Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

Fakultät für Betriebswirtschaft
Institut für Operations Management

Project Scheduling with Activity Exchange

Kumulative Dissertation zur Erlangung der Würde des
„Doctor rerum oeconomicarum (Dr. rer. oec.)"
nach der Promotionsordnung vom 9. Juli 2014 ff.

an der Fakultät für Betriebswirtschaft
Moorweidenstr. 18
20148 Hamburg/Deutschland
der Universität Hamburg

vorgelegt von: Dipl.-Math. Lukas David Berthold
geboren am 20.10.1982 in Gleiwitz

Hamburg, den 13.12.2021

Vorsitzender:	Prof. Dr. Martin Spindler
Erstgutachter:	Prof. Dr. Malte Fliedner
Zweitgutachter:	Prof. Dr. Wolfgang Brüggemann

Datum der Disputation: 07.04.2022

Summary

Motivated by a practice in aircraft maintenance, in this thesis we study models of project scheduling with exchange operations between activities. Here the idea is to move work content within the project network which admits to move work from critical positions to less critical ones. This admits reducing project duration and meeting due dates.
The first paper adresses questions of complexity and identifies NP-hard as well as tractable cases. In the second paper a model is studied where this exchange procedure is fully implemented and a branch-and-bound approach is developed to solve this problem exactly. Within a computational study the performance of this procedure is demonstrated. The third paper adresses the aspect of uncertainty which occurs very naturally in a maintenance context. Since the stochastic nature also increases complexity we develop a heuristic approach and study its behaviour in a comprehensive computational study.
In a broader setting this work can be also considered as a contribution to the growing field of flexible project scheduling, for which the classical assumption of a fixed project structure is relaxed in a novel way.

Zusammenfassung

Motiviert durch eine Praxis aus der Instandsetzung von Flugzeugen, befasst sich diese Arbeit mit Projekt-Scheduling-Modellen mit Austauschoperationen für Aktivitäten. Die Idee hierbei ist es, Arbeitsaufwände innerhalb des Projektnetzwerks zu verlagern und somit Arbeitsinhalte von zeitkritischen hin zu weniger zeitkritischen Stellen zu verschieben. Dadurch kann die Projektdauer reduziert und Zeitvorgaben können eingehalten werden.
Das erste Paper beschäftigt sich mit Fragen der Komplexität und identifiziert NP-schwere sowie in Polynomialzeit lösbare Fälle. Im zweiten Paper wird ein Modell untersucht bei dem die Austauschoption weitergehend umgesetzt wird und ein Branch-and-Bound-Ansatz entwickelt wird um das Problem exakt zu lösen. Im Rahmen einer Rechenstudie wird die Leistungsfähigkeit des Verfahrens nachgewiesen. Das dritte Paper greift den Aspekt der Unsicherheit auf wie er ganz natürlich im Instandsetzungskontext auftritt. Da die stochastische Natur aber auch die Komplexität des Problems erhöht, wird ein heuristischer Lösungsansatz entwickelt, dessen Lösungsverhalten im Rahmen einer umfangreichen Rechenstudie untersucht wird.
In einem umfassenderen Kontext kann die Arbeit auch als Beitrag zum wachsenden Forschungsfeld des Schedulings von flexiblen Projekten betrachtet werden, wofür die klassische Annahme einer festen Projektstruktur auf eine neuartige Weise relaxiert wird.

Contents

Part
Synopsis

Synopsis

1. Introduction

The term maintenance comprises all measures to keep technical equipment operable [Pintelon and Gelders 1992]. This includes preventive activities, which keep a technical system in good condition to prevent failures, as well as corrective ones, which are carried out if failures have occured [Wang 2002]. Thus a proper maintenance is essential to guarantee a long-lasting functionality of a technical system. But maintenance is also usually associated with significant cost and furthermore makes downtimes necessary, such that careful planning is required to keep maintenance as short, cheap and effective as possible. Besides this practical importance the planning of maintenance activities has also garnered theoretical attention and maintenance models have been studied for decades [Wang 2002]. However, a large amount especially of earlier models considers systems consisting of single units or several identical components, which limits their applicability to more complex systems [Cho and Parlar 1991].
Indeed, the maintenance of more complex systems like aircraft or power plants often has the character of a project. For aircraft, for example, where several maintenance cycles exist, this especially holds for heavy maintenance C- and D-checks [Samaranayake 2006]: It consists of a large number of activities which have to be carried out within a given time frame under consideration of limited budget and other resources to reach the intended goal, namely to bring the system into a fully operable state again. The complexity of such a maintenance project is further increased by the different aspects like personnel, inventory and activity schedules which must be coordinated; a significant number of repair activities, for instance, will depend on the outcomes of prior inspections which might lead to considerable uncertainty at the time of planning. Thus performing and managing a maintenance project requires a careful planning. However, the number of project scheduling models genuinely motivated by maintenance is rather limited, see for example [McKendall et al. 2008, Megow et al. 2011, Masmoudi and Hait 2012].
Project scheduling is a well-studied area as well, especially the literature on the Resource Constrained Project Scheduling Problem (RCPSP) and its variants is rather extensive [Hartmann and Briskorn 2010, Schwindt et al. 2015]. However, one drawback of many variants that has been noted is their rigid project structure: The set of activities to be carried out is fixed, as well as the precedence relations between them; furthermore an activity can be only carried out if all its predecessors haven been finished. For many contexts this has turned out to be too strict and especially in the last decade a number of project scheduling models have been proposed which admit more flexibility [Kellenbrink and Helber 2015, Servranckx and Vanhoucke 2019].
Most models which extend project scheduling towards more flexibility rely on logical operators which admit the modelling of alternative paths, subgraphs etc. On the one hand this has the advantage to have a modeling tool for many situations where there are alternative ways to carry out parts of the project; on the other hand such modelling may turn out too generic in some other situations. As a case in point, the component exchange described below cannot be efficiently captured by logical operators. The models studied in the following are motivated by a component exchange procedure called "cannibalization" which is rather common especially in aircraft maintenance [Banghart 2017, Salman et al. 2007]: Given an aircraft under maintenance, assume some component has found to be defective such that repairing it or waiting for a spare part would take too long to finish the maintenance project in time, which would cause significant cost due to cancelled flights, for example. Further assume there is another aircraft in maintenance containing a component of the same type which is intact; moreover this component is at some position which is not time-critical. In such a situation both components may be removed and swap their positions. As a result, the first aircraft gets an intact component such that its maintenance can be finished in time whereas the defective component does not lead to significant delays in maintenance of the second aircraft given that its position is less critical there.

