2$: $\{n_1, n_2\} \in E_2 \Leftrightarrow \{f(n_1), f(n_2)\} \in E_1 \wedge BS(\{f(n_1), f(n_2)\}) \subseteq BS(\{n_1, n_2\})$

Every node and edge of the more generic pattern P_2 is mapped to a node of the more specific pattern P_1. Each single mapping follows the subset relation as well. It should be noted that the intuitive relationship between pattern isomorphism and

DOI: 10.1021/acs.jcim.9b00250
J. Chem. Inf. Model. 2019, 59, 2560−2571

subpattern isomorphism holds, i.e., if $P_1 \subseteq P_2$ and $P_2 \subseteq P_1$, it follows that $P_1 = P_2$.

The Maximum Common Subpattern. $P_{sub} = (N_{sub}, E_{sub})$ is the maximum common subpattern of pattern $P_1 = (N_1, E_1)$ and pattern $P_2 = (N_2, E_2)$ if and only if there exists a subset $N_1' \subseteq N_1$, a subset $N_2' \subseteq N_2$, and bijective node-mapping functions $f_1: N_{sub} \rightarrow N_1'$ and $f_2: N_{sub} \rightarrow N_2'$ such that

- $\forall\, n \in N_{sub}: AS(n) \subseteq AS(f_1(n))$
- $\forall\, n \in N_{sub}: AS(n) \subseteq AS(f_2(n))$
- $\forall\, n_1, n_2 \in N_{sub}: \{n_1, n_2\} \in E_{sub} \Leftrightarrow \{f_1(n_1), f_1(n_2)\} \in E_1 \wedge BS(\{n_1, n_2\}) \subseteq BS(\{f_1(n_1), f_1(n_2)\})$
- $\forall\, n_1, n_2 \in N_{sub}: \{n_1, n_2\} \in E_{sub} \Leftrightarrow \{f_2(n_1), f_2(n_2)\} \in E_2 \wedge BS(\{n_1, n_2\}) \subseteq BS(\{f_2(n_1), f_2(n_2)\})$
- N_{sub} and E_{sub} cannot be extended.

Pattern Similarity. To capture the chemical intuition of similarity, the definition of pattern similarity needs to include more aspects than pattern representations in *CS*. For example, two patterns might match only different molecules that are, however, very similar to each other. In other words, chemical similarity should be reflected in pattern similarity although the patterns match different molecules.

To mimic the chemical intuition of similar patterns, we propose a probability model of extended atomtype occurrence. We therefore define similarity between patterns P_1 and P_2 such that the score approximates the coverage of P_1 in P_2 using the maximum common subpattern P_{sub}. On the basis of a statistic of the occurrences of extended atomtypes, context-based similarity of nodes can be defined. We propose a similarity measure that approximates weighted coverage of $AS(n_{sub})$ in $AS(n_1)$ for compatible nodes of two chemical patterns: two nodes n_1 and n_2 are similar if and only if $AS(n_1) \cap AS(n_2) \neq \emptyset$. This definition supports the intuitive relationship $CS(P_1) = CS(P_2) \Rightarrow \mathrm{Sim}(P_1, P_2) = 1$.

COMPARING SMARTS EXPRESSIONS

Over the past decades, the SMARTS language has been established as the quasi-standard for molecular patterns. Concepts like logical expressions for atoms and bonds as well as recursive pattern definition make SMARTS an extremely powerful language. The SMARTScompare algorithm presented here will cover most of the language constructs and solve the various pattern comparison problems defined above.

SMARTScompare is based on the calculation of a maximum common substructure (MCS) between the two pattern graphs. In the following, we will use the terms *node* and *edge* whenever we mean the corresponding components of a SMARTS expression, while *atom* and *bond* are used for molecules only.

The overall strategy of SMARTScompare can be summarized as follows: First, a method is required that allows two single SMARTS nodes to be compared with each other, and here a fingerprint representing the SMARTS atomtype space *AS* is developed for this purpose. After this concept is extended to SMARTS edges, an MCS algorithm is applied. By modification of the node compatibility function, all of the problems specified above, including the similarity calculation, can be addressed with a single algorithm. Finally, we extend the algorithm to deal with recursive SMARTS expressions.

Fingerprint Generation and Pattern Preparation. *The Extended Atomtype.* The comparison of SMARTS expressions is based on a description of chemical space following a standard chemistry model.[28] Atoms receive an atomtype with a corresponding valence state covering the element, the number and types of incident bonds, and the charge. The SMARTS language is able to distinguish atoms in more detail, so the atomtype is extended with the following SMARTS-specific properties: number of attached hydrogens (no distinction between explicit and implicit hydrogens), number of aromatic bonds, hybridization, number of rings of which the atom is a part, minimum ring size, and number of ring bonds. While the original SMARTS ring features are based on the smallest set of smallest rings (SSSR)[29] concept, which is known to be ambiguous, we employ the unique ring families (URF) approach.[30] Hybridization is not part of the original SMARTS definition, but it is supported by many implementations and therefore is included in the extended atomtype states. The extended atomtypes additionally cover the number of incident aromatic bonds. This is not part of the SMARTS language but improves the distinction of atomtypes in aromatic ring systems. Although technically possible, our current implementation does not support metals, isotopes, or chirality. Figure 1 gives an example of extended atomtypes uniquely assigned to atoms of a molecule. All of the chemically feasible extended atomtypes are enumerated and stored in a list. Table 1 lists the applied criteria for chemical feasibility. Properties like connectivity ("X") that are covered by the chemistry model's valence state or atomtype need no further enumeration and are directly used in the extended atomtypes. Properties like adjacent hydrogen ("H" and "h") that can be limited on the basis of other properties of the atomtype are enumerated in this explicit range. Finally, for the ring properties ("R" and "r") there is no such limitation.

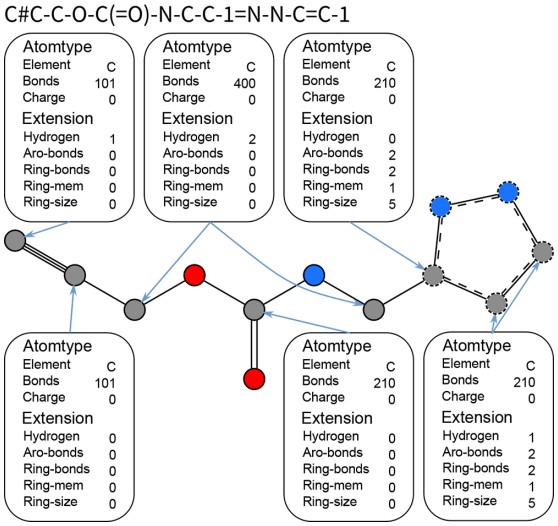


Figure 1. Example of a molecule with the unique annotations of extended atomtypes at the carbon atoms. The three-digit bond code is the number of localized single, double, and triple bonds.

Table 1. Criteria for Chemical Feasibility of the Enumerated Extended Atomtypes

$n_{Hydrogen} + n_{Aro\text{-}bonds}$	$\leq$	n_{Bonds}
$n_{Hydrogen} + n_{Ring\text{-}bonds}$	$\leq$	n_{Bonds}
$n_{Aro\text{-}bonds}$	$\leq$	$n_{Ring\text{-}bonds}$
$n_{Ring\text{-}bonds}/2$	$\leq$	$n_{Ring\text{-}mem}$

DOI: 10.1021/acs.jcim.9b00250
J. Chem. Inf. Model. 2019, 59, 2560−2571

Their enumeration is artificially limited to at most four URFs (rings) of which an atom is a part and a maximum ring size of 24 atoms. There are additional extended atomtypes for atoms in more than four URFs and in rings with a minimum size larger than 24, but they can no longer be distinguished from each other. Table 2 presents all of the covered SMARTS node properties and how they are represented in the extended atomtypes. The enumeration of all feasible feature combinations results in 20 565 unique extended atomtypes, called the atomtype space. A more detailed enumeration scheme is given in the Supporting Information (see SI1).

Table 2. SMARTS Node Properties and How They Are Represented in the Extended Atomtypes

SMARTS node property	symbol	represented in extended atom type
Aromaticity	a, ...	enumerated property
Charge	+, −	part of the atomtype (chemistry model)
Connectivity	X	sum over incident bond types (chemistry model)
Degree	D	difference of connectivity and hydrogen property
(ExplicitlImplicit) hydrogen	H, h	enumerated property
Hybridization	^	enumerated property
Ring connectivity	x	enumerated property
Ring size	r	enumerated property
Ring membership	R	enumerated property
Atomic number	#	part of the atomtype (chemistry model)
Valence	v	weighted sum over incident bond types (chemistry model)

The atomtype space is the basis of a fingerprint descriptor for SMARTS nodes consisting of 20 565 bits, in which each bit corresponds to a specific extended atomtype as depicted in Figure 2. This descriptor captures any logical expressions the SMARTS node description might contain.

Basic Fingerprint Generation. The generation of the specific SMARTS node fingerprints consists of two major parts (see Figure 3):

1. All bits corresponding to extended atomtypes fulfilling the SMARTS node expression are set to active.
2. All set bits violating properties of the environment of the SMARTS node are reset.

Table 3 lists some criteria used for bit reduction in the second step. They are also part of an extended list given in Table S4 in the Supporting Information. How these properties are retrieved from the node environments and postprocessed is also shown in the Supporting Information (see SI2).

For each edge of a SMARTS expression, a fingerprint of nine bits representing bond features is created. Edge fingerprint bits represent single, double, triple, and directed bonds in and outside of rings and aromatic ring bonds. The first four bits belong to nonring bonds and the following four bits to ring bonds in the given order. The ninth bit represents an aromatic ring bond. The edge fingerprint is depicted in Figure 4.

The resulting edge fingerprint can be further pruned by examining the neighboring nodes and edges. If any of the incident nodes does not allow a certain bondtype in its environment, the corresponding bits in the edge fingerprint are reset. After that, incident nodes and edges that could be affected by an update are updated as well until convergence is reached. The node fingerprint update is based on the environment criteria listed in Tables 3 and S4. The criteria for the edge fingerprint updates are listed in Table 4.

We want to highlight that fingerprint pruning is necessary for subset detection. The pruning strategy does not guarantee that each bit in all fingerprints is necessary to describe $CS(P)$ as defined for the generic chemical pattern in eqs 2 and 3. It should be noted that the pruning strategy is conservative since all pruned atom- or bondtypes are irrelevant to describe $CS(P)$.

Small rings up to a size of eight nodes can be detected as aromatic rings. If every node of a ring is aromatic, the edge fingerprints are pruned to reset all bits representing non-aromaticity. This feature allows SMARTS expressions like c1ccccc1 and c:1:c:c:c:c:c:1 to be considered as equal. In general, there is no possibility to detect a generic edge as aromatic within the SMARTS context. In macrocycles there can be aliphatic single bonds between aromatic atoms. Consequently, it is necessary to restrict the supported ring sizes for this feature. According to the NAOMI chemistry model,[28] the limit is set to a maximum number of eight nodes for those rings. This feature is enabled by default in SMARTScompare because it is cumbersome to mark aromaticity explicitly and most often it is

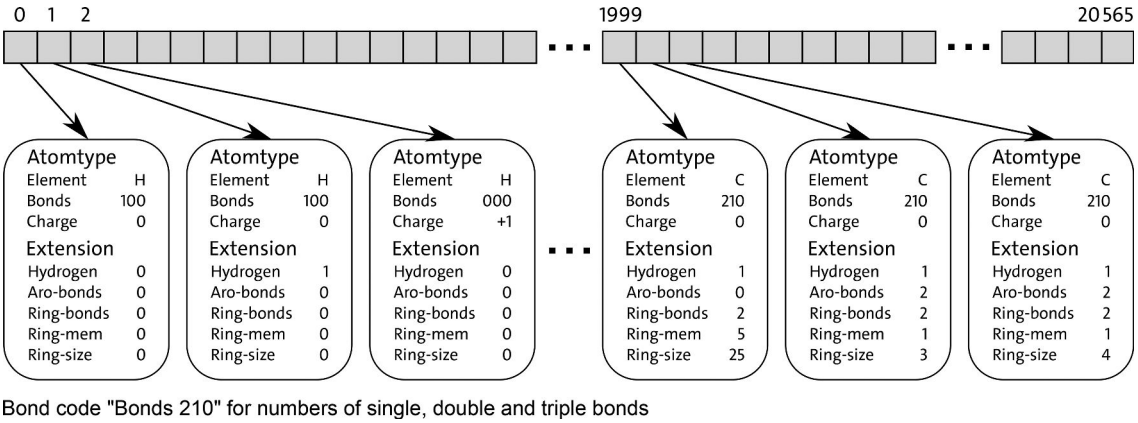


Figure 2. Depiction of the systematically enumerated 20 565 extended atomtypes. The assigned indices correspond to the implementation list position. Hybridization is omitted in the depiction.

DOI: 10.1021/acs.jcim.9b00250
J. Chem. Inf. Model. 2019, 59, 2560−2571

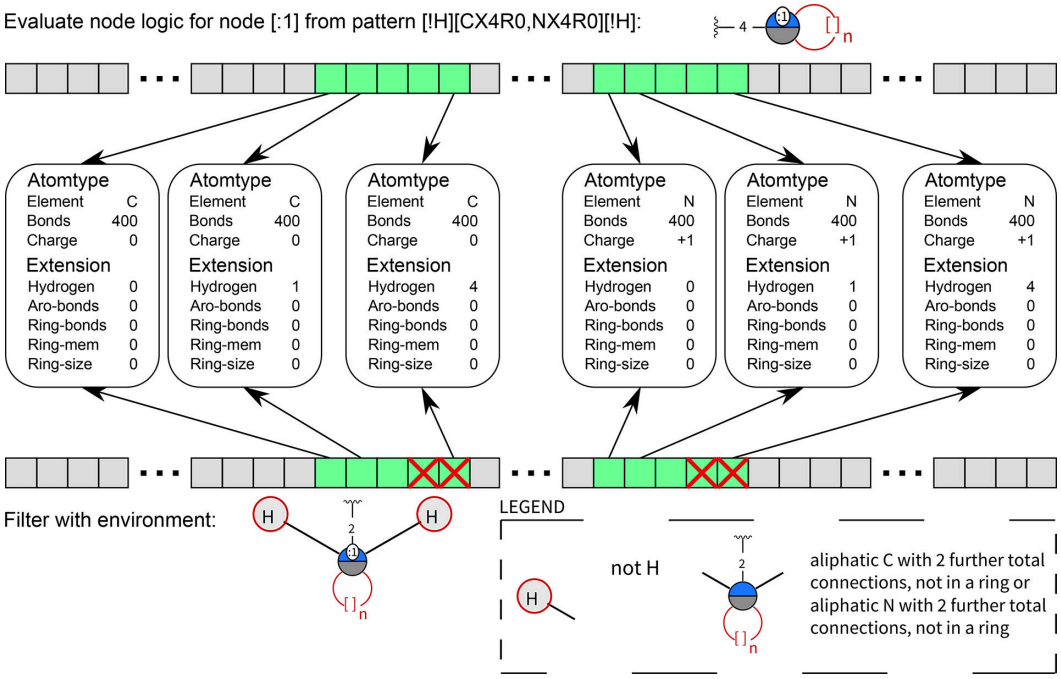


Figure 3. Depiction of the fingerprint generation for the second node (shown at the top) of the pattern [!H][CX4R0,NX4R0][!H]. Below the pattern, the fingerprint is shown. In the center of the figure there are some extended atomtypes corresponding to the bits set in the fingerprint. The lower depiction of the fingerprint shows reset bits of the fingerprint belonging to atomtypes that are not compatible to the environment of the node (see the pattern depiction at the bottom left). Pictures were created with SMARTSviewer.[18,19]

Table 3. Criteria for Fingerprint Bits That Arise from the Node Environment

node environment		extended atomtype
env_{Bonds}	$\leq$	n_{Bonds}
env_{Valence}	$\leq$	$n_{n_{\text{Bonds}}\text{alence}}$
env_{Hydrogen}	$\leq$	n_{Hydrogen}
$env_{\text{!Hydrogen}}$	$\leq$	$n_{\text{!Hydrogen}}$
$env_{\text{Double-bond}}$	$\leq$	$n_{\text{Double-bond}}$
$env_{\text{Triple-bond}}$	$\leq$	$n_{\text{Triple-bond}}$
$env_{\text{Valence}} - env_{\text{Degree}}$	$\leq$	$n_{\text{Valence}} - n_{\text{Degree}}$
$env_{\text{Aro-bond}}$	$\leq$	$n_{\text{Aro-bond}}$
$env_{\text{Ring-bond}}$	$\leq$	$n_{\text{Ring-bond}}$
$env_{\text{!Ring-bond}}$	$\leq$	$n_{\text{!Ring-bond}}$

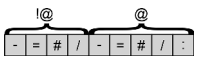


Figure 4. Edge fingerprint. The first four bits represent acyclic bondtypes, and the final five bits represent the cyclic ones. The bits are annotated with their meaning in SMARTS symbols.

Table 4. Edge Fingerprint Environment Pruning Criteria[a]

edge type	criterion
Single bond	both incident nodes allow single bonds
Double bond	both incident nodes allow double bonds
Triple bond	both incident nodes allow triple bonds
Directed bond	any incident node allows double bonds
Aromatic bond	both incident nodes can be aromatic
Ring bond	both incident nodes can be in rings

[a]If any incident node of an edge does not allow a certain bondtype in its environment, the corresponding bits in the edge fingerprint are reset. For example, if the fingerprint of an node incident to an edge contains no extended atomtype with incident single bonds, the edge cannot be a single bond.

implicitly part of SMARTS patterns. As shown in Validation, this feature can sometimes lead to erroneous subset claims.

SMARTS recursion has an influence on fingerprint generation too. The effects are described briefly after the matching algorithm and in detail in the Supporting Information (see SI3).

The Matching Algorithm. The mapping between two SMARTS patterns is calculated using a clique-based connected maximum common induced subgraph approach.[31−33] In the first step, a product graph, often called a compatibility graph, is built. Cliques in this graph represent common subgraphs between the two SMARTS expressions. For graphs, this is a relatively simple step since only the labels of nodes and edges are compared. For SMARTScompare, the compatibility between nodes and edges depends on the problem to be solved. In all cases, we use the subset of all matching atomtypes $AS(n)$ for a node n or bondtypes $BS(e)$ for an edge e. The compatibility criteria are listed in Table 5

For pattern isomorphism, we expect the MCS to be complete, i.e., to match all nodes of the first pattern to all nodes of the second. For subpattern isomorphism, all nodes of the first pattern (the more generic one) have to be matched. In all other cases, we search for the maximum connected common subgraph. By the use of a modified connected MCS algorithm of the Parasols library,[34] all MCSs of maximum size are returned as the result.

DOI: 10.1021/acs.jcim.9b00250
J. Chem. Inf. Model. 2019, 59, 2560−2571

Table 5. Fingerprint Compatibility Criteria for the Different Problem Types

comparison problem	compatibility criteria: for two nodes n_1 and n_2	compatibility criteria: for two edges e_1 and e_2
pattern isomorphism	$AS(n_1) = AS(n_2)$	$BS(e_1) = BS(e_2)$
subpattern isomorphism	$AS(n_2) \subseteq AS(n_1)$	$BS(e_2) \subseteq BS(e_1)$
similarity search	$AS(n_1) \cap AS(n_2) \neq \emptyset$	$BS(e_1) \cap BS(e_2) \neq \emptyset$

An example execution of the algorithm performing a similarity search between two patterns is summarized in Figure 5. Figure 6

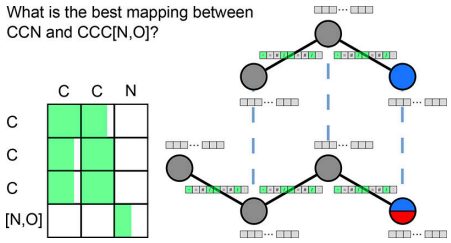


Figure 5. Matching procedure for nonrecursive SMARTS patterns. For each node and edge, a fingerprint is generated, and the compatibility graph is built. In this figure it is represented by the compatibility matrix on the left. In a similarity search, two nodes are compatible if their fingerprints share at least one bit. Partially filled cells of the matrix indicate the percentage of shared bits. Finally, the MCS algorithm computes all cliques of maximum size, thus computing the maximum common subgraph of the two SMARTS patterns.

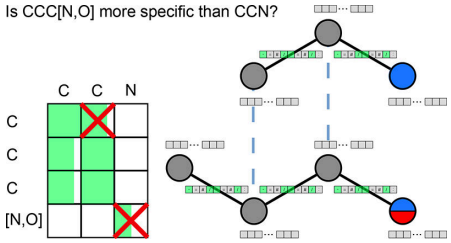


Figure 6. Same matching procedure as in Figure 5, but for the subset search. This employs additional constraints, such that each node of CCC[N,O] must be more specific than compatible nodes of CCN. The two nodes that are not more specific are crossed out. Beyond that, the computation stays the same. CCC[N,O] is not a subset of CCN, since the mapping does not completely cover CCN.

shows a similar example executing a subset search for more specific patterns. These two examples emphasize the algorithmic similarity between the available execution modes similarity search and subset search: there is no difference except for the compatibility criterion. In both figures, the symbolic annotation of fingerprints is performed first. Then the compatibility graph, represented as a compatibility matrix, is created. Finally, the maximum common subgraph is depicted (blue connection lines).

Result Compilation. The algorithm computes all maximum common subgraphs for each comparison with the patterns and their recursions. On the basis of these MCS results, the final mapping result is produced. The computed mapping between two patterns should be independent of pattern-invariant aspects of the SMARTS string, such as node order, order of logic operands, etc. This is achieved by enumerating all possible mappings and scoring them. The fingerprint pairs of all mapped nodes are scored using a similarity function like the Tanimoto coefficient. The result yielding the highest score is finally chosen.

In the case of subset search including recursive patterns, additional constraints have to be applied. They are described in the Supporting Information (see SI5).

Finally, there are a few more aspects that are handled during result compilation. The definition of the generic chemical pattern contains several constraints regarding pattern uniqueness. By pruning the SMARTS node and edge fingerprints, SMARTScompare tries to come as close as possible to a generic pattern representation. This is not always achievable. Explicit hydrogen nodes ([H] or [#1]) are always included in their neighbor's node fingerprints. Thus, they can be considered redundant. The same holds true for wildcard nodes connected with exactly one acyclic edge. Since their fingerprints are completely determined by their environment, they can be considered redundant as well.

SMARTS Recursion. SMARTS recursion is a node property that describes the structural environment of the node to which it is attached. The recursive description, a SMARTS pattern by itself, is compared against the molecule, keeping the first node assigned to the atom matched while matching the recursive description independent from the main pattern (for a detailed description of SMARTS recursion, see SI3–SI5).

Recursive environment descriptions are frequently used, but unfortunately, they substantially complicate the comparison of SMARTS patterns. To handle SMARTS recursion correctly, the matching procedures are usually recursive by themselves. To integrate SMARTS recursion into the comparison algorithm, fingerprints influenced by SMARTS recursion have to be modified, the MCS algorithm has to be implemented in a recursive fashion, and the logic between recursive expressions has to be taken into account.

Recursion Fingerprint Handling. Node environments formulated in SMARTS recursion have an influence on fingerprint generation. Positive environments participate with their first node, since it has to be fulfilled at the attachment node. Negated environments have to be considered for fingerprint generation only if they represent exactly one node; otherwise, they do not influence any node fingerprints.

SMARTS recursion is explicitly considered after the first fingerprint pruning step for the basic pattern. Handling of SMARTS recursion is divided into the following steps:

1. Generation of node and edge fingerprints for the recursive environment pattern.
2. Removal of redundant SMARTS recursion, i.e., recursive environments that are more generic than the basic pattern.
3. Evaluation of the logic in the node expression and restructuring into a disjunctive normal form (DNF). In this context, alternatives are defined as conjunctions of SMARTS recursion, and the node expression is a disjunction of alternatives.
4. Removal of redundant SMARTS recursions within the formulated alternatives.
5. Removal of redundant alternatives from the generated DNFs in SMARTS node expressions.
6. Update of node fingerprints using all of the information on SMARTS recursion grouped into alternatives.
7. Final pruning of all fingerprints using the updated fingerprints of nodes with SMARTS recursion.

DOI: 10.1021/acs.jcim.9b00250
J. Chem. Inf. Model. 2019, 59, 2560−2571

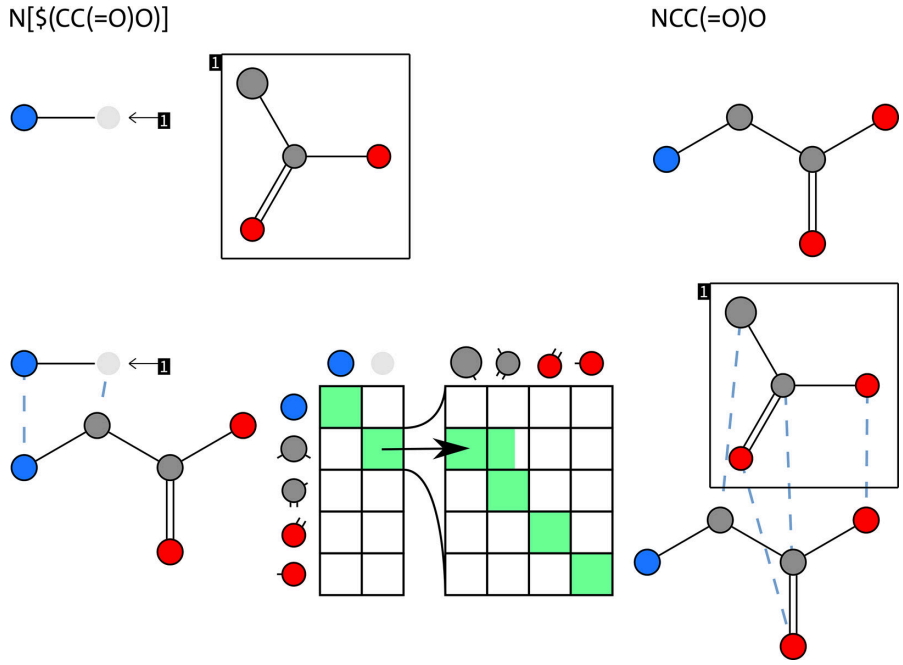


Figure 7. Recursive matching procedure for the comparison of N[$(CC(=O)O)] and NCC(=O)O. The patterns are shown at the top. At the bottom left, the MCS of the structures is shown. In the bottom middle, the compatibility matrix for the structure compatibility and the nested compatibility matrix for the MCS calculation of the recursion of the first pattern and the structure of the second one are shown. At the bottom right, the MCS of the recursion and the structure is depicted. Whenever a recursive node's fingerprint is compatible with another node, the compatibility is finally derived by performing the matching recursively. This means that nodes are compatible if and only if their environments are compatible.

A detailed description of the SMARTS recursion handling during atomtype fingerprint generation is given in the Supporting Information (see SI3).

Recursive Matching. To determine the compatibility of nodes with recursive environments, the matching algorithm is recursive by itself. If at least one of the mapped nodes carries a SMARTS recursion, the compatibility of the nodes is determined by comparing all required combinations of the SMARTS recursive environments. A general overview of this recursive matching can be seen in Figure 7. The first pattern has two nodes, one of them with SMARTS recursion, and the second pattern does not have a recursive expression. The compatibility of the node with SMARTS recursion and a fingerprint-compatible node is a result of a recursive comparison of the SMARTS recursion of the first pattern and the second pattern. This is shown by the two compatibility matrices in Figure 7.

To handle SMARTS recursion correctly, the procedure has to solve the following tasks:

1. Matching a SMARTS recursion to the basic pattern.
2. Matching a SMARTS recursion to another SMARTS recursion.
3. Evaluating the pattern logic to derive constraints for the subset search.
4. Correcting results with respect to negated SMARTS recursions.

A detailed description of all mentioned procedures is given in the Supporting Information (see SI4).

Finally, nodes are compatible if their fingerprints are compatible and their SMARTS recursions are compatible with respect to the comparison mode of interest (subset or similarity).

The presented procedure follows an indirect recursion that occurs when the compatibility of nodes with SMARTS recursion is determined. The MCS algorithm needs to determine the node compatibility of all nodes, including those with SMARTS recursion. To determine the compatibility of nodes with SMARTS recursion, the SMARTScompare algorithm is applied to the recursive SMARTS pattern.

SMARTS Recursion during Result Compilation. The matching procedure results in several partial results for the basic pattern and the recursions. For the final result, an optimal mapping between these patterns is required. A detailed description how this mapping is computed is given in the Supporting Information (see SI5).

Further Features. The algorithm described above was designed for SMARTS comparisons. However, during its development we noticed its potential for error detection in SMARTS expressions:

- The exhaustive atomtype state generation is a powerful chemical feasibility verification for SMARTS expressions. This procedure verifies that each node corresponds to at least one chemical state. This test includes the node's recursive environment as well as the node's logical expression. In general, SMARTS node expressions exceeding element-specific valences and impossible logical constructions are detected. For example, the algorithm will detect the nitrogen in patterns like C[$(N(=O)=O)] as a nitrogen with a valence of 5, which is unsupported by the chemistry model.

DOI: 10.1021/acs.jcim.9b00250
J. Chem. Inf. Model. 2019, 59, 2560−2571

- The algorithm is able to detect many types of redundancy in SMARTS expressions. On the basis of this information, SMARTS expressions can be optimized with respect to computational complexity and human readability. This includes even some kinds of redundancy within the logical constructions of SMARTS recursion. If a recursion is more generic than the environment of its attachment node, it does not specify the pattern and thus could be omitted, as in [$(CO)]O. If SMARTS recursions in logical structures are more specific or more generic, they could be redundant as well. For example, the pattern [(CO)(C*),$(C*)] gives the same description as [$(C*)]. More details are provided in the Supporting Information (see SI3).

GENERALITY OF THE APPROACH

The above-described algorithm for SMARTS pattern comparison works correctly in most practical applications, but it is not complete and correct in general. In this section, we summarize unsupported features and cases where the algorithm fails.

As mentioned before, all kinds of chirality and isotopes are unsupported to date. Isotopes are easy to include in the fingerprint approach, but they are unsupported within the chemistry model of the underlying NAOMI[28] cheminformatics library. Disconnected SMARTS patterns are also unsupported by the algorithm. Chirality is not considered during fingerprint generation. The verification of ligand orders, which is required for chirality handling, is currently unsupported. In order to extend the algorithm in this direction, it would be necessary to verify the ligand orders within the MCS algorithm or in the final result compilation. Currently our implementation denies subset relations of patterns including those unsupported SMARTS features.

One core aspect of the SMARTScompare algorithm is extensive valence counting. With regard to unspecific edges, the abilities to detect pattern identity are limited. As shown in Table 4, edge fingerprints are pruned with regard to their neighboring nodes. This kind of pruning is not chemically aware of whether a certain bondtype is allowed for a specific edge. It is only aware of the existence of a specific bondtype within the set of incident edges of a node.

Saturation of a bondtype is undetected. This might lead to missed pattern identities, as illustrated in the comparison of the patterns *=[CX3]~[CX3] and *=[CX3]-[CX3]. In this example, the algorithm does not detect that the wildcard bond has to be a single bond resulting in pattern identity. A similar example is shown in Figure 8. The two oxygen atoms can only be connected via a double bond and a single bond. To solve this issue, it is necessary to detect and break the symmetry. Both examples can be extended to a general case where SMARTScompare is unable to detect pattern identities because some fingerprints still require pruning to fulfill the non-redundancy requirements described in eqs 2 and 3.

A generic example of patterns of this class exploits the fact that a certain bond occurs exactly once at all incident bonds to an atom. Then there have to be at least two incident edges if those fingerprints allow two different bondtypes. If the pattern is symmetric, then SMARTScompare is unable to detect pattern identity with a localized variant. We can compare the patterns O~[CX3v4H1]~O and O=[CX3v4H1]O, which are not detected as equal by SMARTScompare. Since the pattern with the wildcard edges is symmetric and there is exactly one double

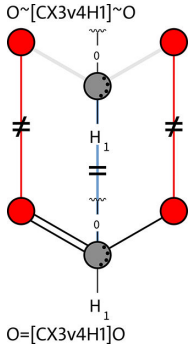


Figure 8. Two SMARTS patterns for formic acid that are not detected as equal. The upper pattern is written more generically than necessary. The pattern is symmetric, and the two wildcard bonds can only be a single bond and a double bond. Thus, neither the edges nor the two nodes describing oxygen are detected as equal.

bond incident to the central carbon node, the two patterns are actually identical.

Compatibility of logical constructions within environments can be hard to determine, and the approach might miss subset relations. There are constructions that are known to be undetectable, such as [$(C[N,O])] and [$(CN),$(CO)]. These two SMARTS match if there is a CN or a CO substructure. The subset relation is detected from the pattern with one recursion to the pattern with two recursions but is missed for the other way around because of algorithmic restrictions for node compatibility. Figure 9 depicts the

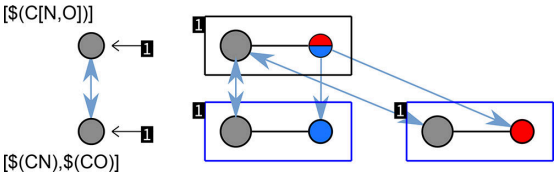


Figure 9. One limitation of the approach is that the subset search is strict and enforces the subset relation for every node. Although the two patterns shown are equal, the approach detects the subset relation in only one direction.

situation. Our algorithm compares recursive expressions individually but is not able to combine chemical information on two recursive expressions into a single one. In general, if there are two alternatives for recursive SMARTS expressions that differ in only one node, the algorithm is unable to detect pattern identity to a pattern where the two recursions are combined into one recursion with the alternative combined within the node.

Another kind of undetected subset relation is shown in Figure 10. Besides the recursion, the two patterns are identical. Thus, all nodes of the structure should be compatible. However, the two nodes of the lower pattern connected by the cyan arrows have different fingerprints, although they can always match the same atom. In fact, the fingerprint of the node in the recursion ('[NH1]') is more general than its counterpart in the basic pattern. However, chemical information from neighboring nodes of the attachment node is not used to prune fingerprints in recursions and the other way around from neighboring nodes of recursions' root nodes. Because of this fingerprint pruning limitation, c−!@[NH1]C(=O)N is not detected to be equal to

DOI: 10.1021/acs.jcim.9b00250
J. Chem. Inf. Model. 2019, 59, 2560−2571

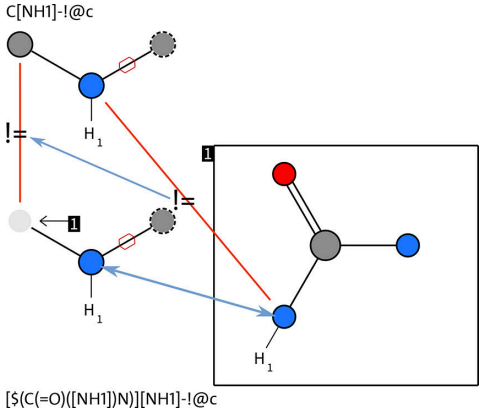


Figure 10. Another limitation of subset detection. For a recursive pattern which is compared with its nonrecursive structure, one of the recursions nitrogen nodes ('[NH1]') should have the same fingerprint as the corresponding node of the pattern's structure. Currently SMARTScompare does not compute recursion to structure mappings within a pattern to improve fingerprint pruning. This leads to a situation where the recursion is not mappable to the other pattern, and thus, the subset relation is missed. For this kind of error, it is necessary for nodes of the recursion to be described the same as nodes of the nonrecursive pattern without being redundant.

[$(C(=O)([NH1])N)][NH1]-!@c. Another example of a missed pruning opportunity in the basic pattern is the fact that [NH2][CH2]N is not detected to be equal to N[$([CH2]-[NH2])]N. This limitation holds for generic recursion expressions that do map parts of the environment of their attachment node. Their fingerprints could be pruned with the corresponding adjacent nodes of the attachment node only if the mapping is unique. The other way around with a unique mapping, fingerprints in the basic pattern could be pruned as well.

In general, negated SMARTS recursions have no influence on fingerprints of neighboring nodes or incident edges. Consider the pattern O[C!$(C=C)](N)~C. The wildcard bond C~C could be pruned with the knowledge of the negated SMARTS recursion !$(C=C) and become C-C. This kind of pruning using negated SMARTS recursion is not supported.

Negated SMARTS recursion is supported by the algorithm, but it is only partially considered in similarity searching. Similarity values do not consider negated SMARTS recursion except when there is a clear incompatibility. For subset search, negated SMARTS recursion in the more generic pattern and in the more specific pattern is supported.

The implementation of negated SMARTS recursion handling is designed to avoid deniable subset relations. Thus, negated recursion might lead to undetected subset relations. There are three criteria that allow subset detection for generic patterns with negated SMARTS recursion:

1. There is a compatible negated recursion in the more specific pattern.
2. The alternative of which the negated SMARTS recursion is part is not needed to detect compatibility.
3. The negated SMARTS recursion root node is incompatible with the mapping partner of its attachment node.

For any negated SMARTS recursion that violates all three criteria, subset relations are not detected. These criteria might be too strict to detect all possible subset relations, but they ensure that there are no false positives. Except for the pruning possibility, negated SMARTS recursion in the more specific pattern has no influence on the subset detection since it always specifies a pattern.

VALIDATION

Since no comparable algorithmic approach is available, we decided to approximately validate our approach using a large compound collection. On the basis of a large data set, one can determine candidates for subpattern relationships. If two patterns P_1 and P_2 both match several compounds of a diverse compound set and each molecule matched by P_2 is also matched by P_1, then P_2 is a candidate to be more specific than P_1. If there is any molecule matched by P_2 that is not matched by P_1, then P_2 is not more specific than P_1.

Validation of Subset Relationships. The compound set used contained ~370 million molecules and comprised all of the ZINC15[35] 2D compounds that were available on June 8, 2017. This set is called the *validation data set*. The collection of patterns was based on the data used for a systematic benchmark of subgraph algorithms[36] and contained 80 073 patterns in total. It consisted of SMARTS patterns as well as substructure-like patterns without any SMARTS properties, and 38 941 of those patterns matched at least one molecule in the validation data set. Patterns matching exactly one hydrogen were discarded and not part of that group of patterns. Of the matching patterns, 3393 used explicit SMARTS properties and the remaining ones were substructures without SMARTS properties.

This experimental setup is used to provide data for possible subset relations between SMARTS patterns. We sorted the entries for patterns by the number of matching molecules in descending order and extracted pattern pairs with potential subset relationships. There are 3 432 874 pattern pairs for which the sets of matched molecules are in a subset relationship. SMARTScompare determined 1 830 553 pattern pairs. The confusion matrix is shown in Table 6. In majority of the comparisons, the experiment and SMARTScompare agree that patterns are not in a subset relation. This is as expected using a diverse pattern set. For 53.3% of the experimental subset pairs, SMARTScompare also computes this relation. There are seven pattern pairs where the experiment proved SMARTScompare to be wrong (see Table 7). Six of them have the pattern [S;D3](-

Table 6. Comparison of the Subset Relations Detected by SMARTScompare and the Reference Data[a]

		experiment		
		S	N	total
SMARTScompare	S	1 830 546	7	1 830 553
	N	1 602 328	1 512 968 600	1 514 570 928
	total	3 432 874	1 512 968 607	1 516 401 481

[a]"S" stands for "subset relation" and "N" for "no subset relation".

DOI: 10.1021/acs.jcim.9b00250
J. Chem. Inf. Model. 2019, 59, 2560−2571

Table 7. List of All False Subset Predictions during the Validation

pattern	supposed subset pattern
[+]	[S;D3](−N)(−[c,C])(−[c,C])
[+,++,+++]	[S;D3](−N)(−[c,C])(−[c,C])
[O+,o+,S+,s+]	[S;D3](−N)(−[c,C])(−[c,C])
[C+,Cl+,I+,P+,S+]	[S;D3](−N)(−[c,C])(−[c,C])
[SX3](*)	[S;D3](−N)(−[c,C])(−[c,C])
[#16v3,#16v5]	[S;D3](−N)(−[c,C])(−[c,C])
a1aaa2a(a1)aaa(a2):a	[n+]12nc3c4c5c(c3cc1c(nc(c2C)C)C)cccc5ccc4

N)(-[c,C])(-[c,C]) as a subpattern, and four of them contain positive charges. The disagreement is caused by the underlying NAOMI chemistry model. Although the model knows sulfur valence states with four explicit bonds, all of them are annotated with no neighboring hydrogen atoms. The compound ZINC000100315425 is out of the description domain of the chemistry model. In the pattern [S;D3](-N)(-[c,C])(-[c,C]), the sulfur node is expected to match sulfur with three single bonds and additional hydrogens. Thus, only positively charged sulfur atoms with three single bonds are assumed to match the pattern. Figure 11 shows all of the mentioned patterns and the ZINC compound. The relevant sulfur is encircled.

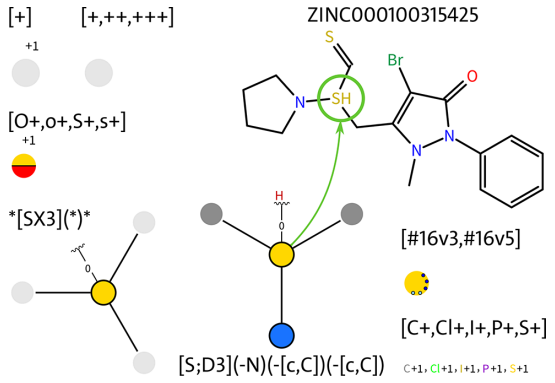


Figure 11. Six patterns that found [S;D3](−N)(−[c,C])(−[c,C]) as a subpattern that could not be experimentally validated, as ZINC structure ZINC000100315425 shows. The encircled sulfur has an explicit adjacent hydrogen, which is outside the model domain of the NAOMI chemistry model. There is no extended atom type representing a sulfur with four bonds and adjacent hydrogen. There is, however, an extended atom type for a positively charged sulfur with three bonds. According to the model, the sulfur should be positively charged and have only three single bonds.

The only remaining case in the first run is about a condensed aromatic pattern. ZINC structure ZINC000000386967 revealed that there may be small rings whose atoms are all aromatic but not every bond of this ring is aromatic, as shown in Figure 12. In this structure, a cycle of size 5 connects two polycyclic aromatic ring systems. The two bonds highlighted in green are not aromatic, although all five ring atoms are aromatic. This shows that the optional feature to detect bonds in rings consisting exclusively of aromatic atoms automatically as aromatic is not always right for SMARTS patterns.

A look at the remaining experimental subset pairs reveals many cases for which the validation data set is insufficient to detect the subtle differences between patterns. Often, certain fragments imply a constant reoccurring environment in the molecules. As long as other environments are theoretically possible, the experimental subset relationship is wrong.

This can be demonstrated with the generic carbonyl oxygen. Besides the typical atomtype for carbonyl oxygen with exactly one double bond, named 'O(010)', the chemistry model also contains an atomtype with a positively charged oxygen with a single and a double bond, named 'O(110)+'. The pyrylium ion justifies its existence. In the case of the generic carbonyl C=O, the model has to assume that the 'O(110)+' atomtype is possible even if there is no molecule where it occurs. Thus, patterns like C=O and C=[OX1] are not considered identical by the model, although one might expect this behavior.

Another common example is the identification of acyclic parts of the patterns. If a node or an edge of a pattern is explicitly acyclic, there are often several subset pattern pairs. For them the experimental subset relation is valid because there is no molecule in the validation data set for which the structure is part of a ring. Figure 13 shows a pattern pair combining both examples. The upper pattern describes two carbons, one of which is acyclic. In the experiment the same holds true for the two chain carbons in the lower left pattern. For SMARTScompare it is possible that the whole substructure is still part of a ring. Therefore, the singly bonded oxygen would be positively charged and have three bonds, as in oxonium. The modification of the pattern shown at the lower right enables subset detection by SMARTScompare. In conclusion, there are several experimentally valid subset relations that could be denied if there were molecules showing the pattern difference. This emphasizes the need for a compound-independent pattern comparison method like SMARTScompare.

CONCLUSION AND OUTLOOK

We have presented a novel solution to the generic chemical pattern comparison problem. The algorithm employing a maximum common subgraph (MCS) calculation is applicable to any substructure-based language for molecular patterns. It is inspired by a theoretical comparison approach for which patterns are identical if and only if their representations in chemical space are identical. A chemistry model is the foundation on top of which we designed a fingerprint descriptor for the pattern's nodes and edges. It is important that the fingerprint includes all available information on the node or edge and its environment in the pattern. Furthermore, redundancy in the pattern hampers correct comparison and has to be removed beforehand.

We implemented our algorithm for the widely used SMARTS language. The fingerprints cover most SMARTS features, including ring properties up to a ring size of 25. As a side effect, it is possible to detect common pattern errors resulting from incorrect valences or contradictions as well as redundant elements in the pattern. SMARTS recursion turns out to be a major difficulty in comparison of SMARTS patterns. The recursively defined environments usually describe more than one node, such that a truly recursive schema is unavoidable for its handling. As a proof of concept, the algorithm is implemented into the software tool SMARTScompare.

With SMARTScompare, it is possible to analyze SMARTS pattern pairs independent of their string representation. The algorithm is capable of similarity assessments and can detect subset relations between patterns. We validated the subset predictions of SMARTScompare using a large pattern collection applied to more than 370 million compounds. In that

DOI: 10.1021/acs.jcim.9b00250
J. Chem. Inf. Model. 2019, 59, 2560−2571

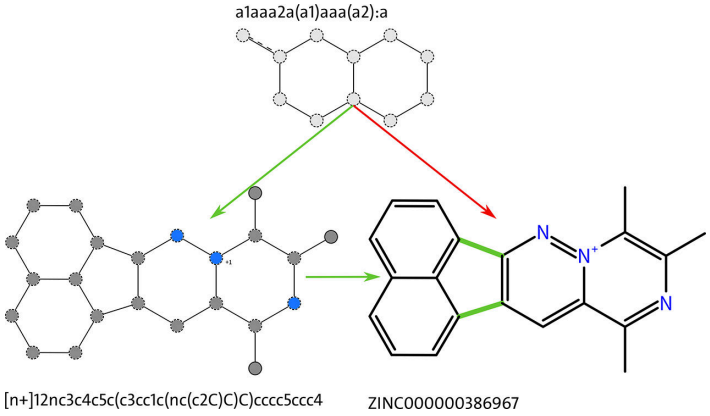


Figure 12. Two patterns in a subset relation in ZINC structure ZINC000000386967 that prohibits the subset detection in the validation. This structure shows that the detection of aromatic ring bonds can be erroneous.

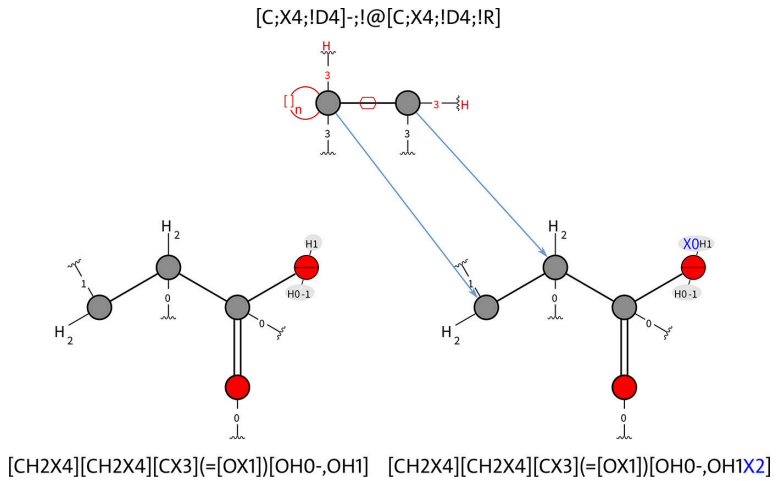


Figure 13. An experimentally valid pattern subset relation and the pattern modification such that SMARTScompare also detects it. Because of a seldom-used atomtype, it is still possible that the lower pattern is part of a ring. Upon application of a small modification, SMARTScompare also detects the subset relation.

experiment, SMARTScompare found thousands of experimentally valid pattern subset relations. The few observed issues merely affect the description domain of the underlying chemistry model and the handling of aromaticity. They have no impact on the correctness of the approach.

In a follow-up publication[27] we introduce a meaningful pattern similarity approach and an optional aromaticity detection method supporting the chemical intuition of aromaticity, which has to be explicitly modeled in SMARTS. We then demonstrate a showcase application of SMARTScompare by performing an in-depth similarity analysis of publicly available SMARTS pattern collections.

■ ASSOCIATED CONTENT

Supporting Information

The Supporting Information is available free of charge on the ACS Publications website at DOI: 10.1021/acs.jcim.9b00250.

Additional chapters providing in detail descriptions of complex parts of the presented algorithm (AdditionalMethodChapters.pdf) and all of the SMARTS patterns that were used for the validation experiments (ValidationPatterns.smarts) (ZIP)

■ AUTHOR INFORMATION

Corresponding Author

*E-mail: rarey@zbh.uni-hamburg.de. Phone: +49 (40) 428387351.

ORCID

Robert Schmidt: 0000-0002-7089-4842
Matthias Rarey: 0000-0002-9553-6531

Present Addresses

†F.O.: Evotec AG - Manfred Eigen Campus, Essener Bogen 7, 22419 Hamburg, Germany.
‡H.-C.E.: SAP SE - Innovation Center Potsdam, Konrad-Zuse-Ring 10, 14469 Potsdam, Germany.
§A.M.: Titel Media GmbH, Ritterstr. 9, 10969 Berlin, Germany.

DOI: 10.1021/acs.jcim.9b00250
J. Chem. Inf. Model. 2019, 59, 2560−2571

Author Contributions

H.-C.E., A.M., and M.R. developed the initial concept of atomtype-based SMARTS subset comparisons (first prototype, 2011). F.O. and M.R. extended the comparison concept toward similarity using a subgraph algorithm (second prototype, 2015). R.S. and M.R. developed the concepts for SMARTS recursion integration, fingerprint pruning, SMARTS node similarity, and MCS. R.S. implemented the SMARTScompare software based on the NAOMI cheminformatics library (current version). E.S.R.E. provided application scenarios for the SMARTScompare approach and analyzed and interpreted the results. R.S., E.S.R.E., and M.R. wrote the manuscript.

Notes

The authors declare the following competing financial interest(s): M.R. declares a potential financial interest in the event that the SMARTScompare software is licensed for a fee to non-academic institutions in the future.

SMARTScompare and SMARTScompareViewer, a tool for visualization of SMARTS relationships, are available for Linux, OS X, and Windows as part of the NAOMI ChemBio Suite (https://uhh.de/naomi) and are free for academic use and evaluation purposes. Furthermore, SMARTScompare and the SMARTScompareViewer are integrated in the SMARTSview server (https://smarts.plus). All feedback is greatly appreciated and supports the further development of SMARTScompare.

REFERENCES

(1) Daylight Chemical Information Systems, Inc. http://www.daylight.com/dayhtml/doc/theory/theory.smarts.html (accessed Aug 20, 2018).

(2) Ash, S.; Cline, M. A.; Homer, R. W.; Hurst, T.; Smith, G. B. SYBYL Line Notation (SLN): A versatile language for chemical structure representation. *J. Chem. Inf. Comput. Sci.* **1997**, *37*, 71−79.

(3) Proschak, E.; Wegner, J. K.; Schüller, A.; Schneider, G.; Fechner, U. Molecular Query Language (MQL)—A context-free grammar for substructure matching. *J. Chem. Inf. Model.* **2007**, *47*, 295−301.

(4) Gaulton, A.; et al. The ChEMBL database in 2017. *Nucleic Acids Res.* **2017**, *45*, D945−D954.