The overarching goal of this thesis is to study the effects of integrating component exchances into

maintenance planning on the planning structure and to generate insights for developing planning procedures that take advantage of component exchanges. In particular, three separate albeit interrelated research questions are investigated in three dedicated articles:

1. Given that the ability to exchange components provides additional flexibility in maintenance plannng, how does this affect the runtime complexity of automated planning procdures when determining the makespan of maintenace projects?

2. Given that an exchange of components requires facilitating access to both of the components that are supposed to be swapped and thus resources need to be invested, e.g. to disassemble or prepare the maintenance objects prior to the exchange, how should resources be coordinated and what are the potential gains of such a coordination in terms of makespan reduction?

3. Given that the exact status of components might be unknown prior to the maintenance activitites, how does this uncertainty change the structure of the planning problem and by which planning policies can this uncertainty be addressed?

The research articles dedicated to each of the questions and their main findings are summarized in the following section.

2. Papers

In the following we give an overview on the topics studied in this thesis; more information and literature on maintenance (especially aircraft maintenance) and (stochastic) project scheduling can be found in the corresponding papers. Each of the papers adresses a special issue, which can be captured by the notions of flexibility - exchange - uncertainty. On the methodological side, they comprise complexity analysis of a relaxed model, a deterministic model solved by an exact solution approach, and a stochastic model solved by an heuristical approach.

1. A problem which already requires some flexibility and which can be also considered as a relaxed version of the models with full exchange is studied in the first paper "Optimal spare part utilization in maintenance projects and its complexity". Here we assume a maintenance project to be given by a project network with some activities representing repair work of defective components of different types. Further there are assumed to be some intact spare components available (for each type) in stock which can replace a corresponding number of defective components. In case of replacement the repair becomes superfluous and the corresponding activity is set to a duration of 0. Now it has to be decided which defective components are to be replaced to minimize project duration, i.e. makespan. We call this the "Project Activity Insertion Problem" (short: Insertion Problem) and study its computational complexity according to a parameter scheme involving the number of types, of positions for each type and of available spare components for each type. This gives a complete classification and in some sense a complexity landscape of this problem.
Interestingly and somewhat unexpectedly, the problem turns out to be hard for rather restricted conditions, so it is strongly NP-hard already for one type and repair time equal to 2. On the other hand for repair time equal to 1 this case becomes solvable in polynomial time by applying results from so-called Sperner Theory, which studies extremal properties of partially ordered sets [Engel 1997].
Even if the problem is natural in a maintenance settings and hence of own interest, the results can be transferred to the component exchange setting studied in the following papers. Indeed, the insertion problem can be reduced to the component exchange problem by assuming exchanges by using virtual activity nodes for the stock. This implies that project scheduling with activity exchange, as it is studied in the following paper, is already hard under rather restricted conditions, especially with limited flexibility w.r.t. exchange and without resources.

2. The second paper "Project Scheduling with Activity Exchange motivated by Maintenance Operations" presents a model which integrates component exchange into a project scheduling model. This is realized by representing component positions by so-called "flexible" activities which can have duration zero (if the component is intact and hence there is nothing to do at this position) and non-zero duration (if it is defective and must be repaired). It is assumed that the functional states of the components are known in advance which can be realized if the components are monitored by a diagnostic system. For specification of the problem also availability of the components must be taken into account; indeed, usually a component is not immediately accessible but typically becomes available after other activities have been completed (disassembly, for example). This is modelled in such a way, that a component becomes available (for exchange) if the corresponding activity becomes eligible (i.e. all its predecessors have been finished). Moreover resources are taken into account, which altogether yields an extension of the Resource Constrained Project Scheduling Problem (RCPSP).
To solve this problem, a Branch-and-Bound is developed based on a branching with so-called delaying alternatives and a dominance rule with cuts sets as introduced in [Demeulemeester and Herroelen 1992] for the RCPSP. The performance of this approach is demonstrated in a comprehensive computational study where a comparison with a standard solver is carried out. Further attention is given to the dependency on instance parameters in some detail.

3. Taking also uncertainty into account is the topic of the third paper "Stochastic Activity Exchange Project Scheduling" where a stochastic variant of the deterministic problem considered before is studied. Indeed, as the functional state for many components is not initially known but is revealed only during direct inspection, it is modelled here as random variables for each component position; the defect probability is assumed to be the same for all components of the same type. Due to the stochastic nature of the problem, a solution is given now as a scheduling policy instead of a schedule, i.e. for each possible decision point a unique decision is specified. Such scheduling policies are regarded for other stochastic scheduling problems like the stochastic RCPSP (SRCPSP) [Stork 2001]. However, since all scheduling policies form a very large solution space (as already one scheduling policy alone may be of exponential size), optimizing over all policies is usually considered to be intractable (so for the SRCPSP [Stork 2001, p. 2]). Accordingly, we apply a heuristic approach for which the effect of exchanges to the objective is taken into account, and study the improvement compared with the case where no exchange is carried out. For this we regard two objectives, makespan and tardiness, and again the computational study gives an overview on the effect of different parameter settings. For example, it turns out that the tardiness objective admits larger improvements and generally it is more favourable to have fewer component types with more positions than more types with only a moderate number of positions.

3. Conclusion

Motivated by a component exchange procedure in maintenance projects, this thesis presents and investigates project scheduling models which integrate this feature. Focusing on different aspects like complexity, flexibility and uncertainty, this leads to different models, making the application of different methodologies and solution approaches necessary. Even if the deterministic problem turns out to be hard under rather restricted assumptions, it can be solved for moderate-sized instances within reasonable time (however depending on instance parameters). A turn to uncertainty leads to a stochastic model requiring a very different approach.
From a more comprehensive point of view, the models and results can be regarded as a contribution to the growing field of project scheduling with flexibility, albeit coming here from a more specific application instead from a more general setting; both aspects are also vital for the development of Operations Management in general. This also points to the fact, that considering and analysing real

projects (not only from maintenance) may be a source of operations leading to novel decision problems which are not yet captured by standard models and established variations.