(5) Baell, J. B.; Holloway, G. A. New substructure filters for removal of pan assay interference compounds (PAINS) from screening libraries and for their exclusion in bioassays. *J. Med. Chem.* **2010**, *53*, 2719−2740.

(6) Bruns, R. F.; Watson, I. A. Rules for identifying potentially reactive or promiscuous compounds. *J. Med. Chem.* **2012**, *55*, 9763−9772.

(7) Rydberg, P.; Gloriam, D. E.; Zaretzki, J.; Breneman, C.; Olsen, L. SMARTCyp: A 2D method for prediction of cytochrome P450-mediated drug metabolism. *ACS Med. Chem. Lett.* **2010**, *1*, 96−100.

(8) Blake, J. F. Identification and evaluation of molecular properties related to preclinical optimization and clinical fate. *Med. Chem. (Sharjah, United Arab Emirates)* **2005**, *1*, 649−655.

(9) Hann, M.; Hudson, B.; Lewell, X.; Lifely, R.; Miller, L.; Ramsden, N. Strategic Pooling of Compounds for High-Throughput Screening. *J. Chem. Inf. Model.* **1999**, *39*, 897−902.

(10) Pearce, B. C.; Sofia, M. J.; Good, A. C.; Drexler, D. M.; Stock, D. A. An empirical process for the design of high-throughput screening deck filters. *J. Chem. Inf. Model.* **2006**, *46*, 1060−1068.

(11) Brenk, R.; Schipani, A.; James, D.; Krasowski, A.; Gilbert, I. H.; Frearson, J.; Wyatt, P. G. Lessons learnt from assembling screening libraries for drug discovery for neglected diseases. *ChemMedChem* **2008**, *3*, 435−444.

(12) Sushko, I.; Salmina, E.; Potemkin, V. A.; Poda, G.; Tetko, I. V. ToxAlerts: A web server of structural alerts for toxic chemicals and compounds with potential adverse reactions. *J. Chem. Inf. Model.* **2012**, *52*, 2310−2316.

(13) Obrezanova, O.; Csányi, G.; Gola, J. M. R.; Segall, M. D. Gaussian processes: A method for automatic QSAR modeling of ADME properties. *J. Chem. Inf. Model.* **2007**, *47*, 1847−1857.

(14) Boström, J.; Falk, N.; Tyrchan, C. Exploiting personalized information for reagent selection in drug design. *Drug Discovery Today* **2011**, *16*, 181−187.

(15) Zhang, L.; Zhu, H.; Mathiowetz, A.; Gao, H. Deep understanding of structure-solubility relationship for a diverse set of organic compounds using matched molecular pairs. *Bioorg. Med. Chem.* **2011**, *19*, 5763−5770.

(16) Schärfer, C.; Schulz-Gasch, T.; Hert, J.; Heinzerling, L.; Schulz, B.; Inhester, T.; Stahl, M.; Rarey, M. CONFECT: Conformations from an expert collection of torsion patterns. *ChemMedChem* **2013**, *8*, 1690−1700.

(17) Guba, W.; Meyder, A.; Rarey, M.; Hert, J. Torsion Library Reloaded: A New Version of Expert-Derived SMARTS Rules for Assessing Conformations of Small Molecules. *J. Chem. Inf. Model.* **2016**, *56*, 1−5.

(18) Schomburg, K.; Ehrlich, H. C.; Stierand, K.; Rarey, M. From structure diagrams to visual chemical patterns. *J. Chem. Inf. Model.* **2010**, *50*, 1529−1535.

(19) Center for Bioinformatics Hamburg. SMARTSviewServer. https://smartsview.zbh.uni-hamburg.de (accessed Oct 11, 2018).

(20) Schomburg, K. T.; Wetzer, L.; Rarey, M. Interactive design of generic chemical patterns. *Drug Discovery Today* **2013**, *18*, 651−658.

(21) Borgelt, C.; Berthold, M. Mining molecular fragments: finding relevant substructures of molecules. *Proc. IEEE Int. Conf. Data Mining* **2002**, 51−58.

(22) Bietz, S.; Schomburg, K. T.; Hilbig, M.; Rarey, M. Discriminative Chemical Patterns: Automatic and Interactive Design. *J. Chem. Inf. Model.* **2015**, *55*, 1535−1546.

(23) Hopcroft, J. E.; Motwani, R.; Ullman, J. D. *Introduction to Automata Theory, Languages, and Computation*, 2nd ed.; Addison Wesley, 2001; Chapter 4, pp 125−168.

(24) Brüggemann-Klein, A.; Wood, D. One-Unambiguous Regular Languages. *Information and Computation* **1998**, *142*, 182−206.

(25) Hovland, D. The inclusion problem for regular expressions. *Journal of Computer and System Sciences* **2012**, *78*, 1795−1813.

(26) Landrum, G. RDKit: Open Source Cheminformatics. https://www.rdkit.org/ (accessed Sept 26, 2018).

(27) Ehmki, E. S. R.; Schmidt, R.; Ohm, F.; Rarey, M. Comparing Molecular Patterns using the Example of SMARTS: Applications and Filter Collection Analysis. *J. Chem. Inf. Model.* **2019**, DOI: 10.1021/acs.jcim.9b00249.

(28) Urbaczek, S.; Kolodzik, A.; Fischer, J. R.; Lippert, T.; Heuser, S.; Groth, I.; Schulz-Gasch, T.; Rarey, M. NAOMI: On the almost trivial task of reading molecules from different file formats. *J. Chem. Inf. Model.* **2011**, *51*, 3199−3207.

(29) Zamora, A. An Algorithm for Finding the Smallest Set of Smallest Rings. *J. Chem. Inf. Model.* **1976**, *16*, 40−43.

(30) Kolodzik, A.; Urbaczek, S.; Rarey, M. Unique ring families: A chemically meaningful description of molecular ring topologies. *J. Chem. Inf. Model.* **2012**, *52*, 2013−2021.

(31) Bron, C.; Kerbosch, J. Algorithm 457: finding all cliques of an undirected graph. *Commun. ACM* **1973**, *16*, 575−577.

(32) Koch, I. Enumerating all connected maximal common subgraphs in two graphs. *Theor. Comput. Sci.* **2001**, *250*, 1−30.

(33) Ehrlich, H. C.; Rarey, M. Maximum common subgraph isomorphism algorithms and their applications in molecular science: A review. *Wiley Interdiscip. Rev.: Comput. Mol. Sci.* **2011**, *1*, 68−79.

(34) McCreesh, C. Parasols. https://github.com/ciaranm/parasols (accessed Sept 26, 2018).

(35) ZINC15. https://zinc15.docking.org (accessed Aug 20, 2018).

(36) Ehrlich, H. C.; Rarey, M. Systematic benchmark of substructure search in molecular graphs - From Ullmann to VF2. *J. Cheminf.* **2012**, *4*, 13.

DOI: 10.1021/acs.jcim.9b00250
J. Chem. Inf. Model. 2019, 59, 2560−2571

Supporting information for: Comparing Molecular Patterns using the Example of SMARTS: Theory and Algorithms

Robert Schmidt,[†] Emanuel S. R. Ehmki,[†] Farina Ohm,[†,‡] Hans-Christian Ehrlich,[†,¶] Andriy Mashychev,[†,§] and Matthias Rarey[*,†]

[†]*ZBH - Center for Bioinformatics, Bundesstraße 43, 20146 Hamburg, Germany*

[‡]*Current address: Evotec AG - Manfred Eigen Campus, Essener Bogen 7, 22419 Hamburg, Germany*

[¶]*Current address: SAP SE - Innovation Center Potsdam, Konrad-Zuse-Ring 10, 14469 Potsdam, Germany*

[§]*Current address: Titel Media GmbH, Ritterstr. 9, 10969 Berlin, Germany*

E-mail: rarey@zbh.uni-hamburg.de

Phone: +49 (40) 428387351

Contents

Table S1: Index of used expressions and functions occurring in the context of SMARTS recursions

Expression	**Meaning**
basic pattern	The SMARTS pattern without SMARTS recursion
(local) structure	In context of recursion matching: generalization of 'basic pattern' meaning the pattern of the SMARTS recursions attachment node
SMARTS recursion	part of the SMARTS language
SMARTS alternative	(several) SMARTS recursions that are connected via logical 'and'
Similarity of fingerprints	Fingerprint similarity is based on common bits, two fingerprints need at least one bit in common to achieve a similarity value > 0
Positive variant of negated SMARTS recursion	The negated SMARTS recursion without the negation. It is also the SMARTS recursion that is actually matched when matching negated SMARTS recursions
Root node of a SMARTS recursion	The first specified node of a SMARTS recursion. In regular SMARTS matching applications, it is matched to the same atom as its attachment node.
Fingerprint similarity	Fingerprints are considered similar, if they have at least one bit in common.Otherwise they are considered dissimilar.
Environment of a node	The incident bonds and adjacent nodes of a node
Function	**Accessed element of SMARTS pattern**
alternative = ALT(<Pattern>, <node>)	Access the first SMARTS alternative that is part of the node
alt = ALT(<Pattern>, <node>, <index>)	Access the i^{th} SMARTS alternative that is part of the node
recursion = ALT(<alternative>, <index>)	Access the recursions that are part of the SMARTS alternative
node = SUB(<recursion>, <index>)	Access nodes of SMARTS recursion

SI1: Enumeration Scheme for Atomtype-Space

The enumeration of atomtype-space starts with the valence states of the NAOMI chemistry model[S1,S2]. The so-called extended atomtype contains a reference to a valence state and specific values for properties occurring in SMARTS expressions.

For each extended atomtype, the enumeration of the extension starts with the hydrogen property. The corresponding valence states have an annotation of the maximum number of attached hydrogens. All values from zero to that limit are enumerated. The second property enumerated is aromaticity. The SMARTS language only supports the binary predicate of aromaticity, within a certain pattern aromaticity can be defined on a more specific level. In ring systems nodes could have two or three incident ring bonds. A non-recursive SMARTS node expression is not able to distinct atoms with two aromatic bonds out of three incident ring bonds from those having three aromatic bonds. But the structure of a pattern does distinct those atoms. For each extended atomtype, the number of incident aromatic bonds is set to zero. If the corresponding valence state can be aromatic, then the number of incident aromatic bonds is enumerated starting at two up to the degree minus the number of attached hydrogens.

The hybridization state is not a regular property in SMARTS, however several SMARTS parsers have a corresponding extension. The NAOMI chemistry model assigns possible hybridization states to atomtypes which can then be enumerated as well.

Next, the SMARTS ring properties are enumerated. The SMARTS definition is based on SSSR ring bases[S3] (Smallest set of smallest rings). Since these are not unique, we use the URF[S4] (Unique ring families) model instead. First the number of unique ring families (URF) an atom is part of is enumerated. For each non-aromatic extended atomtype the enumeration of possible values of R starts at zero and ends at five. The highest number five is meant as a catch all state summarizing all values greater or equal five. After that, the number of incident ring bonds is enumerated. For extended atomtypes that are not part of a ring, this value is set to zero. For all other nodes, the enumeration starts with the

number of incident aromatic bonds or two and ends when it reaches twice the number of ring families. Since hydrogen atoms cannot be part of rings, the number of incident ring bonds is limited to by the total number of bonds minus the number of attached hydrogen. Finally, the minimum URF ring size for each atom is added. For extended atomtypes that are not part of any ring this value is set to zero. For all other states, the enumeration starts at three. For aromatic extended atomtypes, the enumeration stops at nine, with the nine as a catch all state. In all other cases, the enumeration stops with a catch all state at a ring size of 25.

This enumeration results in 20,565 extended atomtypes for the whole NAOMI chemistry model. Unfortunately metals are so far mostly not covered by the model. Note that the ring count and ring size enumeration results in the limitation that SMARTS expressions beyond these limits cannot be distinguished. The *reduced fingerprint* as used for pattern similarity is enumerated in the same way. The only difference is that the ring properties are far more limited. In this case one is the catch all state for the 'R' property and three is the catch all state for the 'r' property. The enumeration applying those limits results in 524 extended atomtypes.

Extended Atomtype Assignment for Atoms

The extended atomtypes are designed to cover the complete atomtype-space. Therefore it is possible to assign each atom in a molecule exactly one extended atomtype. For an atom, all properties covered by the extended atomtype are known, especially the valence state. The following relevant properties are extracted:

- number of adjacent hydrogen atoms
- number of incident aromatic bonds
- number of incident ring bonds
- number of rings (URF) the atom is part of

- hybridization

Note that if the extracted properties are beyond the enumeration limits, they are reset to the respective catch-all values and a warning is produced. Finally the generated extended atomtype is searched within the enumerated list to assign a unique atomtype id.

SI2: A Detailed Overview About Node Fingerprint Pruning

For the purpose of node fingerprint pruning several properties of the node's environment are taken into account.

Table S2: All environment properties of a SMARTS node collected prior to fingerprint pruning.

1. degree
2. valence
3. number of explicit adjacent hydrogen atoms
4. number of possible adjacent hydrogen atoms
5. number of incident single bonds that are not incident to hydrogen atoms
6. number of incident bonds with a minimum valence of two
7. number of incident triple bonds
8. number of incident bonds that can be aromatic bonds
9. number of incident explicit aromatic bonds
10. number of bonds that can be aromatic bonds and can be incident to hydrogen atoms
11. number of explicit ring bonds
12. number of explicit acyclic bonds
13. number of possible single bonds
14. number of possible double bonds
15. number of possible double bonds that are explicitly not aromatic
16. number of possible triple bonds
17. number of incident acyclic bonds that may be incident to hydrogen atoms
18. number of incident non-aromatic bonds to incident aromatic neighbors
19. number of incident non-aromatic bonds to incident aromatic neighbors or hydrogen atoms

Table S2 lists all properties that are collected and used for fingerprint pruning. Using these properties each possible extended atomtype of a node fingerprint is reconsidered. Table S3 lists all properties updated if there are less incident bonds that can be aromatic than expected by the extended atomtype. Finally, all criteria, listed in Table S2 are used to evaluate the compatibility of the SMARTS node environment and the extended atomtype.

Table S3: Environmental properties that are updated during extended atomtype compatibility evaluation if there are less aromatic bonds in the environment that required by the extended atomtype.

1. The degree is increased by the number of aromatic bonds that are required by the extended atomtype but not part of the environment.
2. The valence is increased by the number of aromatic bonds that are required by the extended atomtype but not part of the environment. Note: this is a lower bound valence estimation.
3. There can be neighbors in the environment that can be hydrogen as well as nodes connected via an edge that can be aromatic. In such a situation the number of optional adjacent hydrogen is decreased by the minimum of the number of those neighbors and the number of missing aromatic bonds.

Table S4: The list of all applied criteria for SMARTS node environment to extended atomtype compatibility

1. The degree in the environment must not be larger than the number of bonds of the extended atomtype.
2. The observed minimum valence in the environment must not be larger than the valence of the extended atomtype.
3. The number of explicit adjacent hydrogen atoms in the environment must not be larger than in the extended atomtype.

4. The difference of the valence of the extended atomtype and the environment must not be smaller than twice the number of double bonds that are part of the extended atomtype but not in the environment.
5. The difference of the valence of the extended atomtype and the environment must not be smaller than trice the number of triple bonds that are part of the extended atomtype but not in the environment.
6. The number of single, double and triple bonds in the environment must not be larger than those in the extended atomtype.
7. The difference between valence and degree must not be smaller in the extended atomtype than in the node's environment.
8. The number of adjacent hydrogens must not be smaller in the extended atomtype than those in the environment.
9. There must be enough free single bonds in the environment for all adjacent hydrogen atoms of the extended atomtype.
10. The number of ring bonds must not be smaller in the extended atomtype than those in the environment.
11. The number of acyclic bonds must not be smaller in the extended atomtype than those in the environment.
12. The number of double and triple bonds in the extended atomtype must not be smaller than the number of explicit non single bonds in the environment.
13. The difference in the number of adjacent hydrogen atoms in the extended atomtype and the environment must not be larger than the difference of the number of bonds of the extended atomtype and the degree of the environment.
14. The number of incident aromatic bonds must not be smaller in the extended atomtype than those in the environment.
15. If an extended atomtype has sp2 hybridization because of adjacent aromatic systems, there must be a free single bond that can be the bond to the aromatic system.

SI3: Preparation of SMARTS Recursion for Fingerprint Pruning

This section describes all necessary steps to generate a unique fingerprint for nodes with SMARTS recursion:

1. Generation of node and edge fingerprints as for the basic pattern
2. Removal of redundant SMARTS recursion that are more generic than their local structure.
3. Evaluation of the node expression logic and grouping of SMARTS recursion into logical alternatives (transformation into disjunctive normal form, DNF).
4. Removal of redundant SMARTS recursions within alternatives.
5. Removal of redundant alternatives from the generated DNF in SMARTS node expressions.
6. Update of node fingerprints including all information provided by SMARTS alternatives.
7. Final pruning of all fingerprints using the updated fingerprints of nodes with SMARTS alternatives.

For each SMARTS recursion, node and edge fingerprints are generated and pruned in the same way as for the basic pattern. The SMARTS recursion is processed to create a fingerprint which can be used to prune the fingerprint of the attachment node. The overall process is iterated until convergence is reached.

Like any other SMARTS property, recursions can be logically combined with $\wedge$ and $\vee$. We define a SMARTS *query* as SMARTS recursion or any non-recursive SMARTS property. For nodes with SMARTS recursion, the SMARTS node expression is reformulated into

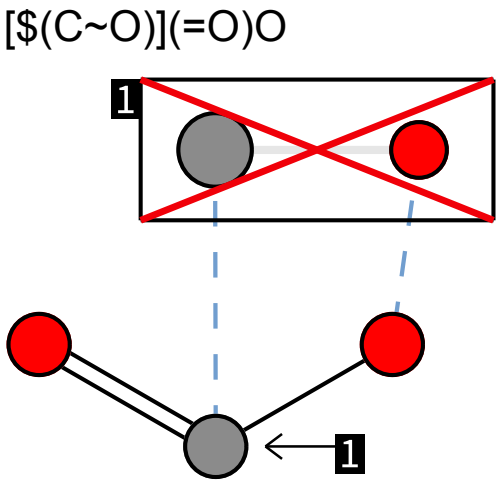


Figure S1: Detection of a redundant SMARTS recursion at its attachment node. C∼O is a superset its local structure C(=O)O.

disjunctive normal form (DNF). We then define *alternatives* as logical ∧ connected (conjunction) SMARTS queries and combining these in turn by logical ∨ (disjunction). The DNF structure of the SMARTS expression is crucial for the following processing and matching steps.

Similar to SMARTS nodes in their environment, there can be redundancy in SMARTS recursion. The following pruning procedure detects three kinds of redundancy within SMARTS recursion. The first kind of redundancy is handled for each SMARTS recursion, the two following kinds of redundancy are related to logical combinations of SMARTS recursion. They are handled in section .

1. Generic SMARTS recursions that are redundant within the local structure of their attachment node. They are entirely removed from the matching procedure.

2. Generic SMARTS recursions that are redundant within their alternatives. They are removed from the alternative.

3. Alternatives of SMARTS recursion that are more specific than others at the same attachment node. They are removed from the SMARTS node expression.

The first redundancy test for SMARTS recursion verifies that a SMARTS recursion is not more generic than the local structure of its attachment node. Such generic recursions yield no further information for the patterns P representation in chemical space $CS(P)$. Thus, they can be removed from the pattern description without loss of specificity. Figure S1 shows an example of a redundant SMARTS recursion. If a positive SMARTS recursion describes exactly one node, it is always considered redundant because the first node of any positive SMARTS recursion is fully exploited during fingerprint generation of the basic pattern. Negative SMARTS recursions are never removed from the pattern. If they are more generic than their attachment node's environment, they indicate an error in their alternative.

Pruning of Recursion Alternatives

The following two redundancy tests are designed to remove any unnecessary SMARTS recursion from the pattern. To do so we have to compare SMARTS recursions to each other which can be performed with a recursive call of the SMARTScompare algorithm. Note that SMARTS recursions can contain further SMARTS recursions causing further recursive calls.

The first test removes redundancy within logical and-expressions, i.e. within alternatives, the last test removes redundancies in logical or-expressions, i.e. between alternatives.

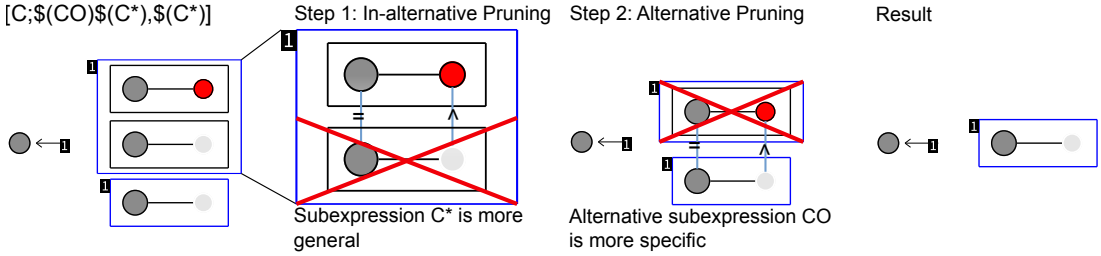


Figure S2: Two steps of the recursion alternative pruning. The first step detects and removes redundancies within alternatives, the second step between alternatives. In the first step, the more general recursion C∗ is removed from $(CO),$(C∗), reducing it to $(CO). In the second step, the more specific alternative $(CO) is removed. This leaves the final pattern [C;$(C∗)]

To remove redundancy within alternatives, all SMARTS recursions of the alternative are compared against each other. If any SMARTS recursion is more generic, than any other

SMARTS recursion of the same alternative, it is considered redundant and removed from the alternative. Within an alternative, each query has to be fulfilled for compatibility. This implies that more generic SMARTS recursions yield no information, if more specific SMARTS recursions are evaluated as well.

Let A_P and A_Q be two alternatives of SMARTS recursion at the node n. To remove redundancy between A_P and A_Q all SMARTS recursions of $S_{Pi} = SUB(A_P, i)$ and $S_{Qj} = SUB(A_Q)$ are compared with each other. A_Q is more specific than A_P if and only if for each SMARTS recursion S_{Pi} there is a more specific SMARTS recursion S_{Qj}. If A_Q is more specific than A_P, then A_Q is considered redundant and removed from the node's alternatives.

If any negated SMARTS recursion is more generic than the attachment node's environment, then its alternative will never match which is considered as an error. The same situation occurs, if any other positive SMARTS recursion in the same alternative is more specific than the negated one.

After the pruning steps, a fingerprint is assigned to each alternative combining the information that is encoded in all of its SMARTS recursions' root nodes.

The fingerprint can then be pruned further exploiting information from the total number of unique neighbors and incident nodes of the attachment node. The number of unique neighbors and incident bonds is calculated mapping compatible nodes and edges onto each other. For this procedure, nodes and edges are compatible, if they share a chemical state. Figure S3 shows the computation of a node environment considering alternatives of SMARTS recursion.

Finally the attachment node's fingerprint is replaced with the logical disjunction of fingerprints from all its alternatives'.

Since a node's fingerprint potentially influences the fingerprints of its neighbor nodes, an iterative update process is required. The pruning criteria already discussed for the basic pattern as listed in Table 3 and in Table 4 are applied. This time, SMARTS recursion is included into the update procedure. If the fingerprint of a node with recursion, SMARTS

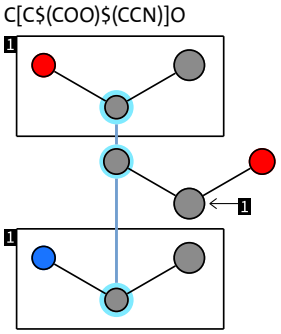


Figure S3: The computation of a node environment considering alternatives of SMARTS recursion. Compatible nodes in the environment of the attachment node and the root nodes of the recursions are combined. They are highlighted in blue. In the given example the environment consists of two nodes, both are connected to the node via a single bond. Without the reduction, the environment would consist of four single bonds and four neighboring nodes.

recursion is pruned, the update is propagated into the SMARTS recursion.

SI4: Explicit Handling of Alternatives While Matching SMARTS Recursions

This section focuses on the necessary aspects to handle the logic between recursive SMARTS expressions during comparison. First of all, recursive SMARTS expressions are compared to the local structure of the other pattern. Then they are compared to each other. Further constraints for node mapping can be derived for subset search and from the use of negated recursive expressions. These will be summarized below.

For each node with SMARTS recursion SMARTScompare calculates the compatibility of alternatives of SMARTS recursions to the local structure and the compatibility to alternatives of SMARTS recursions from the other pattern.

Consider two patterns P and Q and the compatibility computation of the nodes $n \in P$ and $m \in Q$. For both nodes let there be exactly one SMARTS alternative and SMARTS recursion $A_P = ALT(P, n)$, $Sn = SUB(A_P, 0)$ specifying n and $A_Q = ALT(Q, m)$, $Sm = SUB(A_Q, 0)$ specifying m.

The root nodes of the recursions are $Sn_0 = SUB(Sn, 0)$ and $Sm_0 = SUB(Sm, 0)$. In this example, both patterns have exactly one recursion to simplify expressions for the predefined mappings. To compute the compatibility of a SMARTS alternative A_P to either the local structure or another SMARTS recursion, the root node Sn_0 of the recursion has a defined mapping partner. It is defined by the mapping (n, m). The partner is either m, the mapping partner of n or Sm_0, the root node of the recursion Sm at m. The pre-defined mapping partners are used as initial mapping for MCS calculations.
The overall method for recursion handling distinguishes two different types, mapping SMARTS alternatives to the local structure or mapping them to other SMARTS alternatives. Furthermore the two operation modes subset search and similarity search are distinguished.

Before any recursive compatibility computation, the fingerprints of the nodes n and m are tested for compatibility. Using the current mapping for initialization, the MCS is applied

as follows:

Matching a SMARTS Alternative A_P of Pattern P Against the Local Structure of m

In this example the local structure of m is the basic pattern Q, since there are no nested SMARTS recursions.