4. References

Banghart, M. (2017): Identification of Reverse Engineering Candidates utilizing Machine Learning and Aircraft Cannibalization Data. International Journal of Aviation, Aeronautics, and Aerospace 4(4) Art. 5

Cho, D.I., and M. Parlar (1991): A survey of maintenance models for multi-unit systems. European Journal of Operational Research 51, 1-23

Demeulemeester, E., and W. Herroelen (1992): A Branch-and-Bound Procedure for the Multiple Resource-Constrained Project Scheduling Problem. Management Science 38(12), 1803-1818

Engel, K. (1997): Sperner Theory. Cambridge University Press

Hartmann, S., and D. Briskorn (2010): A survey of variants and extensions of the resource-constrained project scheduling problem. European Journal of Operational Research 207, 1-14

Kellenbrink, C., and S. Helber (2015): Scheduling resource-constrained projects with a flexible project structure. European Journal of Operational Research 246, 379-391

Masmoudi, M., and A. Hait (2012): Fuzzy uncertainty modelling for project planning: application to helicopter maintenance. International Journal of Production Research 50(13), 3594-3611

McKendall, A.R., J.S. Noble, and C.M. Klein (2008): Scheduling Maintenance Activities During Planned Outages At Nuclear Power Plants. International Journal of Industrial Engineering 15, 53-61

Megow, N., R.H. Möhring, and J. Schulz (2011): Decision Support and Optimization in Shutdown and Turnaround Scheduling. INFORMS Journal on Computing 23, 189-204

Pintelon, L.M., and L.F. Gelders (1992): Maintenance management decision making. European Journal of Operational Research 58, 301-317

Salman, S., C.R. Cassady, E.A. Pohl, and S.W. Ormon (2007): Evaluating the Impact of Cannibalization on Fleet Performance. Quality and Reliability Engineering International 23, 445-457

Samaranayake, P. (2006): Current Practices and Problem Areas in Aircraft Maintenance Planning and Scheduling - Interfaced/Integrated System Perspective. Proceedings of the 7th Asia Pacific Industrial Engineering and Management Systems Conference 2006, Bangkok, Thailand (APIEMS 2006)

Schwindt, C., and J. Zimmermann (eds.) (2015): Handbook on project management and scheduling. Springer, Berlin

Servranckx, T., and M. Vanhoucke (2019): A tabu search procedure for the resource-constrained project scheduling problem with alternative subgraphs. European Journal of Operational Research 273, 841-860

Stork, F. (2001): Stochastic Resource-Constrained Project Scheduling. Ph.D. Thesis, Technische Universität Berlin, Germany

Wang, H. (2002): A survey of maintenance policies of deteriorating systems. European Journal of Operational Research 139, 469-489

Auflistung der eigenen Arbeitsanteile

Berthold, L., and M. Fliedner: Optimal spare part utilization in maintenance projects and its complexity

Durchführung: 75%

Schriftliche Abfassung: 100%

Berthold, L.: Project Scheduling with Activity Exchange motivated by Maintenance Operations

Durchführung und schriftliche Abfassung: 100%

Berthold, L.: Stochastic Activity Exchange Project Scheduling

Durchführung und schriftliche Abfassung: 100%

Part I
Optimal spare part utilization in maintenance projects and its complexity

Optimal spare part utilization in maintenance projects and its complexity

Lukas Berthold, Malte Fliedner

Abstract: The problem treated in this paper is motivated by scheduling activities within a maintenance project (involving an aircraft, for example). Some of these activities are assumed to be related to defective components which must be either repaired or replaced by intact components of the same type. Whereas we assume that replacement takes negligible time the duration of a repair activity takes a time which depends on the component type. However, only a limited number of intact components is available in stock. Then the question is which defect components should be replaced to minimize the project duration. A corresponding model is proposed and a detailed study of its computational complexity is performed.

1 Introduction and literature review

The decision problem studied in this paper is motivated by maintenance projects as they are carried out especially for complex technical systems like aircraft [Samaranayake and Kiridena 2012]. Such a maintenance project typically consists of many diverse activities some of which concern defective components, which may have been detected by a diagnostic system, for example. In order to restore a proper functional state, each defective component must either be repaired or otherwise be replaced by a functionally identical component. Whereas repair of a component takes some time which is assumed here to depend on the type of the component, replacement by an intact component is supposed to be done in negligible time, i.e. time zero. Thus, to minimize project duration, the latter alternative is always advantageous; however, for each type we assume only a limited number of components available in stock, typically fewer than are defective. Accordingly the question arises which defect components are to replaced to get a project of minimal duration.

Although to our knowledge the problem sketched above and described in more detail in the next section has not been studied elsewhere, there are several more or less related lines of research in the literature.

Technical systems are subject to deterioration; due to the resulting necessity to keep technical equipment in operable state, decision problems concerning maintenance have been extensively studied since decades [Pierskalla and Voelker 1976, Cho and Parlar 1991, Budai et al. 2008]. Earlier work especially focused on stochastically failing systems and tried to find maintenance policies to minimize costs or to increase reliability. These maintenance policies could involve repair, replacement of components, cannibalization (see also below) etc.; for a comprehensive overview on maintenance policies see [Wang 2002]. However, a large amount of work in this direction deals with single-unit systems or multi-unit systems composed from identical components arranged in parallel and/or in series whence its applicability to real complex systems is limited. A model where such ideas are actually applied to a more realistic setting is given by [Regattieri et al. 2015] for aircraft maintenance: Here aircraft are modeled as systems consisting of a few components which can fail according to type-specific stochastic processes. Then the goal is to determine an optimal mix of maintenance operations for repair or replacement to minimize total costs, which include costs for spare parts, repair activities, downtime etc.

More comprehensive maintenance models have to take into account different aspects like inventory of spare parts [Regattieri et al. 2005], assigning personnel to tasks or shifts [De Bruecker 2015, Chen et al. 2017] and also questions of scheduling (see below). Due to the complexity of an approach integrating all of them, proposed models usually take only some of those aspects into account. Corresponding to the model studied in the following, we will restrict to scheduling aspects here. On the one hand, "maintenance scheduling" can refer to the schedule of time windows for maintenance. Indeed, since

maintenance usually makes it necessary to interrupt productive activity of a system, time intervals for maintenance have to be planned between operational tasks. Scheduling maintenance slots between production tasks has been considered in a general machine setting [Bock et al. 2012, Grigoriu and Briskorn 2017], leading to extensions of classical machine schedule problems by maintenance tasks. In a more specific setting of aircraft operations, maintenance visits (beside the time also the location, i.e. a maintenance station at some airport) have to be scheduled between flights to fulfill different maintenance cycles as prescribed by official regulations, a problem also called aircraft maintenance routing [Sriram and Haghani 2003, Safaei and Jardine 2018].
On the other hand, maintenance scheduling may also refer to scheduling of maintenance activities, i.e. what has to be done within those maintenance windows. This especially concerns complex systems like aircraft, power plants or ships, where the maintenance procedures have the character of a maintenance project [Rosales et al. 2014, McKendall et al. 2008]. Indeed, besides the multitude of activities which must be carried out, further characteristics of a project include a variety of aspects which must be taken into account (personnel, equipment, inventory), a considerable portion of unexpected findings due to inspection, and limited resources. All these require a careful planning and management to finish a maintenance project within given time frames to keep downtime of the system as short as possible while ensuring high standards w.r.t. safety and operability [Samaranayake 2012].
Even if project scheduling has been studied extensively as well (see for example the surveys [Brucker et al. 1999, Hartmann and Briskorn 2010] and books [Artigues et al. 2008, Schwindt et al. 2015]), there are only few models at the interface of both fields. Among them, in [Masmoudi and Hait 2012] an extension of the RCPSP with fuzzy sets is studied, taking into account the practical difficulty to estimate probability distributions for activity durations. Another model taking into account also the skills of technicians is studied in [Li and Womer 2009]. Project Scheduling models were also proposed for the maintenance of power plants [Megow at al. 2011] and mining activities [Chen 1994].
Even if the decision problem described at the beginning is rather natural in the context of maintenance projects and hence of interest for its own sake, it is also motivated by a component exchange operation relatively common especially in aircraft maintenance, usually called "cannibalization" [Salman et al. 2007, Banghart 2017]: Assuming that a component is associated with a different amount of work depending on its functional state (defective/intact), the idea is to interchange components of the same type but different functional states between different positions. As a consequence, an exchange of components leads to an exchange of tasks within the maintenance project such that work can be moved from critical positions to less critical ones. Scheduling models integrating this component exchange option are studied in the following papers [Berthold 2021a, Berthold 2021b], where also more references and background especially on aircraft maintenance and corresponding models can be found.
- Compared with the model studied there the model here can be regarded as a special case of the exchange procedure, namely where there are additional component positions (given by the stock) which contain intact components. As a consequence, the results concerning complexity proven here can be also applied to the exchange case.
Furthermore there are some structural links to problems studied in so called Sperner Theory [Engel 1997]. Sperner theory deals with extremal properties of subsets in partially ordered sets. For example, a classical result in this area is Dilworth's theorem stating that the decomposition in a minimum number of chains is equal in size to a maximum anti-chain [Dilworth 1950]. A first link between the scheduling problem studied here and Sperner Theory simply comes from the fact that the precedence relation of a project network forms a directed acyclic graph which in turn induces a partially ordered set via (indirect) precedence. Furthermore and more essentially, the decision at which positions intact components are to be inserted is reflected by the choice of elements of the partial order for a subset with suitable properties, such that there is a natural translation from one setting to the other one. This correspondence will especially admit to establish a special case solvable in polynomial time.

2 Example and problem definition

For an easier understanding of the following problem definition we start with an example: Let a maintenance project be represented by the following modified project network:

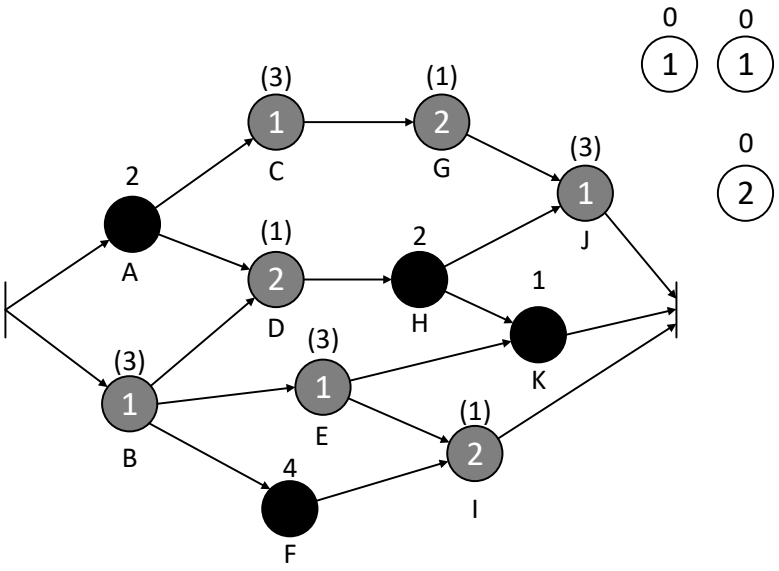


Figure 1: Maintenance project network

Black nodes correspond to standard activities with (pre)determined durations whereas grey nodes correspond to repair activities involving a defective component. The number within the node refers to the component type. So we have four defective components of type 1 and three defective ones of type 2, for example. Furthermore there is a number of available intact spare components for each type (two of type 1 and one of type 2). Replacing a defect component by an intact one is assumed to be done within negligible time and leads to an activity of duration 0 at that position; otherwise the defect component must be repaired which results in an activity of a component-specific duration (3 for type 1 and 1 for type 2).
Instead of dealing with intact components (which shorten durations) it is in the following more convenient to occupy a complementary (but equivalent) view and to deal with activities of additional time which must be inserted:

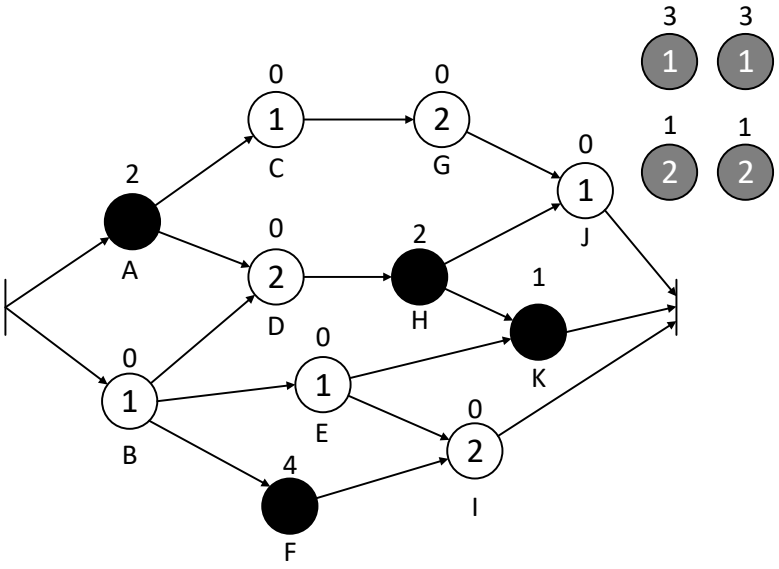


Figure 2: Equivalent project network with repair activities to be inserted

That means for component type 2 that we model the three defective components of which one is to be replaced, as three activities of duration zero of which two need to be replaced by activities with component-specific duration. In both cases we have finally one activity of duration 0 and two of duration p_2^{tp} (analogously for type 1 where two intact resp. two defect components must be placed). Thus, instead of asking which defective components have to be replaced by intact ones the question is which activities have to be performed as repair activities - both are equivalent. Accordingly, we formally assume that the activities initially have duration 0 and a certain number of them must be replaced (we will say "inserted") by non-zero activities.
The objective is to find an insertion which minimizes makespan, i.e. project duration. In the example above a makespan of 6 can be achieved by inserting the type-1 activities of duration 3 at C and E, and by inserting the type-2 activities at D and I. This is also optimal: Indeed, inserting a type-1 activity at B or J would already lead to paths of length at least 7 (BFI resp. $ADHJ$) such that only C and E remain. Since further D or G is inserted, a path of length 6 cannot be avoided.