Each alternative whose fingerprint is similar to m's fingerprint is matched against Q, using the nodes Sn_0 and m as a initial mapping. This is performed for each alternative, even if a compatible alternative is already found.

This procedure starts a recursive matching of the patterns Sn and Q with an initial mapping of Sn_0 to m. If m also has SMARTS recursions, then for this mapped pair, Sn_0 and m, this method could be applied recursively but this is explicitly handled as the second case.

Matching of SMARTS Alternatives

To test the compatibility of SMARTS nodes n and m based on their SMARTS alternatives, the corresponding fingerprints have to be similar. Furthermore, for each SMARTS recursion of an alternative, there must be a compatible SMARTS recursion of the other alternative.

For nodes with SMARTS alternatives, the matching procedure enforces that all SMARTS recursions of at least one SMARTS alternative are mappable to the local structure or another SMARTS recursion of the same alternative.

If this is not the case for any node n with SMARTS alternatives, then n and m are incompatible. In general, SMARTS are considered compatible, if at least one node can be mapped. If SMARTS recursions are compared, then there is a fixed mapping of initial nodes. Thus, at least the two initially mapped nodes have to be compatible. The search for more specific patterns and the inclusion of negative SMARTS recursion introduce further constraints for compatibility.

Constraints for Searching more Specific Patterns

Let $A_P = ALT(P, n, i)$ and $A_Q = ALT(Q, m, j)$ be the alternatives of SMARTS recursion at the nodes n and m in the patterns P and Q. Note that alternatives of SMARTS recursion does not necessarily contain a recursion. For example, in SMARTS node expressions like [N,$(CN)] there are two alternatives, the first does not contain a recursion.

If patterns Q is tested to be more specific than pattern P, then there are three options for compatibility of alternatives $A_P = ALT(P, n, i)$ at n.

1. The alternative A_P is not considered at all. This is allowed since P is tested to be more generic.
2. The alternative A_P is more generic than the local structure of Q at m
3. The alternative A_P is more generic than any alternative $A_Q = ALT(Q, m, j)$ at m

For compatibility of n and m there has to be at least one alternative at n that is more generic than either the local structure or an alternative at m.

Matching Alternatives Against the Local Structure

Let $A_P = ALT(P, n)$ be an alternative of pattern P. A_P is more generic than the local structure of $m \in Q$ if and only if the fingerprints are compatible and each SMARTS recursion $S_n = SUB(A_P, i)$ of A_P is more generic than the structure of Q at m. In the following recursive call alternatives $A_Q = ALT(Q, m, j)$ at m, are automatically assumed to be compatible to Sn_0 if existent. Their compatibility to the alternatives $A_P = ALT(P, n, i)$ is individually handled.

If alternative $A_Q = ALT(Q, m)$ is mapped onto the structure of P at n, then A_Q is compatible if and only if each of its recursions $Sm = SUB(A_Q, j)$ is more specific than the local structure at n. Since P is tested to be more generic, $Sm = SUB(A_Q, j)$ is already more specific than the local structure at n, if its root node is more specific than n.

Matching Alternatives of SMARTS Recursions

Alternative $A_Q = ALT(Q, m)$ at m is more specific than $A_P = ALT(P, n)$ at n if and only if for each SMARTS recursion $Sn = SUB(A_P, i)$ there is a SMARTS recursion $Sm = SUB(A_Q, j)$ such that Sm is more specific than Sn.

To summarize the compatibility of nodes n, m with SMARTS alternatives if pattern P is tested to be more generic than pattern Q there are three cases to distinguish.

Node $n \in P$ has SMARTS alternatives but $m \in Q$ does not

At least one alternative $A_P = ALT(P, n)$ has to be more generic than the local structure at m.

Node $m \in Q$ has SMARTS alternatives but $n \in P$ does not

Each alternative $A_Q = ALT(Q, m)$ has to be more specific than the local structure at n.

Both Nodes $n \in P$ and $m \in Q$ have SMARTS Alternatives A_P and A_Q

Each alternative A_Q has to be more specific than the local structure at n or any alternative A_P. The combined fingerprint of the alternatives A_P that are more generic than the local structure at m or any alternative A_Q has to be more generic than the node fingerprint of m.

Constraints for Negative SMARTS recursion

Negated SMARTS recursion needs special treatment during the algorithm. They describe localized structures that are explicitly excluded from the pattern. If negated SMARTS recursions are compared with the local structure or positive SMARTS recursions, it is tested whether they are more specific than the positive variant of the negated SMARTS recursion. Therefore, the comparison mode for the next recursive call is adjusted. The calculation answers the question, whether the other pattern contains any structures that are explicitly excluded from an alternative of a SMARTS node. If the positive variant of a negated

SMARTS recursion is fully mappable to the local structure or any positive SMARTS recursion, it specifies an incompatible alternative. If negated SMARTS recursions are compared, the MCS of their positive variants is calculated without any adjustments.

In the following the compatibility of the nodes $n \in P$ and $m \in Q$ is evaluated. There is an alternative $A_P = ALT(P, n)$ containing a negative SMARTS recursion $Sn_{NEG} = SUB(A_P, 0)$. Let A_P be relevant for the compatibility of n and m. Then there are two constraints that have to be fulfilled otherwise n and m are not mappable:

General constraint: Sn_{NEG} is not fully mappable to the local structure of Q at m.

m **has SMARTS recursion:** For each alternative $A_Q = ALT(Q, m, j)$ at $m \in Q$ that is compatible to A_P, each positive SMARTS recursion $Sm = SUB(A_Q, m, j)$ is not fully mappable to Sn_{NEG}.

Figure S4 depicts the two mentioned cases. Once the negated recursion is mappable to the local structure and once it is mappable to another recursion. In both cases nodes are not compatible. There are two additional constraints for the subset search. If P is tested to be more generic than Q one of the following two constraints has to be fulfilled:

1. The fingerprint of Sn_{NEG}'s root node is dissimilar A_Q's fingerprint

2. For each alternative $A_Q = ALT(Q, m, j)$ that is compatible to A_P, there is a negated SMARTS recursion $Sm_{NEG} = SUB(A_Q, j)$ which is more specific than Sn_{NEG}.
   Note: A negated SMARTS recursion Sn_{NEG} is more generic than the negated SMARTS recursion Sm_{NEG}, if and only if the positive variant of Sn_{NEG} is more specific than the positive variant of Sm_{NEG}.

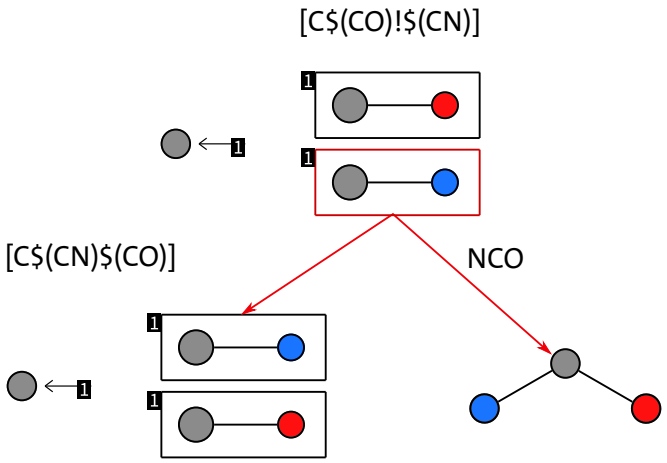


Figure S4: The two types of negative recursion incompatibility. In the first comparison, the negative SMARTS recursion is incompatible to a positive SMARTS recursion, thus the alternatives and nodes are incompatible. In the second comparison, the negative SMARTS recursion is incompatible to the structure and the nodes are incompatible as well.

SI5: SMARTS Recursion During Result Compilation

After determining compatibility of the patterns P and Q, the final mapping result is constructed. The construction starts with the two basic patterns. For each mapped node pair (n, m) with SMARTS recursion, the mapping is extended with the localized mapping for whole alternatives at n and m. The MCS results consist of all mappings of maximum size between the patterns and all SMARTS recursions. Each independent mapping construction selects the MCS mapping result that covers the maximum number of nodes. Note that the pruning of SMARTS recursion alternatives is mandatory to achieve a reasonable mapping at this step.

Suppose Q is tested to be more specific than P. Consider the SMARTS alternative $A_{P1} = ALT(P, n, 1)$ and $A_{P2} = ALT(P, n, 2)$ and the SMARTS recursions $Sn_1 = ALT(A_{P1}, 1)$ and $Sn_2 = ALT(A_{P2}, 1)$ at n. Both SMARTS recursions A_{P1}, A_{P2} are in different alternatives and they are mapped onto the structure of Q at m. In this situation, the alternative A_{Pi} at n of P is chosen that is covered best and gives the maximum score. Since the alternatives are mapped to the local structure, it is guaranteed that the subset relation is obeyed.

Now let there be two SMARTS recursions in different alternatives $A_{Q1} = ALT(Q, 1), Sm_1 = SUB(A_{Q1})$ and $A_{Q2} = ALT(Q, 2), Sm_2 = SUB(A_{Q2})$ at m and let there be one SMARTS recursion $A_P = ALT(P, 1), Sn = SUB(A_P)$ at n, which is incompatible to the structure of Q. A molecule is matched by a pattern with SMARTS recursion, if at least one alternative is compatible. To consider this fact for SMARTS subset search, the alternative mapping Sn to Sm_1 that covers the minimum number of nodes is selected. The final result of the mapping calculation is used to determine the covered nodes of both patterns and to calculate the similarity score and subset relation.

The fact that only one node mapping is calculated may not reflect the intuition of pattern similarity well. This becomes relevant if there are several alternatives of SMARTS recursion in both patterns. In such a situation intuitive similarities would consider several mappings

of the SMARTS alternatives. Computing and averaging them seems inappropriate in terms of computational complexity. To find a meaningful compromise between averaging all node mappings and only selecting the best, an additional mapping is calculated. For this mapping the alternatives of SMARTS recursion are chosen that result in the minimum score. The final score is the mean of the scores for the highest and lowest similarity SMARTS node mapping.

Furthermore, for the mapping construction, special constraints for the subset search are applied. It is tested that each node of the supposed more generic pattern is covered. If an alternative with a negated SMARTS recursion is mapped, then the negated SMARTS recursion is not necessarily fully covered. If negated SMARTS recursions are mapped, the SMARTS recursion with more nodes is likely to be more generic.

The final similarity score is the normalized sum over all mapped node similarities. By default the sum is normalized with the effective number of nodes from both patterns inspired by the Tanimoto coefficient. Equation 1 shows the similarity, whereas *numNodes* is a function returning the minimum number of nodes of a pattern P considering all SMARTS alternatives participating in M and those providing in the minimum number of nodes otherwise.

$$\mathrm{Sim}(P, Q, M) = \frac{\sum_{(n,m)\in M} \mathrm{Sim}(n, m)}{\mathrm{numNodes}(P, M) + \mathrm{numNodes}(Q, M) - |M|} \tag{1}$$

Note that this is just one option for normalizing scores. It is easy to envision alternatives, for example to create asymmetric similarity scores following Tversky[S5].

SI6: Pattern Comparison in RDKit

In RDKit[S6], SMARTS and SMILES are both parsed into the same internal Molecule datastructure. In principle there is no difference between patterns and molecules. The SMARTS matching methods support matching of molecules as well as matching of SMARTS. With additional initialization of molecular properties, one can interpret a SMARTS pattern as a molecule and match another SMARTS pattern against it. This way, SMILES properties like the hydrogen properties, charges and the incident bonds are transferred into the matching. Other properties like the ring properties are not transferred into the matching. This kind of matching does not support any disjunction or SMARTS recursion on the matched SMARTS except for wildcard matchings. The matching result does not necessarily correspond to valid subset relations. For example, since [#8]=[#6] is matched by [#6X3] but the carbon node does not necessarily have three edges like in carbon dioxide [#8]=[#6]=[#8]. Using the specialized method to match queries there is no transfer of properties possible. An example for a property transfer is a nitrogen in a pattern with four single bonds that is detected as charged. In principle the query comparison approach is able to handle disjunctions and negation of properties, but often it fails the explicit subset relations, meaning the matched pattern is not more specific as it should be. [#6] matches [!#7] and [#6,#7] although the matched patterns are more generic than the query. Using the query comparison SMARTS recursion cannot be handled. In conclusion, RDKit is able to detect subset relations of simple patterns that are written as pure conjunctions without any negation if no property transfer is needed. The more general pattern comparison problem allowing simple disjunctions or negation of simple properties, the result does not guarantee finding more specific patterns.

References

(S1) Urbaczek, S.; Kolodzik, A.; Fischer, J. R.; Lippert, T.; Heuser, S.; Groth, I.; Schulz-Gasch, T.; Rarey, M. NAOMI: On the almost trivial task of reading molecules from different file formats. *J. Chem. Inf. Model.* **2011**, *51*, 3199–3207.

(S2) Urbaczek, S.; Kolodzik, A.; Rarey, M. The valence state combination model: A generic framework for handling tautomers and protonation states. *J. Chem. Inf. Model.* **2014**, *54*, 756–766.

(S3) Zamora, A. An Algorithm for Finding the Smallest Set of Smallest Rings. *J. Chem. Inf. Comput. Sci.* **1976**, *16*, 40–43.

(S4) Kolodzik, A.; Urbaczek, S.; Rarey, M. Unique ring families: A chemically meaningful description of molecular ring topologies. *J. Chem. Inf. Model.* **2012**, *52*, 2013–2021.

(S5) Tverky, A. Features of Similarity. *Psychological Review* **1977**, *84*, 327 – 352.

(S6) Landrum, G. RDKit: Open Source Cheminformatics. `https://www.rdkit.org/`, Accessed on: 2018-09-26.

Comparing Molecular Patterns Using the Example of SMARTS: Applications and Filter Collection Analysis

[D4] Ehmki, Emanuel S.R. u. a.: „Comparing Molecular Patterns Using the Example of SMARTS: Applications and Filter Collection Analysis“. In: *J. Chem. Inf. Model.* 59.6 (2019), S. 2572–2586. DOI: 10.1021/acs.jcim.9b00249

http://pubs.acs.org/articlesonrequest/AOR-9hFRXRpgspHNz8TMUUzz

JCIM JOURNAL OF CHEMICAL INFORMATION AND MODELING

Article

Cite This: J. Chem. Inf. Model. 2019, 59, 2572−2586

pubs.acs.org/jcim

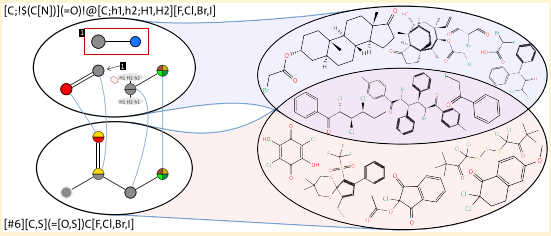
Comparing Molecular Patterns Using the Example of SMARTS: Applications and Filter Collection Analysis

Emanuel S. R. Ehmki, Robert Schmidt, Farina Ohm,† and Matthias Rarey*

ZBH - Center for Bioinformatics, Bundesstraße 43, 20146 Hamburg, Germany

Supporting Information

ABSTRACT: In a recent work, an algorithm to compare chemical patterns, written for example in SMARTS, was presented. This algorithm, called SMARTScompare, is able to assess the identity, subset relation, and similarity of a pair of patterns. Here we used an implementation of SMARTScompare to analyze SMARTS filter sets that were published in the context of, for example, high-throughput screening. We found that the difference in intentions with which the filter sets were designed is mirrored in the similarity values we calculated. The analysis revealed which patterns from one filter set are covered by filters from another set. In one case it became obvious that a filter set is more or less completely covered by another. Furthermore, we analyzed pattern hierarchies for consistency, and we propose a method to remove redundant patterns. SMARTScompare together with SMARTScompareView equips users with powerful methods to visualize, compare, and focus their filter sets.



INTRODUCTION

The size of compound databases has been increasing steadily.[1,2] Equally, the need for sophisticated search and filter methods has been on the rise, too. Most queries are designed to retrieve or exclude classes of molecules or specific types of chemistry. Therefore, many queries are designed as molecular patterns that describe a specific substructure or a group of substructures. In practice, these molecular patterns are often encoded in, for example, SMILES[3] Arbitrary Target Specification (SMARTS)[4] or SYBYL Line Notation (SLN).[5]

Sets of molecular patterns can be used to model unwanted behavior in many fields of application, e.g., high-throughput screening (HTS)[6] or toxicity.[7] Frequently they are used as filters to exclude molecules with unwanted groups before expensive experimental screening starts. Compounds that are matched by any pattern have been shown to express unwanted behavior and can be flagged for further observation or removed right away. Several structural filter sets, mostly in the context of experimental screening, have been published.[2,6−13] With the arrival of large collections of molecular patterns, a whole new set of problems emerged as well. Molecular patterns can be highly complex, and design implications cannot always be foreseen from the start. Therefore, good tools aiding scientists in visualization and browsing,[14,15] designing,[16,17] and comparing molecular patterns are needed.

Considering the importance of SMARTS for the field of cheminformatics, relatively few approaches to compare two SMARTS pattern exist to date. The most straightforward strategy— visually inspecting two patterns—is extremely laborious as pattern sets are growing. A more automatable approach is the comparison of sets of molecules that are matched by two patterns. This makes the analysis of the set relation of two patterns possible. Nonetheless, this approach can only deny subset relation. It cannot prospectively classify one pattern as a sub- or superset of the other. Besides the large computational burden, this makes the approach extremely dependent on the selection of molecules that are used to analyze the patterns. The third and also most algorithmic approach is implemented in part in the Rational Design Kit (RDKit).[18] It enables a node-based comparison of two molecular patterns in the form of SMARTS strings within the boundaries of some restrictions. Roughly sketched, the pattern comparison is possible in cases where the semantics of the SMARTS language are compatible. In cases where the same chemical state (e.g., an sp^3-hybridized carbon) is encoded differently within the SMARTS pattern, the pattern nodes can no longer be mapped correctly. Our newly designed algorithm, called SMARTScompare,[19] is independent of the input form because it maps SMARTS semantics into the general pattern space. This enables us to determine the chemical state of a pattern node, making the comparison independent of the SMARTS string representation. Employing a maximum common subgraph (MCS) algorithm that includes the handling of pattern recursion eventually enables the comparison of SMARTS patterns for nearly all practically relevant scenarios.

For the analysis of structural filters, the possibility to compare chemical patterns with an algorithmic approach is highly desirable. Common structures that are part of several

Received: March 22, 2019
Published: May 24, 2019

DOI: 10.1021/acs.jcim.9b00249
J. Chem. Inf. Model. 2019, 59, 2572−2586

structural filter sets are more likely to be relevant than patterns occurring in exactly one. With hundreds of molecular patterns in a single filter collection, it is challenging to detect patterns related to the same or similar chemical features. When the pattern notations for the same kind of chemistry are different, the analysis of a complete set of patterns becomes an extremely laborious task. Comparing patterns designed by several different scientists for the same purpose can give many insights, sharpen the pattern collection, and help to prevent errors or gaps in the description of substructures of interest. Furthermore, analyses like the one conducted by Capuzzi et al.[20] on PAINS[8] do not have to rely on the molecules that are matched by molecular patterns. Instead, they can be performed directly on the level of molecular patterns, allowing more precise and reliable results.

To be able to conduct comprehensive analyses of molecular patterns, we developed a novel analytic approach.[19] Schomburg et al.[14] presented an intuitive visualization concept for SMARTS patterns. We extended their approach to aid the user in better understanding chemical patterns and structural filter sets. The method is customized to SMARTS and handles nearly all frequently used language elements. With runtimes in the millisecond range, the algorithm can be used interactively even on larger filter collections with a few hundred to thousands of patterns.

THEORY

For a detailed discussion and the proper mathematical formalism, we refer the reader to the companion paper.[19] The following section discusses only briefly the important theoretical aspects that were presented there. Additionally, implications for the implementation and application of the algorithm are discussed. For consistency, the same notation is used.

Theoretical Aspects of SMARTS Pattern Comparison. The general assumption is that a chemical pattern describes a potentially infinite subset of molecules in the chemical space (CS). Therefore, we can approximate the subset relation of two patterns P_1 and P_2 on the basis of how well their subsets of the chemical space, $CS(P_1)$ and $CS(P_2)$, overlap. To get an exhaustive description of the chemical space, we introduce two new description systems. First, the atomtype space represents possible states that atoms can take on in chemical molecules. Second, equivalent to the atomtype space, the bondtype space describes possible states of bonds in molecules. In a graph representation of a molecular pattern, nodes are described via a set of atomtypes and edges via a set of bondtypes. Given that approach, we conclude that if patterns are identical, their sets of matched molecules are identical, too. With additional restrictions applied, the same is valid for the opposite direction. What has been detailed for subset relations can be easily extended to allow similarity analysis.

Since we model chemical information as a set of atomtypes, the similarity of two patterns can be estimated by analyzing the overlap of atomtypes of computed node mappings. Prior to the comparison step, SMARTS string expressions are translated into their graph representations. Each node representation in the SMARTS string is converted to one graph node.

To be able to compare two SMARTS graph structures, each node or edge is described by a fingerprint of atomtypes or bondtypes, respectively. The node and edge fingerprints are of constant size. Each bit represents exactly one atomtype, and the overall size of the fingerprint is determined by the number of atomtypes defined in the chemistry model that is used.

When two SMARTS graphs are compared, first an induced maximum common connected subgraph is computed. The compatibility of nodes depends on the comparison mode, e.g., similarity or set relation. The similarity score of two SMARTS expressions is then calculated on the basis of the fingerprint similarity of compatible nodes. The following sections discuss relevant aspects that have to be considered when designing an implementation of the presented algorithm.[19] The focus lies on handling of aromaticity, wildcards, and ring properties and fine-tuning of similarity calculations.

Implementation-Specific SMARTS Handling. *Wildcards.* Wildcards in SMARTS like "*", "A", and "a" match several elements. In NAOMI,[21] the SMARTS matching supports several options to handle hydrogens during SMARTS matching:

- Hydrogens are not matched at all.
- Wildcards do not match hydrogen atoms.
- The wildcard * matches hydrogen as well.

The second option is enabled by default. Thus, the patterns [#1] and * are considered dissimilar by default.

Ring Size Property. The SMARTS ring count property "R" as defined by Daylight Chemical Information Systems, Inc.[4] describes the number of smallest set of smallest rings (SSSR)[22] rings of which an atom is a part. For the fingerprint approach, however, it is important that the "R<n>" property is unique. Unique ring families (URFs) describe exactly the number of smallest rings of which an atom is a part. Therefore, the NAOMI library describes R′ as the number of URFs[23,24] of which an atom is a part.

In order to describe all possible states, the number of rings and the smallest ring size have to be enumerated up to a constant value. We therefore limit the maximum number of rings to 4 and the maximum smallest ring size to 25. If the values are larger, they can be handled but can no longer be distinguished.

It should be noted that the ring properties increase the size of the fingerprint substantially. Therefore, special care has to be taken if fingerprints are converted to similarity values.

Similarity Calculation. The most simple way to express the similarity of two patterns is to count their matching nodes as determined by the MCS approach. Although this measure might be enough for a rough estimate, it does not do justice to the complexity of the pattern matching problem. A more meaningful value can be achieved if the similarity between two nodes in SMARTS patterns is calculated via the atomtype fingerprints associated with them. Suppose that P_1 and P_2 are the pattern graph structures and M is the matching. Furthermore, suppose that u and v are nodes of graphs, where $u \in P_1$ and $v \in P_2$. The similarity of P_1 and P_2 can then approximately be calculated as shown in eq 1:

$$S(P_1, P_2, M) = \frac{\sum_{(u,v)\in M} S(u, v)}{\text{normalization term}} \quad (1)$$

Possible normalization modes are listed in Table 1a.

Calculating similarity on fingerprints is a well-established strategy in cheminformatics. In SMARTScompare one can select from several comparison modes in order to achieve an outcome that is most fitting for the problem posed (see Table 1b). Aside from the MCS similarity score, all options work with the fingerprint that is reduced or weighted upon

DOI: 10.1021/acs.jcim.9b00249
J. Chem. Inf. Model. 2019, 59, 2572−2586

Table 1. Possible Options for the Normalization and Scoring Level Modes Implemented in SMARTScompare: Each Section Describes a Type of Option That Can Be Selected When the Comparison Mode *Similarity* Is Selected

(a) Normalization Modes
sum of nodes—number of matched nodes
max. number of nodes in both patterns
mean number of nodes in both patterns
min. number of nodes in both patterns
number of nodes in the first pattern
number of nodes in the second pattern
(b) Scoring Level Modes
MCS
elements fingerprint
reduced fingerprint
extended valence state fingerprint
extended valence state weighted fingerprint

comparison. As the name implies, the *elements* fingerprint contains only chemical elements. The *reduced* fingerprint comprises extended valence states as described in the companion paper[19] but has a reduced set of ring membership states that is registered for each node. The *extended valence state* fingerprint is the standard version and contains all possible atomtypes that are part of the NAOMI framework. The *extended valence state weighted* fingerprint is the same as the extended valence state fingerprint but undergoes some weighting before the similarity score is calculated.

The weighting is based on valence state statistics extracted from a set of molecules that represents the type of chemical space in which one seeks to compare patterns. We used a set of ~370 million molecules comprising all ZINC15[25] 2D compounds that were available on June 8, 2017. In the following we refer to this set as the *validation data set*. The weighting of each atomtype in the fingerprint is defined as the relative frequency of that atomtype in the set of molecules used to calculate the valence state statistic:

$$S_{\mathrm{WT}}(X, Y) = \frac{\sum_i w(i)(X_i \wedge Y_i)}{\sum_i w(i)(X_i \vee Y_i)} \tag{2}$$

The standard similarity procedure in SMARTScompare is based on the weighted fingerprint Tanimoto similarity (S_{WT} in eq 2) with a Tanimoto normalization by the nodes of the involved patterns (eq 3):

$$S(P_1, P_2, M) = \frac{\sum_{(X,Y)\in M} S_{WT}(X, Y)}{|P_1| + |P_2| - |M|} \tag{3}$$

The weighting factor $w(i)$ represents the frequency of extended atomtypes as found in the validation data set. This procedure for node similarity is depicted in Figure 1.

This similarity measure can be adjusted by every user of SMARTScompare to a specific kind of chemistry. From every set of molecules, the atomtypes can be extracted with a utility tool and valence state statistics can be generated and employed in SMARTScompare.

At present, fingerprints of pattern edges are not explicitly considered during similarity calculations. Since bondtypes are usually reflected in the fingerprints of the incident nodes, edge properties are implicitly incorporated.

Aromaticity Detection. Some implications of the SMARTScompare aromaticity handling are detailed in the companion paper.[19] Here we discuss why we deemed it necessary to include ancillary aromaticity handling. In SMILES, aromatic systems are detected independently of the specific notation of the input string. With SMARTS, however, bonds are always handled explicitly as aromatic single or double bonds. Aromaticity usually cannot be deduced from SMARTS patterns. Although confusing from the application point of view, there are good reasons for this behavior. While SMILES strings always describe full molecules, SMARTS strings represent substructures, which might not be sufficient for aromaticity detection (also see section 4.7 in the Daylight Theory Manual.[4]). For the SMARTScompare approach, we implemented two *optional* features supporting the handling of aromaticity within SMARTS patterns: *SMILES-like aromaticity detection* and *Detect aromatic bonds*. In SMARTS, C1=CC=CC=C1 is not equivalent to c1ccccc1 or c1:c:c:c:c:c1. However, when parsed as molecules, these three representations are identical. Chemistry toolkits like NAOMI[26] parse a molecule and then determine aromatic ring systems on the basis of the assigned valence states and the Hückel rule. In order to support the chemical intuition of aromaticity (e.g., the π system in a benzene is always aromatic no matter the notation), the SMARTS preprocessing in SMARTScompare includes an aromaticity detection step. This behavior can be enabled with the *SMILES-like aromaticity detection* option. Whenever a ring system is encountered during parsing of a SMARTS pattern, all Kekulé localizations of a SMILES string that are possible with the given ring size and elements are determined. Information on neighboring nodes of the ring system is also included. If any localization is classified as aromatic, the fingerprints of the SMARTS graph are updated to incorporate aromaticity. To all edges a disjunct aromatic bondtype is added, and for all nodes all of the aliphatic node queries are replaced with generic element specifications (e.g., the bond "=" becomes "=,:" and "[C]" becomes "[#6]", whereas "c" is not modified). Additionally, in the case of small rings (eight-membered or smaller) in which each node is exclusively aromatic, the edge fingerprints are updated by resetting all nonaromatic bits to 0. This last step ensures that all aromatic systems are recognized as identical, independent of whether edges are written in implicit or aromatic form. The *Detect aromatic bonds* option allows SMARTScompare to

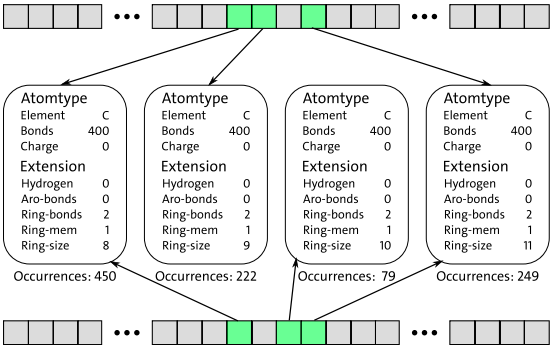


Figure 1. Fingerprints of two nodes. Green squares represent extended atomtypes that exist in the node the fingerprint describes. Beneath each atomtype box, the frequency of the particular atomtype is displayed. It is derived from the validation data set. This value is used as the weight in the calculation of node similarity in case the Tanimoto similarity is selected.

DOI: 10.1021/acs.jcim.9b00249
J. Chem. Inf. Model. 2019, 59, 2572−2586

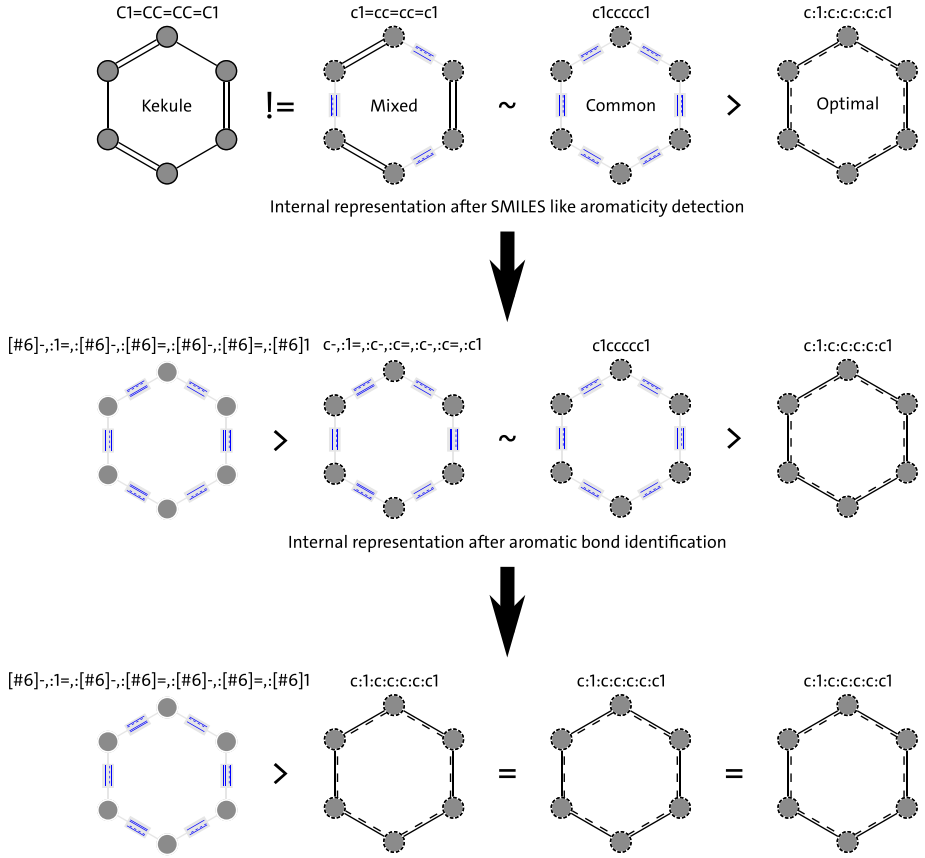


Figure 2. The first row shows possible variations of SMARTS patterns that were designed to match benzene. The second row shows the internal representation of the initial SMARTS pattern in the case where *SMILES-like aromaticity detection* is active. Patterns in the third row represent the internal form when *Detect aromatic bonds* identification is active as well. The logical operators between pattern depictions refer to the compatibility of node and bond fingerprints of the respective pattern pairs: !=, no relation whatsoever; ~, some relation in similarity mode; >, subset relation; =, identity.

transform bond descriptions into one uniform aromatic description. This is achieved by removing all bond information but the aromatic one. Figure 2 showcases this behavior on four notations for matching benzene that are all valid SMARTS strings.

It is important to keep in mind that when this feature of aromaticity handling is used, the underlying model of SMARTS is no longer consistent with the Daylight model.[4] Some internal representations that are needed to detect aromatic systems correctly can no longer be used to match molecules. Additionally, with *SMILES-like aromaticity detection* enabled, the Kekulé form is classified as more generic than its aromatic counterpart. When the *Detect aromatic bonds* option is enabled, nonintuitive behavior is possible in edge cases. An example, that is also described in the companion paper[19] is the handling of bonds that connect two aromatic systems of condensed rings. During pruning of the bond fingerprints they lose the information that they can be single bonds. Only the aromatic state remains, which would lead to wrong subset matching and similarity assessments.

METHODS

To validate the similarity concept, we first tested the coverage of enumerated extended atomtypes using the validation data set by assigning extended atomtypes to all atoms. Second, we tested whether we could model pattern similarity in a chemically intuitive manner.

Validation of Atomtypes. It is important to keep in mind that the NAOMI ChemBio library was designed to handle chemistry that is common in a medicinal chemistry setting. Any atomtypes that are currently not covered by our internal chemical model impair our ability to calculate the similarity and set relations of patterns. The validation data set has sufficient size and breadth to reliably test all necessary atom states of our chemistry model.

There are two fingerprint-related limitations that we deliberately tolerated. The set of extended atomtypes has limitations related to the maximal ring size and the number of URFs[23,24] to which an atom belongs. In an analysis of the ZINC molecules, we found that limiting URF membership and ring size is a good way to balance the coverage of necessary atomtypes and fingerprint length. Overall, we found 506

DOI: 10.1021/acs.jcim.9b00249
J. Chem. Inf. Model. 2019, 59, 2572−2586

molecules containing atoms with more than four URFs (18 atoms) or in rings larger than 24 atoms (12 896 atoms).

Furthermore, 374 atoms in 353 molecules had more attached hydrogen atoms than expected by the NAOMI chemistry model.[27] Atoms for which no extended atomtype is included in NAOMI were aggregated with the atomtype that has the maximal valid count of hydrogen annotated. These aggregations occurred mainly for sulfur and phosphorus atoms.

Lists of all molecules that gave an error with a short description of the type of error (unmodeled hydrogen counts, exceeding ring size enumeration, exceeding URF count) can be found in the Supporting Information.

Validation of the Pattern Similarity Concept. Most applicants of SMARTS consider patterns to be similar if the substructures matched by them are similar. To evaluate our similarity measure, we used the following experimental setup. For a given pattern P_1, we analyzed the fragments matched by P_1 in all molecules $CS(P_1) \subset CS$. Two patterns P_1 and P_2 are considered similar if all of the matched fragments have corresponding similar fragments in the other set. As a similarity measure of fragments, the diameter 6 extended-connectivity fingerprint (ECFP_6)[28] was applied. We employed the following measurement, the substructure similarity, as the reference similarity for SMARTScompare:

$$\mathrm{FL}(P) = \langle f \subseteq m \mid P \text{ matches } f,\ m \in CS(P) \rangle \tag{4}$$

$$\mathrm{sim}(f_m, f_n) = \frac{|\mathrm{ECFP\_6}(f_m) \cap \mathrm{ECFP\_6}(f_n)|}{|\mathrm{ECFP\_6}(f_m) \cup \mathrm{ECFP\_6}(f_n)|} \tag{5}$$

$$S_{\mathrm{ref}} = \frac{1}{|\mathrm{FL}(P_1)| + |\mathrm{FL}(P_2)|}\left[\sum_{f_m \in \mathrm{FL}(P_1)} \max_{f_n \in \mathrm{FL}(P_2)} \mathrm{sim}(f_m, f_n) + \sum_{f_n \in \mathrm{FL}(P_2)} \max_{f_m \in \mathrm{FL}(P_1)} \mathrm{sim}(f_m, f_n)\right] \tag{6}$$

To validate our similarity approach, we performed two experiments in which we focused on two types of patterns. The first experiment was based on substructure-like patterns with almost no SMARTS-specific features. Patterns that matched at least 10 000 molecules in the validation data set were selected from a set of SMARTS patterns published by Ehrlich and Rarey.[29] Similarity of molecules was calculated on the basis of the Tanimoto similarity of the substructures matched by the SMARTS pattern. The second experiment was based on structural filters. Here we used the SMARTS filters taken from ChEMBL (see Table 2). As set of molecules for this experiment, we took the set of unique molecules from ChEMBL22.[30] Those patterns are usually smaller and use more SMARTS features than those in the first experiment.

Figure 3 shows the correlation plots for the two experiments. The experiments are described and analyzed in more detail in the Supporting Information (see SI1). The plots show that substructure-like patterns (Figure 3a) exhibit a good distribution as well as a good correlation, whereas the similarity for the structural filters (Figure 3b) is dominated by similarity scores below 0.2.

Overall our experiments show that pattern similarities calculated by SMARTScompare correlate with molecule similarities derived from matching of fragments to a large reference data set.

Table 2. Publicly Available Filter Sets As Published in ChEMBL[2] (Excluding SMARTCyp)

name	no. of patterns	publication
Bristol-Myers Squibb	180	Pearce et al.[10]
SMARTCyp	42	Rydberg et al.[11]
Dundee	105	Brenk et al.[13]
Glaxo Wellcome	55	Hann et al.[6]
Inpharmatica	91	no publication available
MLSMR	116	website offline/project discontinued
LINT	57	Blake[12]
PAINS	481	Baell et al.[8]
SureChEMBL OCHEM/ ToxAlerts	166	Sushko et al.[7,39]

SMARTScompareViewer. Even for an expert user, SMARTS subset relations determined by the SMARTScompare algorithm are sometimes hard to comprehend. To visually support the interpretation, we have developed the SMARTScompareViewer, an intuitive tool that visualizes SMARTS patterns and shows the calculated node mappings that result from the SMARTScompare algorithm. The SMARTScompareViewer is based on the SMARTSView concept and the SMARTSViewer.[14,15] Besides showing the actual node mapping, the SMARTScompareViewer can emphasize the difference between SMARTS nodes. This helps the user to understand unexpected results, meaning that there are more or fewer nodes mapped than expected. SMARTScompareViewer is also available on the web as part of the SMARTSviewServer (https://smarts.plus).

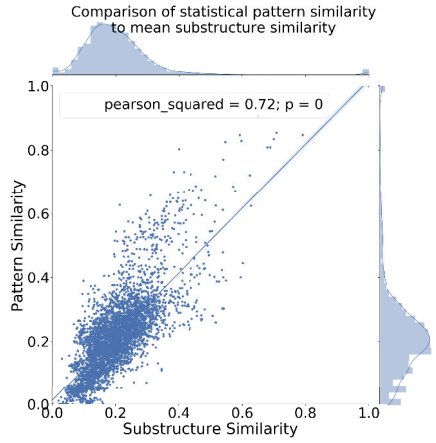
The SMARTS Filter Sets. With an implementation of SMARTScompare, we are able to perform a comprehensive comparison of publicly available filter sets for the first time. Filter sets are used by many pharmaceutical companies and research groups in academia to remove molecules with unwanted properties such as reactivity, toxicity, or assay interference. Since these filter sets are mostly handcrafted, it is interesting to compare different flavors of patterns aimed at the same property. Usually, specific substructural features are responsible for unwanted behavior of compounds. Substructures can be described via SMARTS patterns, allowing computationally easy selection of molecules having such chemical properties.

One of the first pharmaceutical companies to publicly release such a filter set was GlaxoSmithKline (then Glaxo Wellcome Research).[6] The SMARTS patterns published are structured into four groups: reactive functional groups, unsuitable leads, unsuitable natural products, and another group summarizing all SMARTS filters for acids, bases, electrophiles, and nucleophiles. The main purpose of the published filter sets is to avoid pooling of compounds that react with each other within one batch during HTS.

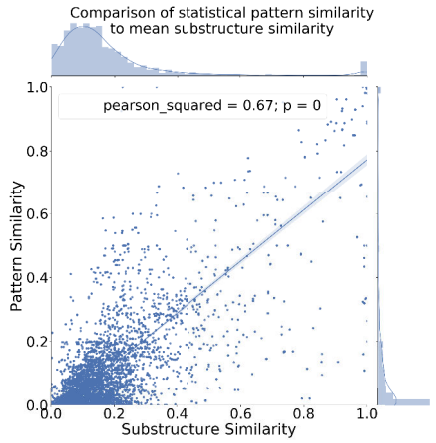
A second set of SMARTS patterns with a similar focus was published by Bristol-Myers Squibb.[10] The authors were looking for a consistent way to remove compounds from HTS screening decks and to flag compounds that do not disturb screenings but are helpful in assay evaluation when flagged as potentially disruptive. Other filter sets were designed with a broader focus in mind.

James Blake of Array BioPharma Inc. investigated how to reduce general attrition during drug development.[12] Apart from the usual suspects drug-likeness, lead-likeness, and

DOI: 10.1021/acs.jcim.9b00249
J. Chem. Inf. Model. 2019, 59, 2572−2586



(a) Similarity correlation for substructure like patterns



(b) Similarity correlation for structural filters

Figure 3. Correlation plots for pattern and substructure similarity. Pattern similarity refers to the calculated value of SMARTScompare for two patterns. The substructure similarity is the Tanimoto similarity of the substructures that the nodes of the patterns match. Plot (a) corresponds to the first experiment with substructure-like patterns, and plot (b) corresponds to the second experiment with filters that include more SMARTS features.

implications of computed properties, the author addressed functional groups linked to problematic performance during the drug development process. Blake tied a certain percentage of the overall attrition to "reactive groups and compounds or functionalities that have been shown to be mutagenic or carcinogenic". Those groups also "tend to give false positives in high-throughput screens". Researchers from the University of Dundee in Scotland published a filter set with a similar focus but specializing in neglected diseases.[13]

Baell and Holloway focused solely on pan-assay interference compounds (PAINS).[8] As a result of the increased interest in HTS at the time of their writing, they wanted to develop substructure filters that efficiently encode structural information on problematic compounds. With such a description at hand many interfering compounds and their analogues can be excluded from pending bulk purchases.[8] The substructures were originally published in SLN[5] and later translated into SMARTS patterns.[31] The OCHEM project also translated the SLN notation of the PAINS patterns.[7,32] In this case they were translated by hand. In the Results we discuss the difference in the two translations.

There are two other projects in the context of HTS screening, but no official information regarding their purpose could be found as of the release date of this work. The now-discontinued National Institutes of Health Molecular Libraries Small Molecule Repository (NIH-MLSMR), which was a part of the now-retired Molecular Libraries Probe Production Centers Network (MLPCN), assembled a set of SMARTS filters that is now managed by Evotec.[2,33] The purpose of the assembled filter sets was to remove "chemically reactive functional groups that would interfere with HTS, as well as compounds likely to be promiscuous aggregators"[34,35] with a reference to McGovern et al.[36] ChEMBL has another set of SMARTS filters that were derived by Inpharmatica Ltd.,[37] for which also no official publication exists. The filter set was initially intended to be used on ChEMBL data (when they were still a commerical product of Inpharmatica Ltd.). The purpose was to filter groups that are generally undesirable during drug design.[34]

The following two filter sets have a focus that diverges from those already presented. Sushko et al.[7,32] assembled a web server[38] for structural alerts for toxic chemicals and compounds with potential adverse reactions. The OCHEM/ToxAlerts[7] initiative is an ongoing project, and every user is encouraged to submit new structural alerts. SureChEMBL uses SMARTS patterns for structural highlights on their website that are extracted from the OCHEM/ToxAlerts SMARTS set. These patterns are also part of the ChEMBL releases.[2]

SMARTCyp by Rydberg et al.[11] has a completely different focus from all other SMARTS sets. SMARTCyp is an in silico method that predicts reaction sites of drug-like molecules in cytochrome P450. The idea is to precalculate the reactivity for certain atoms that are part of a chemical group as the activation energy for oxidation reactions. For each atom of the chemical group, a SMARTS pattern is generated and, together with the corresponding energy, stored in a database. When a query molecule is given, a score for each atom is extracted from the precalculated energy values using the fitting SMARTS pattern.

ChEMBL published SMARTS versions of the previously listed structural filters with their 2017 release.[2] A summary of the SMARTS filter sets can be found in Table 2. All of the SMARTS patterns used in this work were taken from one of the SMARTS sets listed in Table 2. Except for SMARTCyp, which was extracted from the publication, all of the filter sets were taken from the MySQL ChEMBL release dump.[2] All of the patterns were used unchanged except for seven patterns from the PAINS filters. These patterns were modified to reduce the number of explicit hydrogens. The replacement of explicit hydrogens by implicit ones substantially reduces the run time of our MCS calculation, which maps all explicit atoms exhaustively. These modified PAINS patterns are provided in the Supporting Information.

In the case of PAINS filters, we had the opportunity to compare two sets of patterns that were both translated from SLN into SMARTS. One set of PAINS SMARTS patterns were taken from the ChEMBL release as stated above. The

DOI: 10.1021/acs.jcim.9b00249
J. Chem. Inf. Model. 2019, 59, 2572−2586

other SMARTS set, used in the publication of Schorpp et al.,[40] was extracted from the OCHEM website.[38]

5. RESULTS AND DISCUSSION

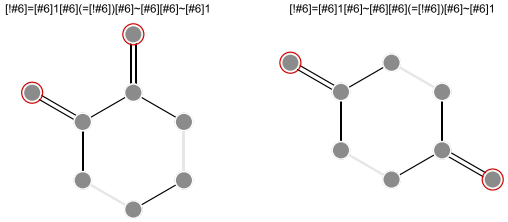
In the following paragraphs we will showcase the capabilities of the algorithm detailed in the companion paper[19] and summarized at the beginning of Theory. The focus of our analysis lies with the set relations of patterns and their similarity. One of our central assumptions is that patterns in filter sets were designed with a specific purpose in mind. We are estimating the intentions of the authors on the basis of the labels that patterns were given and the publications with which they were released, if they exist.

Set Relations and Pattern Redundancy in Filter Sets. One of the simplest application cases of our algorithm is the analysis of existing pattern sets with regard to their content. Often we want to know whether a certain type of chemistry is already contained in the filter set. For this work, four questions are of importance in particular:

- Does our algorithm find all of the related patterns?
- Since SMARTS patterns of the respective filter sets were labeled with the systematic or trivial name, how large is the unwanted overlap between patterns for different purposes?
- Do the patterns actually have the desired form (as specified in the corresponding label)? Sometimes a more precise or general form of a pattern might meet the desired criteria better than the presented one.
- Can we reduce the number of patterns by identifying and removing redundant ones?

To analyze the filter sets regarding their overlap and coverage of chemical functionality, we employed subpattern calculations and string search on the pattern labels.

Consistency of Label and Pattern. Our first experiment focused on patterns that are designed to filter out reactive compounds containing quinone-like substructures. On the one hand, we designed two generic patterns (Figure 4) to find as



(a) The ortho variant of our quinone query patterns (1,2-benzoquinone)

(b) The para variant of our quinone query patterns (1,4-benzoquinone)

Figure 4. Two generic SMARTS patterns to find as many quinone SMARTS strings as possible: (a) ortho version; (b) para version. The focus here lies on finding all patterns that are labeled as quinones and then analyzing the bycatch.

many quinone patterns as possible. In total, we found 14 patterns, of which 11 match the para variant and three the ortho variant. Eleven of them have annotations referring to quinones. The three patterns without a quinone label are annotated as *keto_keto_gamma(5)* (PAINS), *Dye 1 (1)* (MLSMR), and *Dye 4* (MLSMR), as can be seen in Figure 5. For comparison, we conducted a simple string search on the labels of patterns. Any SMARTS pattern whose label contained

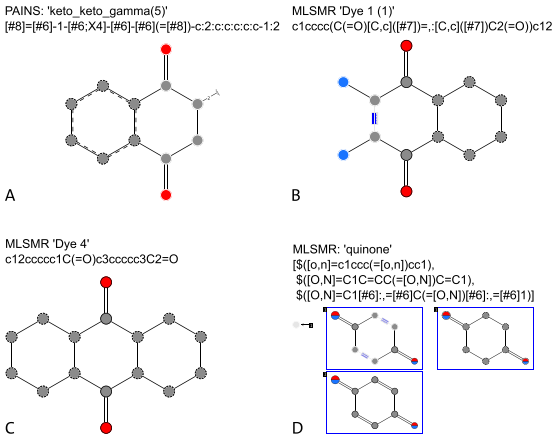
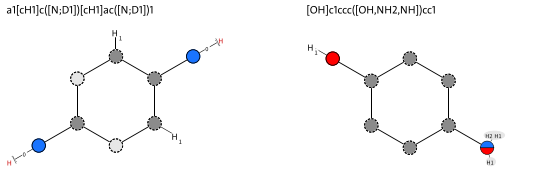


Figure 5. Four patterns that are matched by the quinone query patterns shown in Figure 4 without having a quinone label. Pattern A is part of the PAINS filter set. Patterns B and C are part of the MLSMR filter set. Pattern D is a more generic SMARTS pattern that is part of the MLSMR filter set and renders patterns B and C obsolete when used together.

the string "quinone" was selected as a result. Seventeen patterns were found. Patterns designed to match derivatives of quinones may still contain the "quinone" substring and are in the solution set of our query. This naming scheme results from the systematic nomenclature that is employed in organic chemistry. One difference in numbers of patterns is attributable to the class of hydroquinones (Figure 6).



(a) *para_hydroiminiquinone* as given in the BMS filter set.

(b) *hydroquinone* as given in the Dundee filter set.

Figure 6. Two quinone derivatives that are found in the pattern sets of BMS and Dundee.

Quinones and their derivatives, when hydrogenated at the heteroatom, are transformed into their quinol form. These aromatic forms are not matched by our query patterns as we have designed them. Hence, they are not in the list of patterns we find with either of our two query patterns. They could, however, be easily incorporated. Another difference between the string search and the pattern query are the two patterns displayed in Figure 7. The description of the ring and bonds to the heteroatoms diverge drastically from our notation for quinones. It is in general debatable whether the two patterns can still be classified as quinones. Last but not least, simple notation differences have an impact on substring search too. "Quinone" can also be written as "chinone", which in the case of string matching means we would miss two patterns (77 and 78) from the Dundee filter set.