The following problem definition formalizes the general situation:

An instance $\mathcal{A}$ of the Project Activity Insertion Problem (PAIP) is given by the following data:

- a set of *activities* $A = \{0, \ldots, n+1\}$ where 0 will act as start activity and $n+1$ as end activity
- a binary *precedence* relation $P \subseteq A \times A$ such that (A, P) is an acyclic digraph and 0 (resp. $n+1$) is direct or indirect precedessor (resp. successor) of all other nodes
- a set $\mathcal{T} = \{1, \ldots, \bar{\tau}\}$ of *types*
- a partition $A = \bigcup_{\tau=0}^{\bar{\tau}} A_\tau$ (i.e. with $A_i \cap A_j = \emptyset$ for $i \neq j$) where
  A_0 is the set of *fixed* activities with $0, n+1 \in A_0$,
  $A \backslash A_0 = \bigcup_{\tau=1}^{\bar{\tau}} A_\tau$ is the set of *flexible* activities, with
  A_τ being the non-empty set of (flexible) *activities of type* τ for $\tau = 1, \ldots, \bar{\tau}$
  The corresponding type function is defined as $tp(i) = \tau$ for $i \in A_\tau$.

  We will also refer to activities as "positions" or "nodes" corresponding to their representation in a project network.

- durations of fixed activities $d_i^{fx} \in \mathbb{N}_0$ for $i \in A_0$ with $d_0^{fx} = d_{n+1}^{fx} = 0$
- durations of flexible types $d_\tau^{tp} \in \mathbb{N}$ for $\tau = 1, \ldots, \bar{\tau}$
  (we will usually avoid the superscript if it is clear if the subscript refers to a fixed activity or a type)
- number of activities of type τ which have to be inserted r_τ, for $\tau = 1, \ldots, \bar{\tau}$

Since the numbers $\bar{r}$ of activities to be inserted for each type are often treated as separate parameter we will frequently use the notation $\mathcal{A}_{\bar{r}} = (\mathcal{A}, \bar{r})$ to make their value explicit.
An *insertion* of $\mathcal{A}$ is a family $\mathcal{I} = (I_\tau)_{\tau \in \mathcal{T}}$ consisting of subsets $I_\tau \subseteq A_\tau$ with $|I_\tau| = r_\tau$ for all $\tau = 1, \ldots, \bar{\tau}$. An insertion $\mathcal{I}$ of $\mathcal{A}$ leads to the corresponding *inserted PAIP-instance* $\mathcal{A}_\mathcal{I}$ which is a common CPM project network $\mathcal{A}_\mathcal{I} = (A, P, (d_i^\mathcal{I})_{i \in A})$ with processing times

$$d_i^\mathcal{I} := \begin{cases} d_i^{fx} & , i \in A_0 \\ d_\tau^{tp} & , i \in I_\tau \\ 0 & , else \end{cases} \qquad \text{for all } i \in A$$

That means durations of fixed activities remain the same, inserted activities get the type-specific durations and the remaining (flexible and non-inserted) ones get duration 0. Along with a project network, an insertion also induces the associated earliest start schedule (in the following denoted by $\mathfrak{s}(\mathcal{A}_\mathcal{I})$) and its makespan (short $MS(\mathcal{A}_\mathcal{I})$), such that often we will simply speak of the schedule or the makespan resulting from a given insertion or which an insertion yields. Thus, as soon as an insertion is given, schedule and makespan are determined.
Now the PAIP can be stated as follows: Given a PAIP-instance $\mathcal{A}$, find an insertion $\mathcal{I}^*$ which minimizes the makespan of the inserted PAIP-instance, i.e. $MS(\mathcal{A}_{\mathcal{I}^*}) = min_\mathcal{I}\, MS(\mathcal{A}_\mathcal{I})$. The corresponding decision version additionally specifies a bound K for the makespan and asks if there is an insertion leading to a makespan at most K.

3 Complexity analysis

Determining the computational complexity of a combinatorial optimization problem is an important step towards a better understanding of the hardness of the problem. It admits to identify relatively easy tractable cases as opposed to difficult ones. It also conveys first insights as to which parameters hardness of instances may be driven. We assume that the reader has some familiarity with basic notions of computational complexity like $\mathcal{O}$-notation, polynomial time algorithm, (pseudo-)polynomial reduction, NP-completeness (in the strong sense); for corresponding literature see [Garey and Johnson 1979, Papadimitriou 1994, Arora and Barak 2009].
For complexity analysis we will consider the decision version of the PAIP, which is easily seen to be contained in NP: An insertion $\mathcal{I}$ simply consists of disjoint subsets of A from which $\mathcal{A}_\mathcal{I}$ can be constructed in linear time. Then $MS(\mathcal{A}_\mathcal{I})$ can be computed in linear time as well by forward pass computation. Hence, given $\mathcal{A}$ and an upper bound K for the makespan, it can be verified in linear time if there is an insertion leading to a makespan of at most K.
For a more fine-grained complexity analysis of the PAIP we regard restrictions to classes defined via upper bounds on the number of types, of positions for each type and of positions to be inserted. Accordingly, each triple (T, S, R) defines a class containing the PAIP-instances which satisfy the bounds for those parameters given by that triple i.e. instances $\mathcal{A}$ with $t \leq T$, $|A_\tau| \leq S$, $r_\tau \leq R$ for all $\tau = 1, \ldots, \bar{\tau}$. An entry "-" at a position means that no bound is applied on the corresponding parameter. In the following we will denote such a class simply by the corresponding triple.
Since $|A_\tau| < r_\tau$ for some τ will give an infeasible instance - it is not possible to insert in more positions than available - we need only regard classes with $S \geq R$. In fact we can further restrict to classes with

$S > R$ since for a type τ with $|A_\tau| = r_\tau$ all its positions will be inserted such that they can treated as fixed activities of duration p_τ. Thus we need only regard cases (T, S, R) with $T \geq 1$ and $S > R \geq 1$ (with unbounded "-" treated greater than any finite value). Note that $(T, S, R) \subseteq (T', S', R')$ if $T \leq T'$, $S \leq S'$ and $R \leq R'$.