In general, the pattern labels describe molecules or set of molecules from the class of quinones well. Aside from the two examples shown in Figure 7, the labels are very accurate and descriptive. Depending on the design of the query pattern, all

DOI: 10.1021/acs.jcim.9b00249
J. Chem. Inf. Model. 2019, 59, 2572−2586

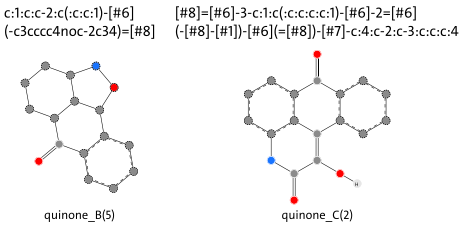


Figure 7. Two patterns from the PAINS filter set that we did not find with the generic quinone patterns (Figure 4). Both are labeled as quinone patterns but also include heteroatoms, rendering them unmatchable for our generic query.

of the patterns labeled as quinones or their derivatives can be found. Nonetheless, everything depends on the formulation of the query pattern. However, we did find patterns containing a quinone-like subpattern that we would have been unable to identify without our algorithm. It is important to remember that a subset relation of patterns is manifested as overlapping sets of molecules that are matched by each of the patterns. In our case this would have an impact on the necessity of patterns. In the case where we would include our query patterns into the filter set that also contains the dye patterns (Figure 5), we could remove those and still filter out all molecules that would have been matched by the dye patterns. Furthermore, the MLSMR filter set already contains a very general quinone pattern (Figure 5D). It matches all molecules that are matched by the dye patterns shown in Figure 5B,C. If all three patterns are used together, which they usually are in a filter set, the dye patterns are redundant and may be removed from the filter set without loss.

All of the quinone patterns, including our query patterns, are visualized in the Supporting Information.

Pattern Hierarchies and Redundancy. Our second experiment revolved around the question of which more specific patterns are rendered redundant when we start with a very generic pattern. For this experiment, we selected allenes as example chemical group.

For our search we used the pattern *=C=*, as it is part of the MLSMR and SureChEMBL OCHEM/ToxAlerts filter set. We searched for more specific patterns than the generic allene in all of the filter sets and found 12 matches and eight different patterns in total. Those patterns that occurred in more than one filter set were identical, including the annotation.

When visualizing the hierarchy of the resulting patterns, a specificity tree as shown in Figure 8 can be constructed that shows the most generic pattern at the top and more specific patterns on the subsequent levels below. Each pattern below the top pattern is redundant in the sense that each molecule that would be matched by one of the lower patterns will in every case also be matched by the top pattern. This is especially interesting when we turn our attention toward the origin of patterns that are part of the hierarchy. Five patterns are from the same SureChEMBL OCHEM/ToxAlerts toxicity set, of which four are obsolete and can be removed without compromising the integrity of toxicity filtering mechanism.

In our third experiment, we focused on a bottom-up approach by selecting a relatively specific sulfonyl halide pattern with an anchor wildcard node. Here we were curious whether there were any more generic patterns that might be of interest to our fictive task. The search for more generic patterns found 13 patterns including the query from the

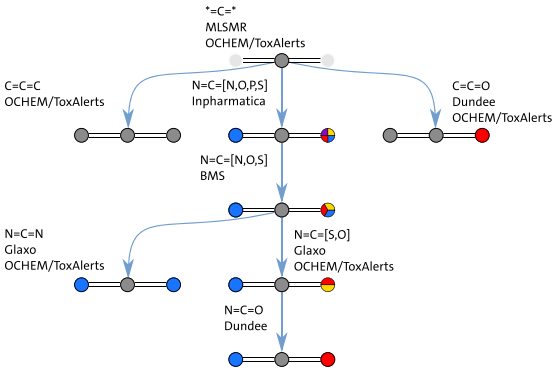


Figure 8. Hierarchical scheme of patterns specifying allene-like chemical groups. The hierarchy starts with the most generic allene pattern at the top and more specific patterns toward the bottom. Equal level does not imply equal specificity, but each pattern on a lower level is more specific than a pattern on the level above.

Inpharmatica filter set. The 13 matches can be grouped into seven partitions of equal patterns. Those equal patterns differed in the order of the SMARTS nodes. The visualization of the set relation we found is shown in Figure 9.

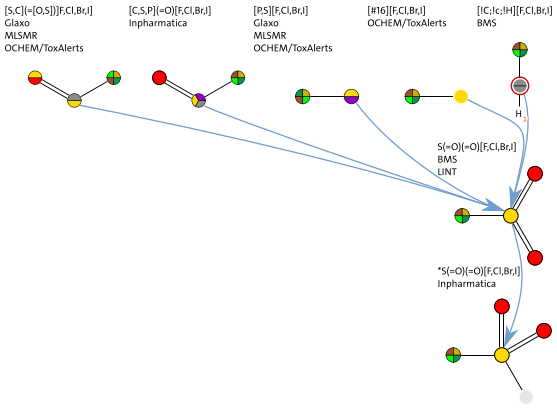


Figure 9. Hierarchical scheme of more generic patterns than the sulfonyl halide pattern. The hierarchy starts again with more generic patterns at the top and gets more specific toward the bottom.

Using SMARTScompare, we are able to analyze the degree to which the intention, derived from the textual label that comes with the pattern, is translated into the final SMARTS pattern. Because of the complexity of the SMARTS language, unforeseen molecules are matched, especially when very generalized patterns are used. The second interesting point is the redundancy of patterns in filter sets. Especially when expressions get more complicated, keeping track of each function that has been cast into a pattern gets increasingly more difficult. By analyzing set relations, we are now able to produce hierarchical schemes that give an easy-to-perceive overview of the subset relations of patterns. This allows us to eliminate redundant patterns and sharpen the focus of the filter set.

Filter Set Similarity Analysis. For generic pattern set comparison, the explicit search for more specific or generic

DOI: 10.1021/acs.jcim.9b00249
J. Chem. Inf. Model. 2019, 59, 2572−2586

patterns is not well-suited. The binary form of the result is too coarse-grained and forbids a subtle analysis of small differences between patterns. This problem is better addressed with a similarity-based approach.

In order to calculate the similarity between two filter sets, we apply the following procedure:

- For each pattern of the query filter set, we compute the similarity to each pattern in the second filter set.
- For each query pattern, we select the most similar pattern from the second set.
- The average of the similarity values of the most similar pattern pairs is calculated between the two sets as an asymmetric similarity measure of those two sets.

The self-similarity of a SMARTS pattern set is calculated as described above, with the one difference that the query pattern was excluded. In this way, we get an idea of the most similar pairs of patterns in a filter set.

The similarity value of two different sets can be considered as a degree of coverage of one filter set by the other. Our analysis of pattern set self-similarity, on the other hand, was conducted with the assumption that higher dissimilarity covers a greater variety of chemical functions and the set contains fewer redundant patterns. Similarity between node fingerprints of a pattern pair was calculated with the weighted Tanimoto coefficient shown in eq 2. The weights were derived from the validation data set as described in Similarity Calculation. In the following paragraphs we discuss the similarity values between filter sets and, if the given data allow for it, tentatively analyze possible reasons.

The Glaxo set constitutes an excellent beginning since it was the first filter set published and also was designed with a similar goal as the majority of the other sets. Figure 10 displays the accumulated values for the Glaxo set's most similar pairs. There is one major peak in the histogram with its maximum at ∼0.3 and a local maximum at ∼0.5. Since there are no filter pairs with similarity 1.0, the filter set does not contain any duplicates. With a self-similarity mean of 0.34, the Glaxo filter set lies at the lower end of self-similarity values (see Figure 13 for more values).

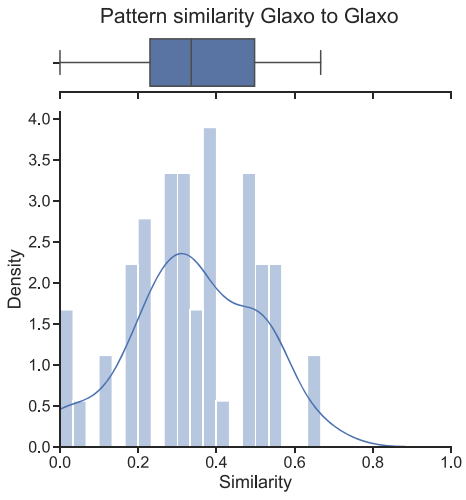


Figure 10. Counts of the self-similarity values for the Glaxo filter set. The mean similarity lies at 0.34. The set has no identical patterns. The density plot has a maximum at ∼0.3 and another local maximum at ∼0.5.

When we compare the Glaxo set with all of the other filter sets, we get the similarity values displayed in Figure 11. The

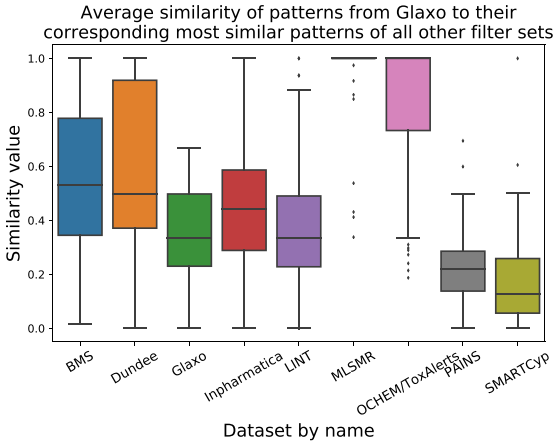


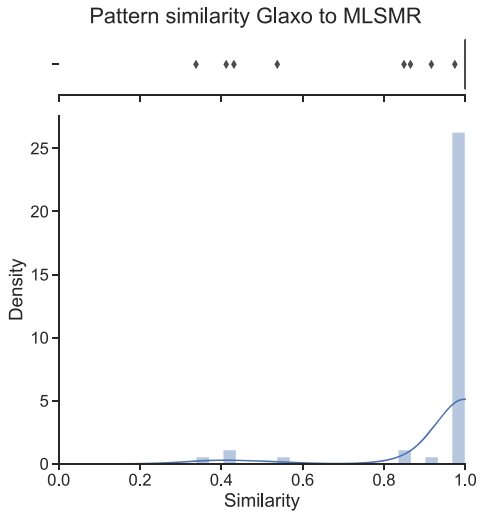
Figure 11. Box plots of the similarity histograms for pattern similarity for all patterns from the Glaxo Wellcome filter set to their corresponding most similar patterns in all of the other filter sets. The Glaxo box plot represents the self-similarity of the Glaxo filter set.

filter set similarities that draw our attention are the ones with extreme similarity or dissimilarity values. MLSMR and SureChEMBL OCHEM/ToxAlerts are the two with the highest similarity values, while SMARTCyp and PAINS are most dissimilar to the patterns of the Glaxo filter set. The similarity value distributions of these four filter sets are displayed in Figure 12. The similarity between the Glaxo and MLSMR sets is surprisingly high. The extent of the similarity indicates that the Glaxo filter set is almost fully contained in the MLSMR set. However, it does not indicate that most of the MLSMR patterns are from the Glaxo set. In Figure 13 we can see that the similarity value from MLSMR to Glaxo is 31% lower. In contrast, the histograms in Figure 12c,d barely contain similarity values above 0.5, showing that most of the Glaxo patterns are not covered by PAINS and SMARTCyp.

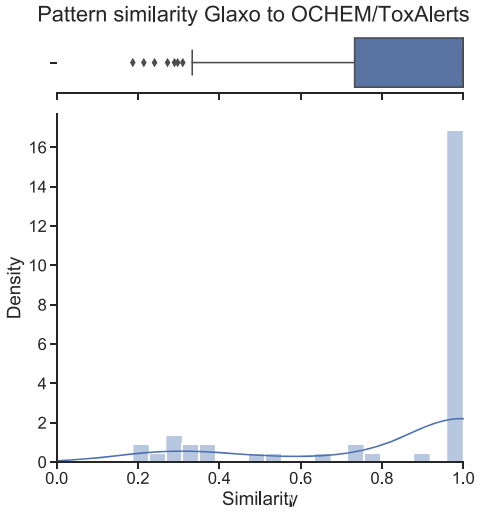
Finally, we extended this experiment to a large-scale all-against-all filter set comparison, as can be seen in Figure 13. Most of the average similarities fall between 0.4 and 0.6. Also, SMARTCyp and PAINS are most dissimilar to all of the other filter sets. The asymmetry of our set similarity measure is clearly visible for the SMARTCyp set. The average similarity for any filter set to SMARTCyp is below 0.25, and in most cases even below 0.2. In the opposite direction, the average similarity from SMARTCyp to any other filer set is above 0.25 except for PAINS (0.16). The other extremes are Glaxo to MLSMR with a similarity of 0.95 and Glaxo to SureChEMBL OCHEM/ToxAlerts with a similarity of 0.83.

There might be many reasons why the Glaxo−MLSMR similarity value is so high. Since the Glaxo set was the first to be published and the MLSMR project was a concerted effort of several institutions only a few years later, it probably was incorporated into the MLSMR filter set almost unchanged. When we were analyzing the two filter sets with the SMARTScompare tool, we found that there are only nine

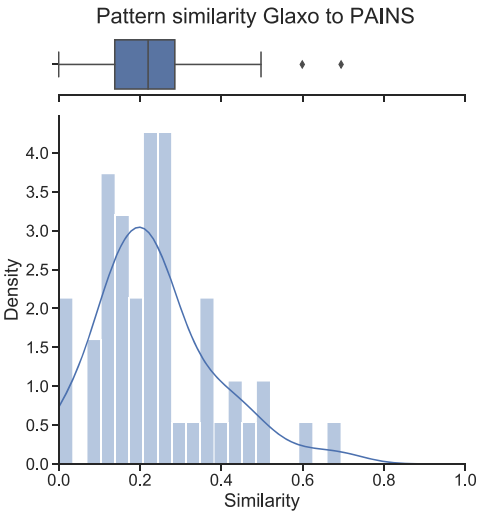
DOI: 10.1021/acs.jcim.9b00249
J. Chem. Inf. Model. 2019, 59, 2572−2586



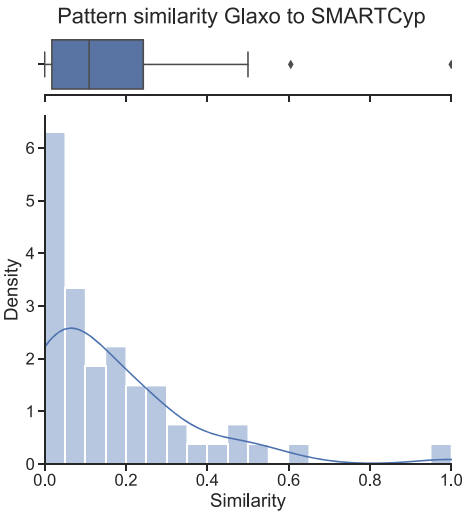
(a) Histogram ofthe most similar values ofthe MLSMR filter set when each pattern ofthe Glaxo filter set was used as query once.



(b) Histogram ofthe most similar values of the SureChEMBL OCHEM/ToxAlerts filter set when each pattern ofthe Glaxo filter set was used as query once.



(c) Histogram ofthe most similar values ofthe PAINS filter set when each pattern ofthe Glaxo filter set was used as query once.



(d) Histogram ofthe most similar values ofthe SMARTCyp filter set when each pattern ofthe Glaxo filter set was used as query once.

Figure 12. Histograms of similarity values of the Glaxo Wellcome patterns from searches for the most similar ones in four other filter sets. The four filter sets were selected on the basis of their high similarity (a, b) or dissimilarity (c, d).

patterns in the Glaxo set that are not identically included in the MLSMR set. They are shown in Table 3.

Eight of the nine patterns can be observed as differing values in the histogram in Figure 12a and the box plot in Figure 11, where they are classified as outliers by the box plot visualization. The ninth pattern (R22) differs from its most similar MLSMR pattern only in the order of the halogens. The reverse similarity lies at 0.64, which also makes sense when one considers the slightly different focus of the MLSMR filter set. They not only tried to avoid pooling of reactive compounds but also were interested in filtering out promiscuous aggregators as defined by McGovern et al.,[36] leading to a drop of similarity of 31%. The high similarity between the Glaxo and SureChEMBL OCHEM/ToxAlerts filter sets makes sense too if we take into account that many toxic compounds exhibit toxicity because of their reactive nature. When we have

DOI: 10.1021/acs.jcim.9b00249
J. Chem. Inf. Model. 2019, 59, 2572−2586

Journal of Chemical Information and Modeling Article

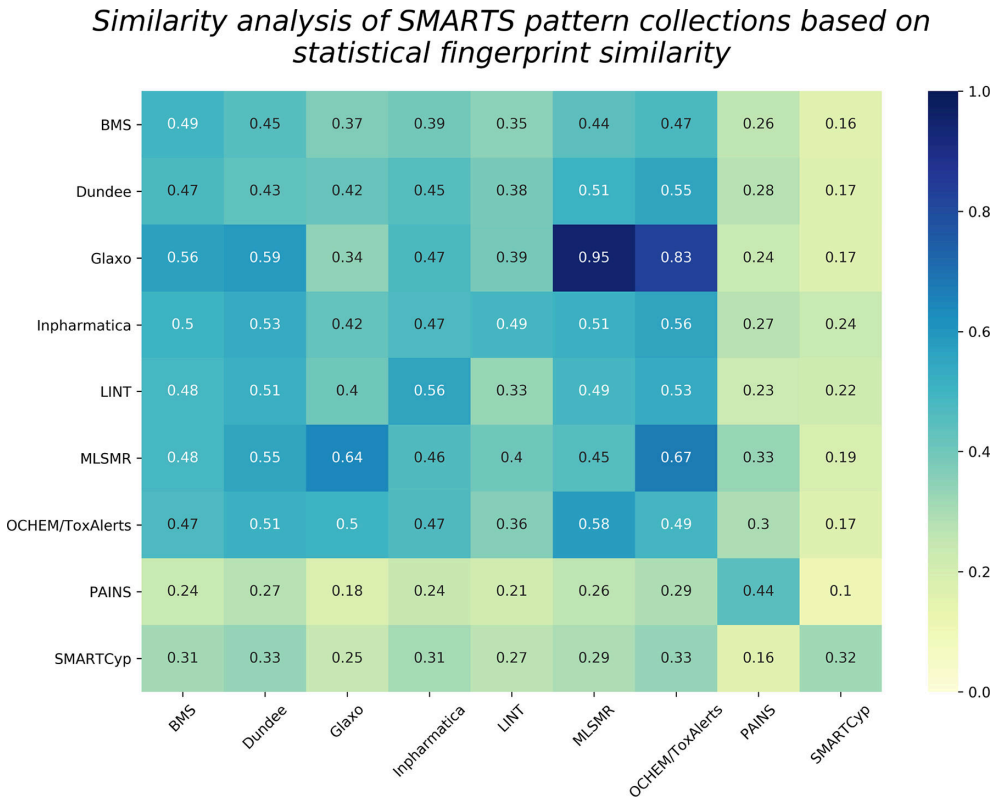


Figure 13. All-against-all filter set similarity matrix plot. Each cell describes the average similarity for patterns from one filter set and their corresponding most similar patterns in the other filter set. On the diagonal, the filter set self-similarity is measured. Here the most similar nonidentical pattern is averaged; diagonal entries of 1 would indicate plenty of pattern duplicates.

Table 3. Patterns from Glaxo That Are Not Identically Part of the MLSMR Filter Set

annotation	pattern
N1 Quinones	O=C1[#6]~[#6]C(=O)[#6]~[#6]1
R1 Reactive alkyl halides	[Br,Cl,I][CX4;CH,CH2]
R3 Carbazides	O=CN=[N+]=[N−]
R23 Carbodiimide	N=C=N
I1 Aliphatic methylene chains 7 or more long	[CD2;R0][CD2;R0][CD2;R0][CD2;R0][CD2;R0][CD2;R0][CD2;R0][CD2;R0]
R11 Isocyanates & Isothiocyanates	N=C=[S,O]
R16 beta carbonyl quaternary Nitrogen	C(=O)C[N+,n+]
R9 Paranitrophenyl esters	C(=O)Oc1ccc(N(=O)~[OX1])cc1
R22 P/S Halides	[P,S][Cl,Br,F,I]

a look at the similarity of SureChEMBL OCHEM/ToxAlerts to Glaxo, on the other hand, we see the difference in reactivity and toxicity ingrained into the similarity value. Toxicity is caused not only by reactive compounds but also compound classes that interfere with signaling processes in the body and compounds that are translated into their toxic form by means of enzymatic activity. This leads to a drop of 33%.

The dissimilarity of the Glaxo and SMARTCyp filter sets (0.17) can be explained by the very different focuses of the filter sets. The Glaxo filter set was specifically designed to exclude highly reactive compounds from screening libraries. SMARTCyp was designed to describe very small, often only one atom in size, chemical groups that act as keys in a dictionary of density functional theory calculations, which is orthogonal to the goal of HTS library design.

The dissimilarity of the PAINS and Glaxo filter sets (0.24), on the other hand, is not explained easily, and formulating a sophisticated hypothesis would require a more thorough analysis. One reason about which we are confident is the substantial difference in pattern size. The PAINS pattern set includes many patterns describing whole molecules, while the Glaxo filter set focuses on reactive chemical groups. Figure 14 shows an example of a pattern from the PAINS filter set. The similarity of PAINS to SMARTSCyp (0.1) constitutes the global minimum of our similarity analysis. This supports our hypothesis that large parts of the dissimilarity can be attributed to size difference: large, very specific SMARTS expressions in the case of PAINS filters and one-atom expressions in the case of SMARTCyp.

To deepen our understanding of the filter sets and SMARTS−SMARTS similarity we analyzed highly similar pattern pairs. We searched for the most similar pattern for each of the nine patterns from the Glaxo−MLSMR example that were not found in both sets (Table 3).

We will discuss the patterns *R23 Carbodiimide* and *R11 Isocyanates & Isothiocyanates* exemplarily. Both have the same most similar pattern within the MLSMR filter set, namely, *=C=*, annotated as *allene* (see Figure 15). First, we notice

DOI: 10.1021/acs.jcim.9b00249
J. Chem. Inf. Model. 2019, 59, 2572−2586

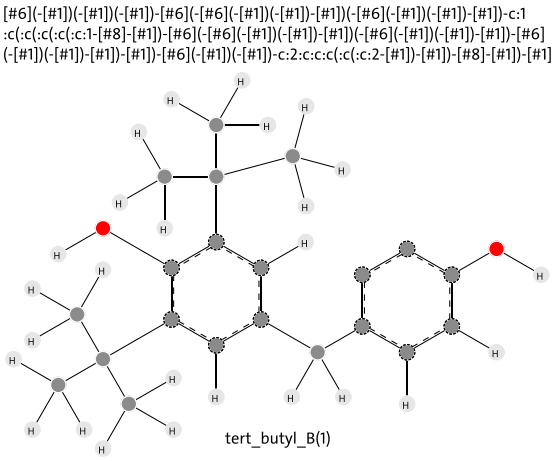


Figure 14. SMARTS pattern taken from the PAINS filter set (*tert_butyl_B(1)*) that looks like a molecule was translated into a SMARTS pattern.

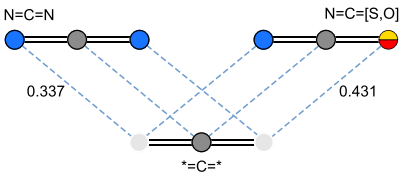


Figure 15. Two patterns from Glaxo with a common most similar pattern from MLSMR. In the upper part of the picture, the two Glaxo patterns are shown with their similarity values to the MLSMR pattern, which is displayed in the lower part of the figure. Although all of the nodes participate in the mapping, both patterns have similarities smaller than 0.5. The wildcards in the MLSMR pattern are the reason for those low similarity scores.

that the MLSMR pattern is more generic than both of the Glaxo patterns. Therefore, it will match all molecules matched by the two Glaxo patterns. When interpreting the SMARTS patterns, we could say that all cumulated double bonds around carbon atoms are considered as reactive by the authors of the MLSMR filter set. The Glaxo patterns, on the other hand, require heteroatoms to be included in order for cumulated double bonds to be deemed too reactive. Moreover, visualization of the Glaxo SMARTS patterns makes it very clear that both patterns could be replaced by a single pattern of N=C=[S,O,N] without losing any filtering power, i.e., we would still match the same molecules.

Another example, this time with high similarity (0.64), was taken from the Inpharmatica to SureChEMBL OCHEM/ToxAlerts filter set comparison (Figure 16). Upon examination of the two patterns displayed, the high similarity value is easily explained chemically. Both patterns describe a carbonyl-like group between two carbons, one of which is connected to one of the most abundant atoms in medicinal chemistry, a halogen (F, Cl, Br, I). Differences are a possible nitrogen at the position α to the carbonyl carbon and a higher specificity for the bond to the atom at the α position in the other direction. Because of the different pattern sizes (one pattern with four nodes and the other with five nodes), the similarity is capped at 0.8. Therefore, the similarity value of 0.64 is relatively high

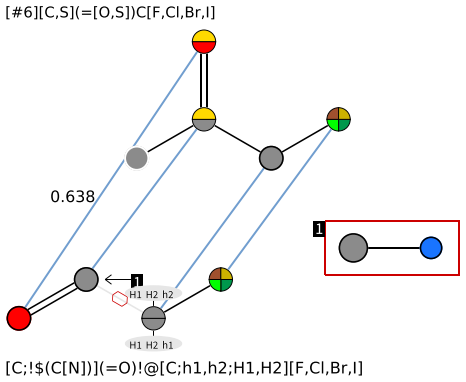


Figure 16. Two patterns from the Inpharmatica and SureChEMBL OCHEM/ToxAlerts filter sets. They are annotated as *Filter64_-halo_ketone_sulfone* and *Alpha_Halo_Carbonyl*. In contrast to the previous example (Figure 15), there is no subset relation between these patterns. Parts of them are vice versa more specific.

considering the pattern differences. Because of the statistical weighting of individual bits, atomtypes with low occurrence like halogens and sulfur substantially increase the similarity. When the reduced fingerprints without weighting are applied, the similarity drops to 0.29.

In conclusion, pattern similarity is influenced heavily by the difference in pattern size (the number of nodes) and the weighting of bits of node fingerprints. This behavior is intended and reflects, to the best of our knowledge, the intuitive understanding of pattern similarity. The pure number of common active bits used for molecular similarity is mostly irrelevant for chemical pattern similarity since the number of atomtypes per element varies substantially. The weighting scheme compensates for this effect. We have observed that patterns with high similarity mostly differ in one or a few nodes only. Moreover, most of the time the difference can be attributed to one node allowing exactly one element while the other allows a second or third element alternatively. Reasons for dissimilarity, on the other hand, are so diverse that we do not feel confident in making a generalizing assertion.