In the following we will prove the complexity of the problem classes as distinguished by this scheme. The following figure provides an overview of the main complexity results:

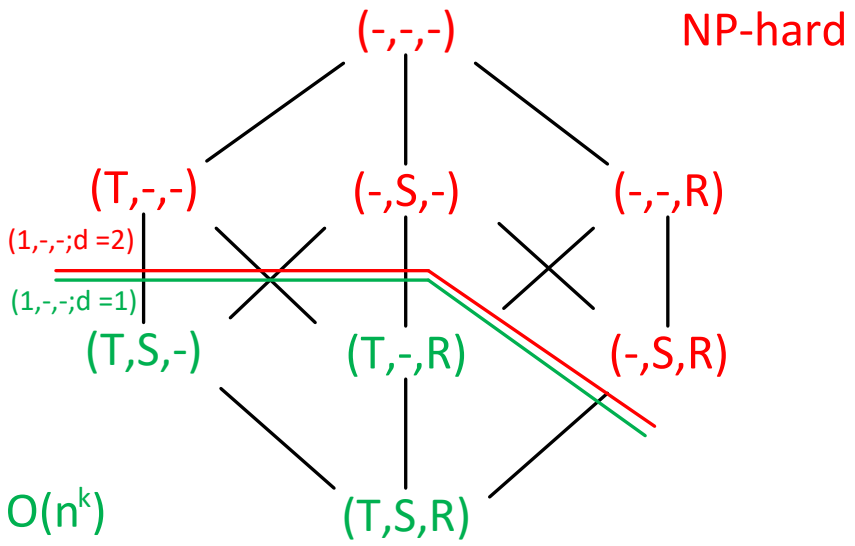


Figure 3: Complexity results

This figure contains all possible combinations according to the scheme considered; however note that cases $(-, S, -)$ resp. $(T, S, -)$ actually correspond to $(-, S, R)$ resp. (T, S, R) since a bound S on the number of positions implies a bound R on the number of activities to be inserted.

3.1 (T,S,R) and (T,-,R)

These cases are solvable by rather straightforward polynomial algorithms:

Proposition 1. *(T,-,R) can be solved in $\mathcal{O}(n^{TR+1})$ for all $T, R \geq 1$.*

Proof. Given an instance $\mathcal{A}$ in $(T, -, R)$, there are not more than $\prod_{\tau=1}^{\bar{\tau}} \binom{|A_\tau|}{r_\tau} \leq \prod_{\tau=1}^{\bar{\tau}} |A_\tau|^{r_\tau} \leq n^{TR}$ possible insertions. Since the makespan for each inserted PAIP-instance can be computed in linear time, we get a running time $\mathcal{O}(n^{TR+1})$ to determine the optimal makespan. □

Proposition 2. *(T,S,R) can be solved in linear time.*

Proof. For an instance $\mathcal{A}$ in (T, S, R) there are not more than $\prod_{\tau=1}^{\tau=t} |A_\tau|^{r_\tau} \leq \prod_{\tau=1}^{t} S^R \leq S^{TR}$ and hence a bounded number of possible insertions. Thus the optimal makespan can be determined in linear time. □

However, dropping also the bound on the number of types T yields a hard case as seen next.

3.2 (-,-,R)

This case can be shown to be NP-complete in the strong sense by reduction from 3-Partition:

Proposition 3. *(-,-,1) is strongly NP-complete.*

Proof. By (pseudo-polynomial) reduction from 3-PARTITION, which asks, given $3n$ (different) integers the sum of which is nB, if there is a partition into triple sets such that the sum of each triple is equal (namely to B); i.e. given integers $a_1, \ldots, a_{3n}$ and B with $\sum_{i=1}^{3n} a_i = nB$, if there are sets $T_i \subseteq \{a_1, \ldots a_{3n}\}$ with $\bigcup_{i=1}^{n} T_i = \{a_1, \ldots, a_{3n}\}$, $|T_i| = 3$ and $T_i \cap T_j = \emptyset$ which satisfy $\sum_{j \in T_i} a_j = B$ for each $1 \leq j \leq n$. This problem remains strongly NP-complete if one restricts to the case where $B/4 < a_i < B/2$ for each i [Garey and Johnson 1979, p. 224].
So let $(a_i)_{1 \leq i \leq 3n}$ be such an instance of 3-PARTITION. We construct a PAIP-instance $\mathcal{A}$ with $\bar{\tau} = 3n$ types as follows: At first we a define a path $v_1 \ldots v_{3n}$ with v_i being a node of type i. We take n copies of this path and link the v_1's to a common start node and the v_n's to an end node. The type-specific durations are set to $d_\tau := a_\tau$, the number of positions to be inserted is set to $r_\tau = 1$ for all $\tau = 1, \ldots, 3n$. Note that the numerical values of $\mathcal{A}$ are polynomially bounded in those of $(a_i)_{1 \leq i \leq 3n}$, since they are simply taken over. Now we have to show that $\mathcal{A}$ has an insertion leading to a makespan $MS \leq B$ iff there is a 3-partition of $a_1, \ldots a_{3n}$.