Aromaticity Handling. When handling SMARTS comparison strictly as described in theory,[4] patterns that would be perceived as similar or even identical are not classified as such. This is due to different notations for aromatic systems and a lack of techniques to transform them. Earlier in this work we discussed two modes that we included in the implementation of the comparison algorithm. Those two modes introduce aromaticity detection in which patterns with aromaticity are handled more intuitively during comparison.

To demonstrate this effect, we compare two translations of PAINS patterns into SMARTS that were originally published in SLN.[8] The one we took from the MySQL ChEMBL dump was automatically translated with the tool CACTVS.[31] The other set was translated by hand as part of the OCHEM project.[7] Figure 17 shows an example of two translated SMARTS that stem from the same pattern written in SLN. The two patterns on the left show the compatibility of the pattern nodes in similarity mode with both modes for aromaticity correction switched off. Because of the different notations for the imidazo[1,2-*a*]pyridine matching part of the pattern, most of the pattern nodes are classified as dissimilar. Since mismatches of nodes have an influence on the compatibility

DOI: 10.1021/acs.jcim.9b00249
J. Chem. Inf. Model. 2019, 59, 2572−2586

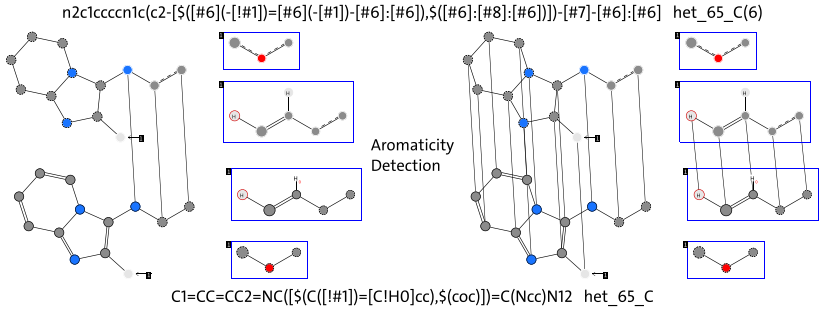


Figure 17. Comparison of two SMARTS patterns that stem from the same pattern written in SLN and were translated by two different sources.[7,31] The pattern was originally published as SLN N[1]C(=C(NC:C)N[6]C=CC=CC=@1@6)Any[IS=C(Hev)=CHC:C,C:O:C].[8] The two patterns on the left show the SMARTS comparison with both modes for aromaticity correction switched off. The comparison on the right, on the other hand, was conducted with both modes switched on. The patterns at the top of the graphic were taken from the set that was automatically translated by CACTVS[31] and is also stored in ChEMBL. The patterns at the bottom of the figure are the ones taken from OCHEM.[7]

of neighboring nodes, the similarity of other nodes in the patterns decreases as well. When making use of the aromaticity correction modes, this incompatibility is overcome by amending node and bond fingerprints internally.

The main two differences we found during analysis of the two sets of SMARTS patterns were different notations for aromaticity and the use of SMARTS language features. While the ChEMBL patterns describe the aromaticity more explicitly with aromatic nodes and edges, the OCHEM patterns use the implicit form of aromaticity. Nodes are written in their aromatic form, and bonds are given implicitly. In terms of use of the features the SMARTS language provides, the OCHEM patterns utilize them way more. Specifications of hydrogens are given as explicit hydrogens in the ChEMBL patterns, while the OCHEM patterns make use of hydrogen counts and declaration of explicit bond counts. A third major difference we found is the use of generic element notation, #<*n*>. CACTVS makes heavy use of this notation, while the hand-crafted OCHEM SMARTS contain this kind of description only when no aliphatic/aromatic classification is possible. Other minor differences exist but add no information necessary for understanding the conceptual differences.

Aromaticity handling is a complex problem with many pitfalls. On the side of SMARTS matching, many different implementations of the original Daylight theory exist. Each implementation has it is own aromaticity handling that depends on the chemical model used. To be independent of the notation that is used to describe potentially aromatic systems in SMARTS, the two described methods were introduced. The presented modes cover the theoretically correct way of handling aromaticity as well as the chemically intuitive one. Nonetheless, we recommend being very careful when using these modes, as there might be unexpected side effects.

CONCLUSION

We have presented the application of SMARTScompare, which allows analysis of chemical patterns in the form of SMARTS expressions in an unprecedented way. SMARTScompare makes it possible to compare SMARTS patterns independent of their string representation and supports most features of the SMARTS language. It is capable of similarity assessments and can detect subset relations between patterns. Beyond SMARTScompare, we developed the SMARTScompare-Viewer, a tool that provides intuitive depictions of SMARTScompare pattern mappings.

As a first application study, we used our newly developed tools to analyze several published filter sets. For most of the filter sets, publications are available that summarize the purpose for which they were designed. For those filter sets without a publication, we at least had the labels that were given to each pattern contained in the filter sets. This allowed us to analyze the similarities of filter sets against the background of their original purpose. The data suggest that the Glaxo filter set is fully contained within the MLSMR filter set, with only nine patterns that were adjusted. We also can easily see that the SMARTCyp set has a totally different focus compared with the rest of the filter sets. The PAINS filter set consists of many large patterns that seem like they were directly translated from SMILES strings for specific molecules. As a consequence, filters from PAINS are very dissimilar to filters from other sets used in HTS screening. Last but not least, we are able to see differences in design intentions ingrained in the similarity values of the pattern set comparison. Our analysis suggests that the main difference between the Glaxo and SureChEMBL OCHEM/ToxAlerts filter sets is the focus on chemical instability in the first case and the focus on toxicity in the second case.

From an analysis of the factors that have the greatest influence on pattern similarity, pattern size (the number of nodes), the number of atomtypes an element has, and the weighting of bits of node fingerprints emerge as the strongest factors affecting similarity.

Aromaticity poses a problem, as no standard aromaticity detection is included in the SMARTS language. We introduced two optional modes that can be engaged if needed. If in use, aromaticity is handled in a way that makes the comparison of two patterns more independent of their notation.

For cheminformaticians interested in designing filter sets, SMARTScompare offers an easy approach to learn from filter sets and combine rules with own experiences. For chemists interested in understanding why compounds are filtered out, SMARTScompare offers many new opportunities for analysis. With a pattern of interest from one filter set, similar patterns from other sets become searchable, making direct comparisons possible for the first time.

DOI: 10.1021/acs.jcim.9b00249
J. Chem. Inf. Model. 2019, 59, 2572−2586

■ ASSOCIATED CONTENT

Supporting Information

The Supporting Information is available free of charge on the ACS Publications website at DOI: 10.1021/acs.jcim.9b00249.

Additional chapters providing more detailed information about the similarity validation (AdditionalApplicationChapters.pdf), list of molecules (with ZINC IDs) violating hydrogen properties as specified by our NAOMI chemistry model (UnmodeledHydrogenCounts.smi); list of molecules (with ZINC IDs) exceeding the enumeration limit for the minimum ring size (RingSizeEnumerationExceedance.smi); list of molecules (with ZINC IDs) exceeding the enumeration limit for the number of unique ring families of which an atom is a part (UrfCountEnumerationExceedance.smi); SmartsView depictions of our quinone query patterns and all matched patterns from the filter sets, with ChEMBL pattern ID, filter set, and annotation for each pattern (QuinonePatterns.pdf); seven patterns from the PAINS[2,8] filter set that we modified to speed up the calculations (ModifiedPAINSPatterns.csv) (ZIP)

■ AUTHOR INFORMATION

Corresponding Author

*E-mail: rarey@zbh.uni-hamburg.de. Phone: +49 (40) 428387351.

ORCID

Emanuel S. R. Ehmki: 0000-0003-0457-4856

Robert Schmidt: 0000-0002-7089-4842

Matthias Rarey: 0000-0002-9553-6531

Present Address

†F.O.: Evotec AG - Manfred Eigen Campus, Essener Bogen 7, 22419 Hamburg, Germany.

Author Contributions

F.O. and M.R. initially tested the algorithm presented in the companion paper[19] on the pattern sets that are available in ChEMBL. R.S. implemented the SMARTScompare software based on the NAOMI cheminformatics library (current version). E.S.R.E. provided application scenarios for the SMARTScompare approach and analyzed and interpreted the results. E.S.R.E., R.S., and M.R. wrote the manuscript.

Notes

The authors declare the following competing financial interest(s): M.R. declares a potential financial interest in the event that the SMARTScompare software is licensed for a fee to non-academic institutions in the future.

SMARTScompare and SMARTScompareViewer, a tool for visualization of SMARTS relationships, are available for Linux, OS X, and Windows as part of the NAOMI ChemBio Suite (https://uhh.de/naomi) and for free for academic use and evaluation purposes. Furthermore, SMARTScompare and the SMARTScompareViewer are integrated in the SMARTSview serverhttps://smarts.plus. All feedback is greatly appreciated and supports the further development of SMARTScompare.

■ REFERENCES

(1) Gaulton, A.; Bellis, L. J.; Bento, A. P.; Chambers, J.; Davies, M.; Hersey, A.; Light, Y.; McGlinchey, S.; Michalovich, D.; Al-Lazikani, B.; Overington, J. P. ChEMBL: A large-scale bioactivity database for drug discovery. *Nucleic Acids Res.* **2012**, *40*, D1100−D1107.

(2) Gaulton, A.; et al. The ChEMBL database in 2017. *Nucleic Acids Res.* **2017**, *45*, D945−D954.

(3) Weininger, D. SMILES, a Chemical Language and Information System: 1: Introduction to Methodology and Encoding Rules. *J. Chem. Inf. Model.* **1988**, *28*, 31−36.

(4) Daylight Chemical Information Systems, Inc. http://www.daylight.com/dayhtml/doc/theory/theory.smarts.html (accessed Aug 20, 2018).

(5) Ash, S.; Cline, M. A.; Homer, R. W.; Hurst, T.; Smith, G. B. SYBYL Line Notation (SLN): A versatile language for chemical structure representation. *J. Chem. Inf. Comput. Sci.* **1997**, *37*, 71−79.

(6) Hann, M.; Hudson, B.; Lewell, X.; Lifely, R.; Miller, L.; Ramsden, N. Strategic Pooling of Compounds for High-Throughput Screening. *J. Chem. Inf. Model.* **1999**, *39*, 897−902.

(7) Sushko, I.; Salmina, E.; Potemkin, V. A.; Poda, G.; Tetko, I. V. ToxAlerts: A web server of structural alerts for toxic chemicals and compounds with potential adverse reactions. *J. Chem. Inf. Model.* **2012**, *52*, 2310−2316.

(8) Baell, J. B.; Holloway, G. A. New substructure filters for removal of pan assay interference compounds (PAINS) from screening libraries and for their exclusion in bioassays. *J. Med. Chem.* **2010**, *53*, 2719−2740.

(9) Bruns, R. F.; Watson, I. A. Rules for identifying potentially reactive or promiscuous compounds. *J. Med. Chem.* **2012**, *55*, 9763−9772.

(10) Pearce, B. C.; Sofia, M. J.; Good, A. C.; Drexler, D. M.; Stock, D. A. An empirical process for the design of high-throughput screening deck filters. *J. Chem. Inf. Model.* **2006**, *46*, 1060−1068.

(11) Rydberg, P.; Gloriam, D. E.; Zaretzki, J.; Breneman, C.; Olsen, L. SMARTCyp: A 2D method for prediction of cytochrome P450-mediated drug metabolism. *ACS Med. Chem. Lett.* **2010**, *1*, 96−100.

(12) Blake, J. F. Identification and evaluation of molecular properties related to preclinical optimization and clinical fate. *Med. Chem. (Sharjah, United Arab Emirates)* **2005**, *1*, 649−655.

(13) Brenk, R.; Schipani, A.; James, D.; Krasowski, A.; Gilbert, I. H.; Frearson, J.; Wyatt, P. G. Lessons learnt from assembling screening libraries for drug discovery for neglected diseases. *ChemMedChem* **2008**, *3*, 435−444.

(14) Schomburg, K.; Ehrlich, H. C.; Stierand, K.; Rarey, M. From structure diagrams to visual chemical patterns. *J. Chem. Inf. Model.* **2010**, *50*, 1529−1535.

(15) Center for Bioinformatics Hamburg. SMARTSviewServer. https://smartsview.zbh.uni-hamburg.de (accessed Oct 11, 2018).

(16) Bietz, S.; Schomburg, K. T.; Hilbig, M.; Rarey, M. Discriminative Chemical Patterns: Automatic and Interactive Design. *J. Chem. Inf. Model.* **2015**, *55*, 1535−1546.

(17) Schomburg, K. T.; Wetzer, L.; Rarey, M. Interactive design of generic chemical patterns. *Drug Discovery Today* **2013**, *18*, 651−658.

(18) Landrum, G. RDKit: Open Source Cheminformatics. https://www.rdkit.org/ (accessed Sept 26, 2018)

(19) Schmidt, R.; Ehmki, E. S. R.; Ohm, F.; Ehrlich, H.-C.; Mashychev, A.; Rarey, M. Comparing Molecular Patterns using the Example of SMARTS: Theory and Algorithms. *J. Chem. Inf. Model.* **2019**, DOI: 10.1021/acs.jcim.9b00250.

(20) Capuzzi, S. J.; Muratov, E. N.; Tropsha, A. Phantom PAINS: Problems with the Utility of Alerts for Pan-Assay interference CompoundS. *J. Chem. Inf. Model.* **2017**, *57*, 417−427.

(21) Urbaczek, S.; Kolodzik, A.; Rarey, M. The valence state combination model: A generic framework for handling tautomers and protonation states. *J. Chem. Inf. Model.* **2014**, *54*, 756−766.

(22) Zamora, A. An Algorithm for Finding the Smallest Set of Smallest Rings. *J. Chem. Inf. Model.* **1976**, *16*, 40−43.

(23) Kolodzik, A.; Urbaczek, S.; Rarey, M. Unique ring families: A chemically meaningful description of molecular ring topologies. *J. Chem. Inf. Model.* **2012**, *52*, 2013−2021.

(24) Flachsenberg, F.; Andresen, N.; Rarey, M. RingDecomposerLib: An Open-Source Implementation of Unique Ring Families and Other Cycle Bases. *J. Chem. Inf. Model.* **2017**, *57*, 122−126.

(25) ZINC15. https://zinc15.docking.org (accessed Aug 20, 2018).

DOI: 10.1021/acs.jcim.9b00249
J. Chem. Inf. Model. 2019, 59, 2572−2586

(26) Urbaczek, S.; Kolodzik, A.; Fischer, J. R.; Lippert, T.; Heuser, S.; Groth, I.; Schulz-Gasch, T.; Rarey, M. NAOMI: On the almost trivial task of reading molecules from different file formats. *J. Chem. Inf. Model.* **2011**, *51*, 3199−3207.
(27) Urbaczek, S.; Kolodzik, A.; Fischer, J. R.; Lippert, T.; Heuser, S.; Groth, I.; Schulz-Gasch, T.; Rarey, M. NAOMI: On the almost trivial task of reading molecules from different file formats. *J. Chem. Inf. Model.* **2011**, *51*, 3199−3207.
(28) Rogers, D.; Hahn, M. Extended-Connectivity Fingerprints. *J. Chem. Inf. Model.* **2010**, *50*, 742−754.
(29) Ehrlich, H. C.; Rarey, M. Systematic benchmark of substructure search in molecular graphs—From Ullmann to VF2. *J. Cheminf.* **2012**, *4*, 13.
(30) ChEMBL22. http://chembl.blogspot.com/2016/09/chembl-22-released.html (accessed Dec 5, 2018).
(31) Ihlenfeldt, W. D.; Takahashi, Y.; Abe, H.; Sasaki, S. Computation and Management of Chemical Properties in CACTVS: An Extensible Networked Approach toward Modularity and Compatibility. *J. Chem. Inf. Model.* **1994**, *34*, 109−116.
(32) Sushko, I.; et al. Online chemical modeling environment (OCHEM): Web platform for data storage, model development and publishing of chemical information. *J. Comput.-Aided Mol. Des.* **2011**, *25*, 533−554.
(33) Evotec AG. https://www.evotec.com/en (accessed Oct 1, 2018).
(34) The ChEMBL Team (ChEMBL Helpdesk). Personal communication, 2018.
(35) National Institutes of Health. NIH Grand Issue Note MLSMR. https://grants.nih.gov/grants/guide/notice-files/NOT-RM-07-005.html (accessed Oct 1, 2018).
(36) McGovern, S. L.; Helfand, B. T.; Feng, B.; Shoichet, B. K. A specific mechanism of nonspecific inhibition. *J. Med. Chem.* **2003**, *46*, 4265−4272.
(37) Inpharmatica Inc. http://www.inpharmatica.co.uk/ (accessed Oct 1, 2017).
(38) Sushko, I.; Tetko, I. V.; Salmina, E.; Potemkin, V. A.; Poda, G. OCHEM/ToxAlerts Webserver. https://ochem.eu//alerts/show.do?render-mode=full (accessed April 29, 2019).
(39) SureChEMBL ToxAlerts WebPage. https://www.surechembl.org/knowledgebase/169485-non-medchem-friendly-smarts (accessed Feb 18, 2019).
(40) Schorpp, K.; Rothenaigner, I.; Salmina, E.; Reinshagen, J.; Low, T.; Brenke, J. K.; Gopalakrishnan, J.; Tetko, I. V.; Gul, S.; Hadian, K. Identification of Small-Molecule Frequent Hitters from AlphaScreen High-Throughput Screens. *J. Biomol. Screening* **2014**, *19*, 715−726.

DOI: 10.1021/acs.jcim.9b00249
J. Chem. Inf. Model. 2019, 59, 2572−2586

Supporting information for: Comparing Molecular Patterns using the Example of SMARTS: Applications and Filter Collection Analysis

Emanuel S. R. Ehmki,[†] Robert Schmidt,[†] Farina Ohm,[†,‡] and Matthias Rarey[*,†]

[†]*ZBH - Center for Bioinformatics, Bundesstraße 43, 20146 Hamburg, Germany*

[‡]*Current address: Evotec AG - Manfred Eigen Campus, Essener Bogen 7, 22419 Hamburg, Germany*

E-mail: rarey@zbh.uni-hamburg.de

Phone: +49 (40) 428387351

SI1: Validation Experiments of the Similarity Approach

ZINC Molecules

We first took 100 substructure-like patterns from our validation experiment that have approximately 10,000 matches on the *validation data set*. We computed the reference similarity and compared it with the calculated pattern similarity. The correlation is plotted in Figure S1. Overall, we observe highly similar correlation results for the three pattern similarity methods (weighted fingerprint similarity, reduced fingerprint similarity, and fingerprint similarity) and the underlying MCS similarity with squared Pearson correlation coefficients of 0.72, 0.71, 0.7, and 0.66, respectively. The substructure like patterns used to evaluate simi-

larity are equally described by all measures. For all similarity plots we observe that pattern similarity has a tendency to overestimate the similarity value. We conclude that our similarity measure is applicable to substructure like SMARTS patterns and that the similarity implementations do not significantly differ.

S2

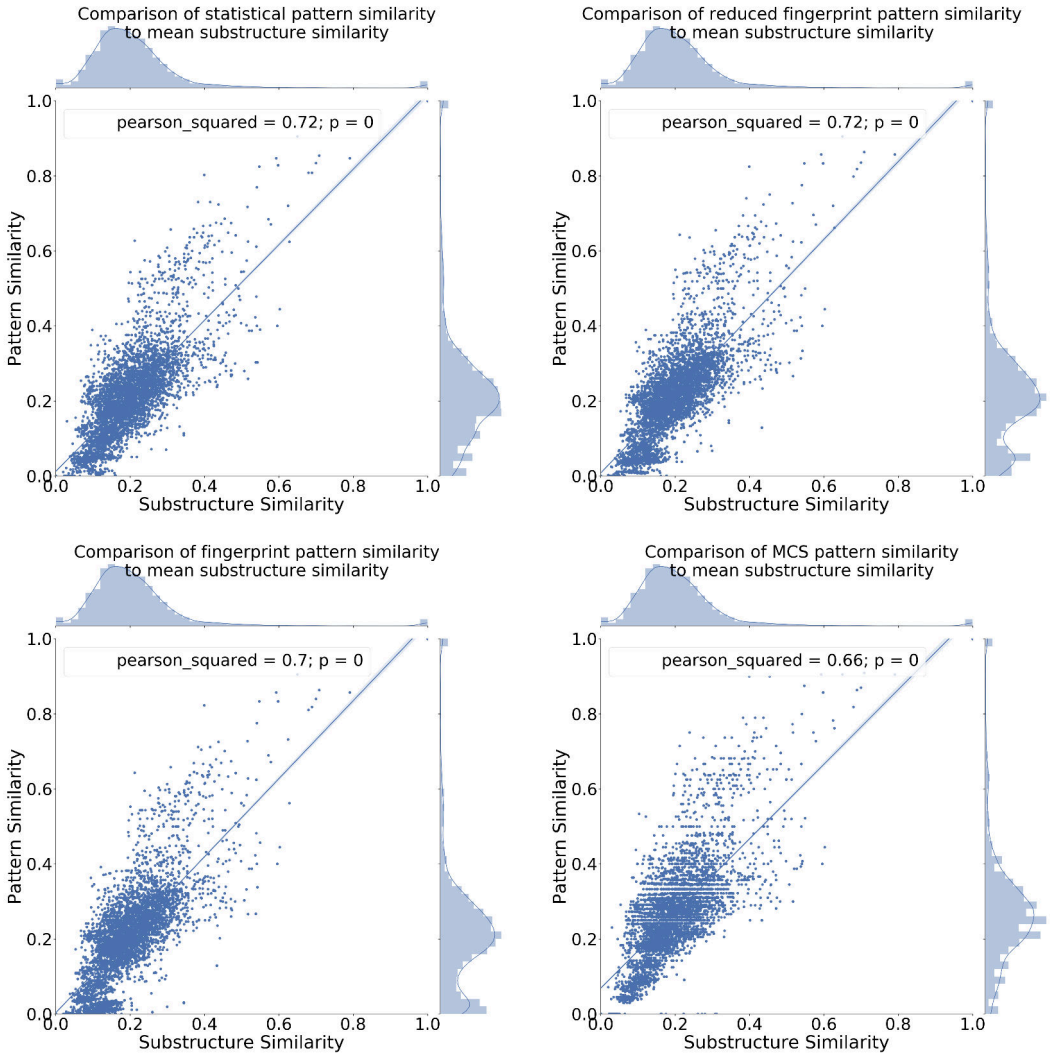


Figure S1: Correlation plot of three similarity measures SMARTScompare provides. For this plot, 100 patterns with 10,000 hits on the *validation data set* were considered. Each plot shows the correlation for the implemented similarity measures, statistical pattern similarity, reduced fingerprint pattern similarity, fingerprint pattern similarity, and the underlying MCS similarity to the reference substructure similarity. All three measures and the MCS similarity have a similar squared Pearson correlation coefficients. The histograms on the axes of the plots show the distribution of similarity values. They emphasize the differences of all three methods at best. We observe similar distributions to the MCS similarity for the three measures.

ChEMBL22 Molecules

We matched all patterns from the filter sets against ≈1.3million molecules from ChEMBL22[S1]. To reduce the computational complexity, we limited the maximum number of matches to 50,000. Patterns above this threshold were not considered. To achieve a reasonable reference similarity, we enforced a minimum of 5,000 hits per pattern to be considered. Also, we omitted any similarity value that is zero for any of those measures. Since they dominated the distribution, the analysis becomes more relevant. The similarity correlation is shown in Figure S2. In this experiment we observe that all our methods underestimate the substructure similarity. Furthermore, the differences between the methods of weighted atom type and fingerprint similarity methods are still not significant. Regarding the squared pearson correlations coefficients of 0.67, 0.65, 0.59, and 0.58, we conclude that our method is also suited to measure pattern similarity. We performed another experiment considering only similarity values of comparisons where at least one pattern has an node or edge which is explicitly marked as acyclic (see Figure S3). Here we observe squared Pearson correlation coefficients of 0.7, 0.64, and 0.49. Similar to Figure S2 most of the values are smaller then 0.4. This plot shows the differences in between the similarity measures best, since most of the extended atom types are related to ring properties. The MCS similarity in this experiment is quite interesting with the regression line close to the line of unity. Since the MCS similarity only depends on the pattern sizes and the size of the mapping, there are only a few similarity values that are actually achieved. This results in several lines representing a certain pattern similarity and the corresponding range for substructure similarity. In all plots the MCS similarity can be understood as upper bound for any other similarity calculated by SMARTScompare.

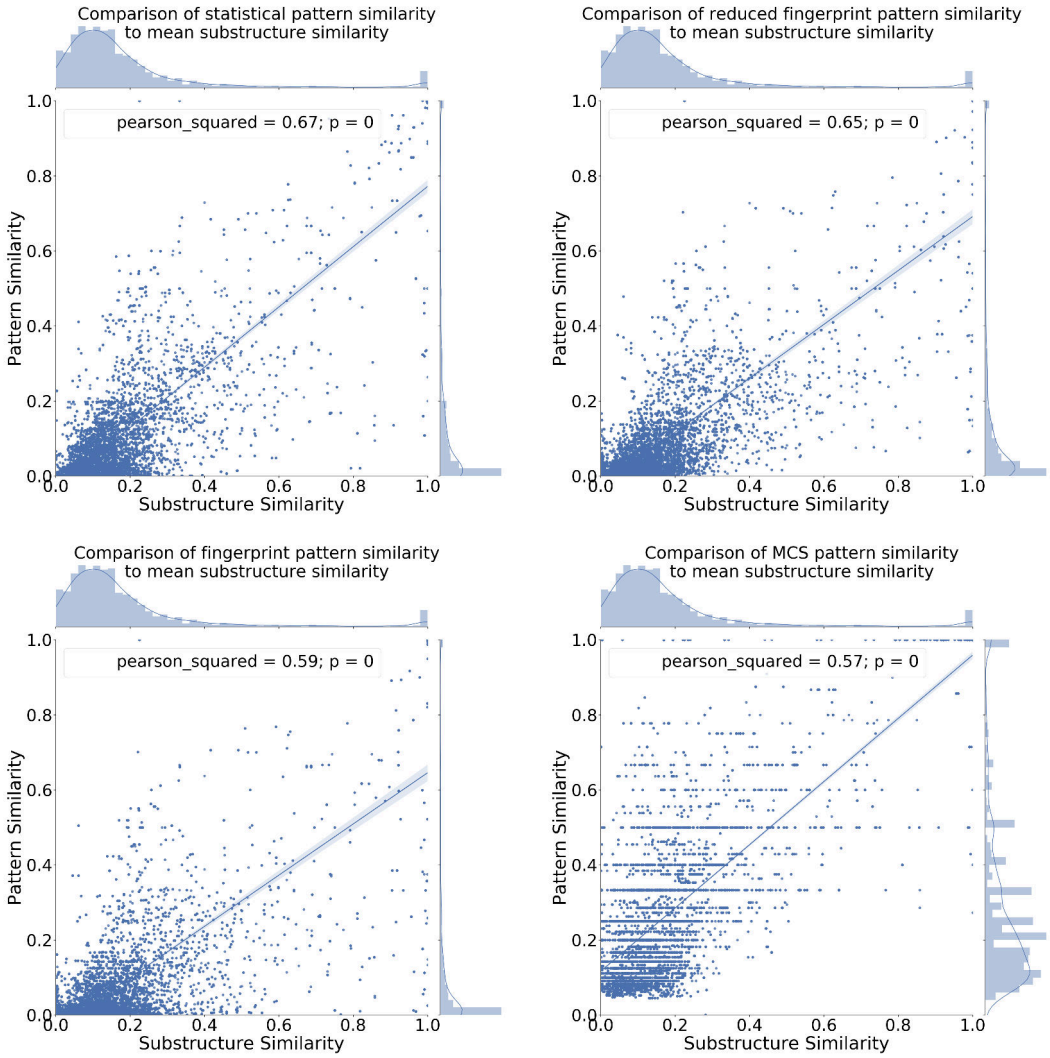


Figure S2: Correlation plot of three similarity measures SMARTScompare provides and the MCS similarity. For this plot, all patterns from the 9 sets were considered matching 5,000 up to 50,000 molecules on ≈1.3 million molecules from ChEMBL22. Values that are zero for any measure are omitted. Each plot shows the correlation for a similarity measure to the reference substructure similarity. In this experiment the MCS similarity shows a different distribution in comparison to the other three measures.

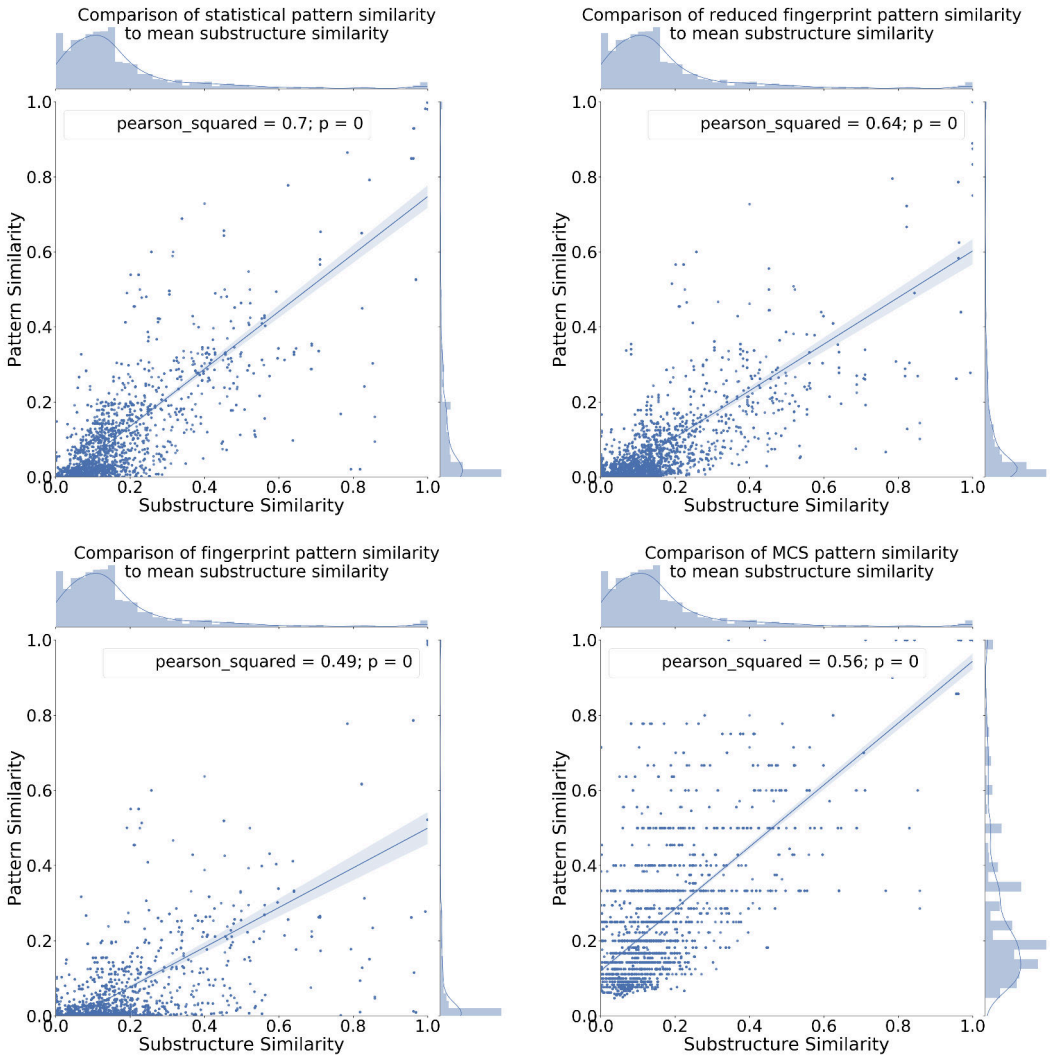


Figure S3: Correlation plot of the three similarity measures SMARTScompare provides. This plot is based on a selection of the data of Figure S2. Only those similarity values were considered, where at least one pattern has an explicit acyclic node or acyclic bond. In this experiment we observe different correlation values as well as distinguishable distributions.

Conclusion

For large substructure like patterns, similarity is dominated by the underlying MCS mapping. This can change if more SMARTS properties and logical formulations are used. Overall, these experiments show that the pattern similarity calculated by SMARTScompare approximates the similarity which can derived from matching fragments to a large reference data set.

References

(S1) ChEMBL22. 2016; http://chembl.blogspot.com/2016/09/chembl-22-released.html, Accessed on: 2018-12-05.

SMARTS.plus - A Toolboxfor Chemical Pattern Design

[D5] Ehrt, Christiane u. a.: SMARTS.plus – A Toolbox for Chemical Pattern Design. 2020. DOI: 10.1002/minf.202000216

https://onlinelibrary.wiley.com/doi/10.1002/minf.202000216

Application Note www.molinf.com

molecular informatics

DOI: 10.1002/minf.202000216

SMARTS.plus – A Toolbox for Chemical Pattern Design

Christiane Ehrt[+],[a] Bennet Krause[+],[a] Robert Schmidt[+],[a] Emanuel S. R. Ehmki,[a] and Matthias Rarey*[a]

Abstract: The number of publications concerning Pan-Assay Interference Compounds and related problematic structural motifs in screening libraries is constantly growing. In consequence, filter collections are merged, extended but also critically discussed. Due to the complexity of the chemical pattern language SMARTS, an easy-to-use toolbox enabling every chemist to understand, design and modify chemical patterns is urgently needed. Over the past decade, we developed a series of software tools for visualizing, editing, creating, and analysing chemical patterns. Herein, we highlight how most of these tools can now be easily used as part of the novel SMARTS.plus web server (https://smarts.plus/). As a showcase, we demonstrate how researchers can apply the web server tools within minutes to derive novel SMARTS patterns for the filtering of frequent hitters from their screening libraries with only a little experience with the SMARTS language.

Keywords: Chemical Patterns · SMARTS Visualization · SMARTS Comparison · Medicinal Chemistry · Filter Collections

Chemical patterns are one of the workhorses of cheminformatics. By describing a generic structural feature of molecules, they are of central importance for classifying and organizing compound collections. In contrast to a classical substructure, a chemical pattern allows logical expressions and atom/bond specifications via properties. Invented in the late 80s by Daylight Information Systems,[1] today, the SMARTS language is the quasi-standard for the description of chemical patterns. Although not complete, SMARTS is very feature-rich allowing chemists to precisely specify a structural pattern they have in mind. Unfortunately, the SMARTS language is quite complex and many researchers struggle in formulating their patterns due to the cryptic nature of SMARTS notations. Furthermore, even for experienced computational chemists, it is sometimes hard to spot errors in SMARTS expressions making their development usually a trial-and-error process.

Over the past decade, we developed a series of software tools supporting researchers in designing and analysing chemical patterns using the SMARTS language. Recently, we developed a web server named SMARTS.plus[2] to circumvent the software installation hurdle making SMARTS analytics available to even occasional users and students. In the following, we will first briefly summarize the functionality, how it can be accessed in SMARTS.plus und will round off with a use case, namely the application of the web server tools to derive novel patterns for the filtering of pan-assay interference compounds.

Although systematic names exist in chemistry, the daily language of chemistry is structure diagrams. Chemical patterns have a lot in common with structure diagrams; roughly spoken, they are just a more generic form. The most important aspect to make chemical patterns comprehensible is therefore an adequate visualization. Following the IUPAC nomenclature for structure diagrams as closely as possible, we carefully designed the graphical depiction of molecules to patterns. Figure 1 shows an example of the resulting SMARTSview image for a complex pattern.[3] Substructural features and structural variances get ascertainable, immediately showing the great value of this approach. Once having had a first look at a SMARTSview image, it becomes evident that a graphical editor is indispensable. Therefore, we developed a powerful graphical editor, SMARTSeditor, as a standalone tool[4] which is available for academic use.[5]

Chemical patterns are mostly generated based on example molecules. There is a class A of molecules having a certain property that a class B of molecules does not have. Often, patterns are designed by continuously monitoring which molecules of class A do not yet match and which of class B do still match. The question arises how this process can be best supported algorithmically. In computer science, several algorithms exist for so-called frequent and contrast pattern mining on graphs.[6] These methods are also applied to molecules (see for example[7]); however, they usually do not end up in SMARTS expressions. Therefore, we developed SMARTSminer[8] as a one-stop solution from sets of molecules to a SMARTS pattern. Although SMARTSminer is not able to make use of all SMARTS features (for example,

[a] C. Ehrt,[+] B. Krause,[+] R. Schmidt,[+] E. S. R. Ehmki, M. Rarey
Universität Hamburg, ZBH – Center for Bioinformatics,
Bundesstraße 43, 20146 Hamburg, Germany
phone: +49 40 42838–7351
fax: +49 40 42838–7352
E-mail: rarey@zbh.uni-hamburg.de

[+] These authors contributed equally to this work.

Application Note www.molinf.com molecular informatics

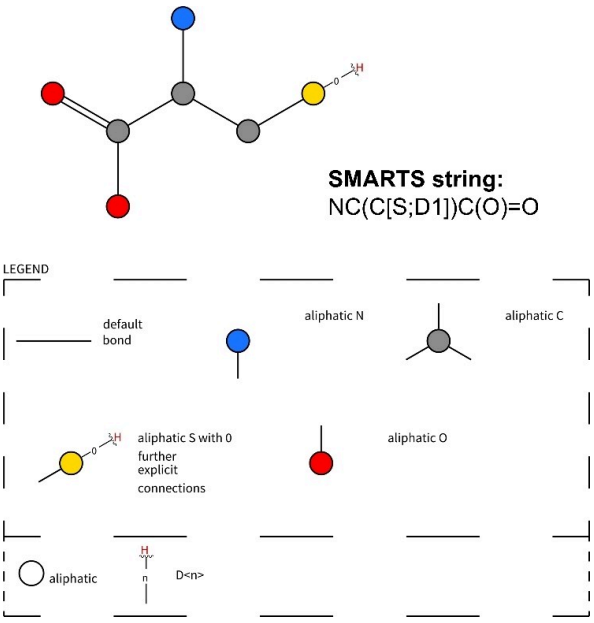


Figure 1. SMARTSview visualization of a typical SMARTS pattern for the exclusion of problematic compounds from molecular datasets. It is extracted from the publication of Pearce and co-workers.[9] The SMARTS pattern describes molecules with a thiol warhead that might covalently modify cysteine residues in a protein in an unselective manner.

recursion is not supported), it simplifies the design of patterns enormously. Within seconds to minutes, it processes sets of hundreds of molecules and creates a SMARTS pattern to enrich or separate them. SMARTSminer is integrated into our standalone editor; to give users a kick-start for pattern design, it is also available on the SMARTS.plus web server.

Comparing two molecules is an almost daily task in cheminformatics. More precisely, we either ask for a substructure relationship (is substructure B contained in molecule A) or a similarity relationship mostly answered with topological fingerprints like the Extended-Connectivity Fingerprints (ECFP).[8] These questions apply to chemical patterns as well and are critical for the analysis of pattern collections. The comparison of SMARTS expressions belongs to the most challenging algorithmic problems in cheminformatics. In 2019, we presented SMARTScompare,[10] an algorithmic approach to address the substructure search and the similarity search on chemical patterns. An atom type fingerprint covering more than 20,000 states is employed in a complex, recursive comparison algorithm.[9] For the first time, a method enables the identification of more specific or generic or just similar chemical patterns in pattern collections - independent of the way they are formulated. SMARTScompare is a command line tool that processes hundreds of pattern comparisons in a few seconds. Recently, we added SMARTScompare to the SMARTS.plus web server for pattern comparison including visualization, as well as for searching public pattern collections.

SMARTS.plus combines SMARTSview, SMARTS-miner and SMARTScompare in an easy-to-use web server based on Rails available at https://smarts.plus. It connects the standalone tools allowing to visually analyse SMARTS expressions. It runs in four modes: In the 'View' mode, the user can enter a SMARTS expression and get a visual representation including a figure legend for less experienced users. In the 'Compare' mode, two expressions can be uploaded and the server calculates substructure relationships and pattern similarity. A graphical depiction of the node mapping helps to comprehend the pattern relationship. In the 'Search' mode, a SMARTS expression is compared to currently nine public pattern collections enabling to browse through the most similar ones. To this end, we used the SMARTS collections as applied for the annotation by the ChEMBL database.[11] These collections include the well-known PAINS filters (PAINS),[12] the SureChEMBL Non-MedChem Friendly SMARTS (SureChEMBL),[13] the Bristol-Myers Squibb HTS Deck filters (BMS),[9] the NIH MLSMR Excluded Functionality filters (MLSMR),[14] the University of Dundee NTD Screening Library Filters (Dundee),[15] filters of unwanted fragments derived by Inpharmatica Ltd. (Inpharmatica),[5] the Pfizer lint filters (lint),[16] and the Glaxo Wellcome Hard filters (Glaxo).[17] Finally, in the 'Create' mode, two compound sets can be uploaded and SMARTS-miner is applied to suggest patterns frequently found in the first set and rarely found in the second. A browser shows the molecules hit for each of the created patterns.

In the following, we describe a workflow (Figure 2) for applying the SMARTS.plus tools to derive novel SMARTS filters for the characterization of frequent hitters or pan-assay interference compounds, i.e., compounds that are frequently found to be active in multiple high throughput screening assays, e.g. due to aggregation or high reactivity. As an example case study, we selected a set of molecules which might unselectively modify cysteine residues in proteins according to the studies of Dahlin et al.[18] The common structural feature of these compounds is the benzodiathiazole scaffold (using the link https://smarts.plus/ you can visualize the corresponding SMARTS pattern: '[*,#1]-[#6]-1=[#6]-[#6](-[*,#1])=[#6](-[*,#1])-[#6]-2=[#7]-[#16]-[#7]=[#6]-1-2'). Whereas six of the analysed compounds were shown to react covalently by monitoring the presence of compound thiol adducts after addition of CoA, 12 further compounds sharing this scaffold did not lead to covalent adducts. SMARTSminer was used to derive a chemical SMARTS pattern that enables differentiation between the positive support structures (thiol-modifying compounds) and the negative support structures (no reaction with free thiols was observed). This can be achieved in the 'Create' mode by uploading both sets to the web server and defining the minimum percentage of

Application Note www.molinf.com molecular informatics

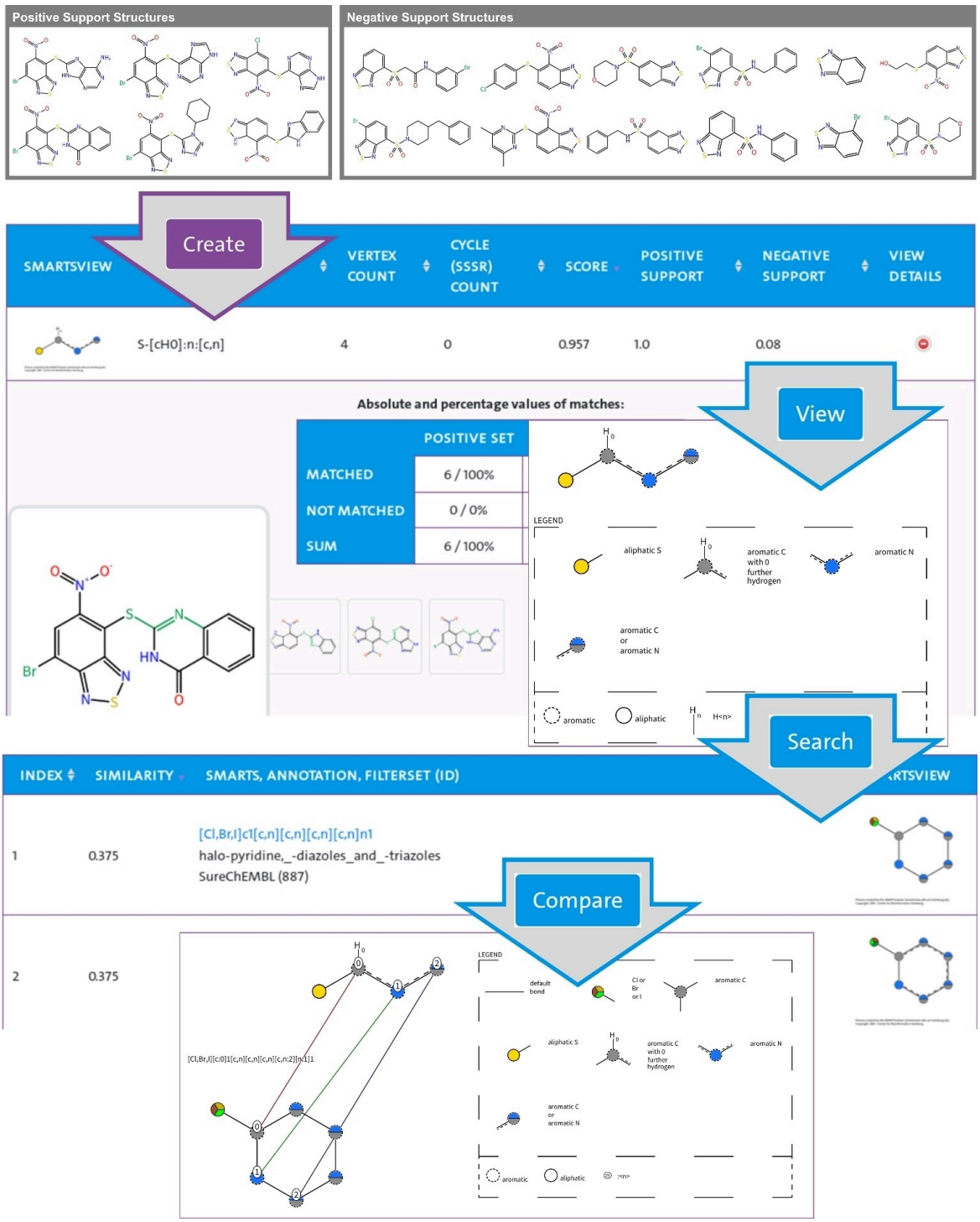


Figure 2. An example workflow utilizing the SMARTS.plus tools in the 'Create', 'View', 'Search', and 'Compare' mode. Here, we show how two molecule sets can be compared creating a SMARTS pattern that enables a good differentiation between both sets. The derived pattern can be immediately visualized in the 'View' mode. Subsequently, multiple filter collections can be searched for similarities to the created pattern. The differences between the pattern of interest and highly similar patterns can finally be visualized in the 'Compare' mode.

Application Note www.molinf.com molecular informatics

compounds from the positive support structures (90% in this example) and the maximum percentage of compounds from the negative support structures (10% in this example) that should be matched by the generated SMARTS pattern. The resulting patterns can be visualized with SMARTSview by one click ('View' mode). However, our search resulted in numerous patterns with identical positive and negative support leading to identical confusion matrices.

To further filter the results, the option 'Small patterns' can be selected on the bottom of the results page. In case of multiple patterns with the same support values, small patterns are preferred, large ones eliminated. Here, this leads to a reduction of the results to three SMARTS patterns. The chosen pattern can subsequently be compared to already available SMARTS filter collections to find potential patterns already covering the compounds of interest.

This can be achieved in the 'Search' mode where we can insert the SMARTS pattern of interest and compare it to all patterns within the previously mentioned SMARTS filter collections: PAINS, SureChEMBL, BMS, MLSMR, Dundee, Inpharmatica, lint, and Glaxo. In our example, the maximum similarity is 0.375; so we might conclude that we found a novel SMARTS pattern for the characterization of typical frequent hitters. However, it is often difficult to compare the most similar patterns by eye. Therefore, we can finally use the 'Compare' mode to get a better understanding of the basic differences between the newly created and the most similar pattern. We can click on the most similar SMARTS string and a graphical representation of the differences is issued. Now, we can immediately see that the previously defined pattern of the MLSMR collection is missing the sulfur atom which is crucial in our newly designed SMARTS pattern. In consequence, we were able to derive a novel structural pattern for the filtering of unselectively reacting compounds in screening datasets based on experimental data.

The described workflow can be pursued on our SMARTS.plus web server within minutes. It enables interested researchers to derive appropriate conclusions based on their experimental results in a highly intuitive way. We hope that current efforts to derive new SMARTS patterns for the description of frequent hitters will benefit from our freely available web server.

Acknowledgement

Open access funding enabled and organized by Projekt DEAL.

Conflict of Interest

None declared.

References

[1] Daylight Chemical Information Systems, Inc., https://www.daylight.com/dayhtml/doc/theory/theory.smarts.html (accessed Sep 30, 2020).
[2] Center for Bioinformatics Hamburg. SMARTS.plus, https://smarts.plus/ (accessed Sep 30, 2020).
[3] K. T. Schomburg, H. C. Ehrlich, K. Stierand, M. Rarey, *J. Chem. Inf. Model.* **2010**, *50*, 1529–1535.
[4] K. T. Schomburg, L. Wetzer, M. Rarey, *Drug Discovery Today* **2013**, *18*, 651–658.
[5] Center for Bioinformatics Hamburg, NAOMI ChemBio Suite, uhh.de/naomi (accessed Sep 30, 2020).
[6] C. Jiang, F. Coenen, M. Zito, *The Knowledge Engineering Review* **2012**, *28*, 75–105.
[7] C. Borgelt, M. R. Berthold, in *2002 IEEE International Conference on Data Mining, 2002. Proceedings.*, **2002**, pp. 51–58.
[8] S. Bietz, K. T. Schomburg, M. Hilbig, M. Rarey, *J. Chem. Inf. Model.* **2015**, *55*, 1535–1546.
[9] B. C. Pearce, M. J. Sofia, A. C. Good, D. M. Drexler, D. A. Stock, *J. Chem. Inf. Model.* **2006**, *46*, 1060–1068.
[10] R. Schmidt, E. S. R. Ehmki, F. Ohm, H. C. Ehrlich, A. Mashychev, M. Rarey, *J. Chem. Inf. Model.* **2019**, *59*, 2560–2571.
[11] A. Gaulton, A. Hersey, M. Nowotka, A. P. Bento, J. Chambers, D. Mendez, P. Mutowo, F. Atkinson, L. J. Bellis, E. Cibrian-Uhalte, M. Davies, N. Dedman, A. Karlsson, M. P. Magarinos, J. P. Overington, G. Papadatos, I. Smit, A. R. Leach, *Nucleic Acids Res.* **2017**, *45*, D945–D954.
[12] J. B. Baell, G. A. Holloway, *J. Med. Chem.* **2010**, *53*, 2719–2740.
[13] SureChEMBL, https://www.surechembl.org/knowledgebase/169485-non-medchem-friendly-smarts (accessed Sep 30, 2020).
[14] MLSMR, https://www.yumpu.com/en/document/view/12367541/mlsmr-excluded-functionality-filters-nih-molecular-libraries- (accessed Sep 30, 2020).
[15] R. Brenk, A. Schipani, D. James, A. Krasowski, I. H. Gilbert, J. Frearson, P. G. Wyatt, *ChemMedChem* **2008**, *3*, 435–444.
[16] J. F. Blake, *Med. Chem.* **2005**, *1*, 649–655.
[17] M. Hann, B. Hudson, X. Lewell, R. Lifely, L. Miller, N. Ramsden, *J. Chem. Inf. Comput. Sci.* **1999**, *39*, 897–902.
[18] J. L. Dahlin, J. W. Nissink, J. M. Strasser, S. Francis, L. Higgins, H. Zhou, Z. Zhang, M. A. Walters, *J. Med. Chem.* **2015**, *58*, 2091–2113.

Received: August 18, 2020
Accepted: September 28, 2020
Published online on October 8, 2020

Eidesstattliche Versicherung

Hiermit erkläre ich an Eides statt, dass ich die vorliegende Dissertationsschrift selbst verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Siegburg, den _____

_____

Robert Julius Schmidt