'$\Rightarrow$' Assume that there is an insertion $\mathcal{I}$ of $\mathcal{A}$ leading to an inserted network with makespan $MS \leq B$. Then each (inserted) chain has a length at most B whence at most three positions are inserted due to $B/4 < a_i$ for each i. On the other hand, since in the whole network precisely one position for each type is inserted, the sum of durations of the inserted network is $\sum_{i=1}^{3n} a_i = nB$ whence each of the n chains has a length B, thus also containing exactly three inserted positions due to $a_i < B/2$ for each i. Altogether the inserted positions of each chain correspond to a triple of a 3-partition of given instance.
'$\Leftarrow$' Conversely assume that there is a 3-partition $(T_j)_{1 \leq j \leq n}$ of $a_1, \ldots, a_{3n}$. Define from this an insertion of $\mathcal{A}$ as follows: For each triple $T_j = \{a_{j_1}, a_{j_2}, a_{j_3}\}$ insert in the jth chain the positions $v_{j_1}, v_{j_2}, v_{j_3}$ (of types j_1, j_2, j_3 , respectively). Then each chain has length B (equal to the sum of each triple) and each type i is inserted only once in the whole network (since each a_i occur in only one triple). Thus we have a feasible insertion leading to an inserted network with makespan B. □

Since each *(-,-,1)*-instance is contained in *(-,-,R)* for each $R \geq 1$, it follows

Corollary 1. *(-,-,R) is strongly NP-complete for all $R \geq 1$.*

and hence also for *(-,-,-)*, the unrestricted PAIP:

Corollary 2. *(-,-,-) , i.e. the unrestricted PAIP, is strongly NP-complete.*

3.3 (T,-,-)

This case is NP-complete in the strong sense as well, already for rather restricted conditions, especially for $T = 1$ (and hence also for each $T \geq 1$). However, to establish this result requires some more work than for the last case. Let $MS^+(\mathcal{A})$ denote the upper bound obtained by inserting all $|A_1|$ flexible nodes of a PAIP network $\mathcal{A}$ (independent of r_1). Then we have:

Proposition 4. *(1,-,-) is strongly NP-complete (even for instances $\mathcal{A}$ with $p_1 = 2$ and $MS^+(\mathcal{A}) \leq 5$).*

Proof. By reduction from Independent Set, i.e. the problem to decide for a given graph and bound K if it has set of K pairwise non-adjacent nodes. This problem is strongly NP-complete since it is NP-complete and does not contain any numerical values [Garey and Johnson 1979, p. 194].

At first we describe the construction of a PAIP-instance which will be subsequently used in the reduction. The instances will be constructed in two steps:
The PAIP-networks used in following reduction are built from smaller gadgets which are subsequently linked according to an underlying graph. For given $n \in \mathbb{N}$, each gadget Gd_n has the following structure (edges directed upwards):

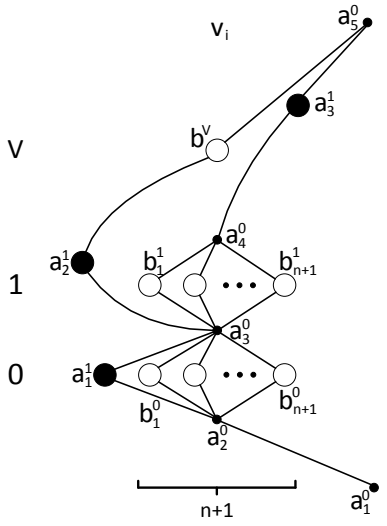


Figure 4: Gadget for the case (1,-,-)

The (black) fixed nodes have duration either 0 (a_i^0, small) or 1 (a_i^1, large), the (white) flexible nodes duration 2 if inserted (and 0 if not). There are three groups (we will refer to them as "levels") of flexible nodes: The upper 'V'-level contains one node (b^V), the middle '1'-level (b_i^1) and the lower '0'-level (b_i^0) each contain $n+1$ nodes.
Now let an arbitrary (undirected) graph $G = (V, E)$ on n nodes be given. For each node v_i we take a copy $Gd_n(v_i)$ of the gadget above and identify their start resp. end nodes. Furthermore the gadgets are linked corresponding to the nodes in G as follows: For $v_i, v_j \in V$ with $\{v_i, v_j\} \in E$ we draw an arc from a_4^0 in $Gd_n(v_i)$ to b^V in $Gd_n(v_j)$ (since G is undirected there will also be a link with reversed i, j).

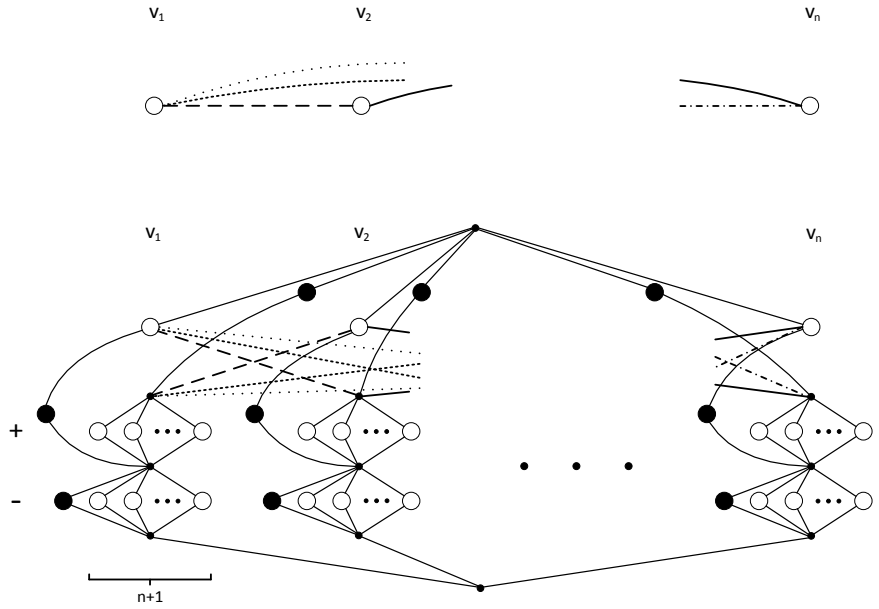


Figure 5: Different edges in G are dashed differently, corresponding edges occur in $\mathcal{A}^G$ with the same dashing pattern. Edges in $\mathcal{A}^G$ are directed upwards.

Given finally a value $r_1 := n^2 + n + k$ (for some $k \in \mathbb{N}$) as the number of nodes to be inserted we get a completely specified PAIP-instance $\mathcal{A}^G_{n^2+n+k} = (\mathcal{A}^G, n^2 + n + k)$. Note that $\mathcal{A}^G_{n^2+n+k}$ can be constructed from G in polynomial time and that its numerical values are polynomially bounded in n as well; since for an instance of Independent Set only a bound $\leq n$ is specified, this already suffices for a pseudopolynomial transformation to get strong NP-completeness afterwards.
Now it remains to show the following

Lemma 1. *G has an independent set of size k iff $\mathcal{A}^G_{n^2+n+k}$ has an insertion with makespan equal to 4.*

Proof. For both directions we have to regard different paths from start to end in $\mathcal{A}^G$. Each such path is of one of the following six types:

$P1:\ a_1^0 - a_2^0 - a_1^1 - a_3^0 - a_2^1 - b^V - a_5^0$
$P2:\ a_1^0 - a_2^0 - a_1^1 - a_3^0 - b_i^1 - a_4^0 - a_3^1 - a_5^0$
$P3:\ a_1^0 - a_2^0 - a_1^1 - a_3^0 - b_i^1 - a_4^0 - b^V(\textit{in other gadget}) - a_5^0$
$P4:\ a_1^0 - a_2^0 - b_i^0 - a_3^0 - a_2^1 - b^V - a_5^0$
$P5:\ a_1^0 - a_2^0 - b_i^0 - a_3^0 - b_j^1 - a_4^0 - a_3^1 - a_5^0$
$P6:\ a_1^0 - a_2^0 - b_i^0 - a_3^0 - b_j^1 - a_4^0 - b^V(\textit{other gadget}) - a_5^0$

Now we can turn to prove the directions:
'$\Rightarrow$' Let $U \subseteq V$ be an independent set of G of size k. It defines a canonical insertion I_U of $\mathcal{A}^G_{n^2+n+k}$ as follows: For $v \in U$ the V- and 1-level of $Gd_n(v)$ are inserted, i.e. $\{b^V, b_1^1, \ldots, b_{n+1}^1\} \subseteq I_U$ for that gadget. For $v \in V \backslash U$ the 0-level of $Gd_n(v)$ is inserted, i.e. $\{b_1^0, \ldots b_{n+1}^0\} \subseteq I_U$ for that gadget. All other flexible nodes remain uninserted, thus there are $k(n+2) + (n-k)(n+1) = n^2 + n + k$ inserted nodes whence I_U is an insertion of $\mathcal{A}^G_{n^2+n+k}$.
Now we check that the makespan of the inserted network does not exceed 4: Let be P an arbitrary path from start to end. Depending on the type of P we get the following lengths of P:

$P1$: $l(P) = 4$ if b^V is inserted, 2 otherwise.
$P2$: $l(P) = 4$ if b_i^1's are inserted, 2 otherwise.

$P3$: Let a_4^0 be contained in the gadget of v_i and b^V in that of v_j. Thus (a_4^0, b^V) is an arc from the gadget of v_i to the gadget of v_j whence $[v_i, v_j]$ is an edge of G. Therefore v_i or v_j is not contained in the independent set U. Hence the 1-level of V_i (including b_i^1) or b^V is not inserted according to definition of I_U. Thus only at most one flexible node along P is inserted whence $l(P) \leq 3$.
$P4$: Since V-level and 0-level are not inserted for the same gadget either b_i^0 or b^V is not inserted. Thus $l(P) = 3$.
$P5$: Either 1-level or 0-level is not inserted for the same gadget whence $l(P) = 3$.
$P6$: Either b_i^0 or b_j^1 is not inserted whence $l(P) \leq 4$.
Thus the length of each path is at most 4 whence the makespan of the inserted PAIP-instance is also at most 4.

'$\Leftarrow$' Let G be a graph and k such that $\mathcal{A}^G_{n^2+n+k}$ has an insertion I with makespan at most 4. At first we show that for each gadget either the 0- or the 1-level is (at least partially) inserted. A makespan of at most 4 enforces that for each gadget $Gd_n(v_i)$ the 0-level or the 1-level is not inserted, otherwise there would be inserted b_i^0 and b_j^1 whence the path of type $P5$ containing them would have length 5. As a consequence one gets that for each gadget there are altogether at most $n+1$ inserted nodes in the 0- and 1-level.
Furthermore for each gadget either the 1-level or the 0-level is (at least partially) inserted: Otherwise let $Gd_n(v_i)$ do not contain any inserted nodes in the 0- and 1-level. But then there are only at most $(n-1)(n+1)+n = n^2+n-1$ inserted nodes instead of n^2+n+k as assumed.
Let $U := \{v \in V | b^V \text{ in } Gd_n(v) \text{ is inserted}\}$. We want to show that U is an independent set of size at least k in G. It holds $|U| \geq k$ since all 0- and 1-levels altogether contain at most $n(n+1) = n^2+n$ nodes, leaving at least k nodes in V-level. Now let $v, v' \in V$, $v \in U$, $\{v, v'\} \in E$. It suffices to show $v' \notin U$. Since $v \in U$ the V-level of the v-gadget is inserted whence its 0-level is not inserted otherwise we would have a $P4$-path of length 5. Thus the 1-level of the v-gadget is inserted. Using that there is an arc from a_4^0 in $Gd_n(v)$ to b^V in $Gd_n(v')$ (since $\{v, v'\} \in E$) we get a non-inserted V-level of the v'-gadget (otherwise there would be a $P3$-path of length 5) whence $v' \notin U$. Altogether we get an independent set of size $\geq k$ and hence of size k.
This proves the lemma and concludes the whole proof. □

For a somewhat higher-level view on the mechanics of the proof, note that the inserted nodes of the V-level correspond to an independent set of the underlying graph. To guarantee independence, one has to make sure that insertion of the V-level corresponding to one node forces non-insertion of the V-level of a node which is adjacent to the first one. This is done by, figuratively spoken, transmitting the "pressure" of insertion via several intermediate and alternating steps which can be sketched as follows:

insertion V-level $\rightarrow$ non-insertion 0-level $\rightarrow$ insertion 1-level $\rightarrow$ non-insertion adjacent V-level
$MS \leq 4$ $r_1 \geq n^2+n$ $MS \leq 4$

A similar reduction based on a slightly modified construction can be used to establish the next result:

Proposition 5. *(-,2,1) is strongly NP-complete (even for instances $\mathcal{A}$ with $d_\tau = 2$ for all $\tau \in \mathcal{T}$)*

Proof. The PAIP-instances used in this proof are built from gadgets looking as follows:

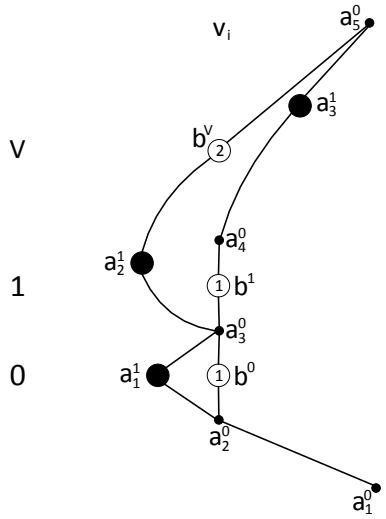


Figure 6: Gadget for proof of (-,2,1)

The only differences to the gadgets in the proof above are that
- 0- and 1-level each consists of only one node
- the V-level node is of different type than the nodes of 0- and 1-level (which have the same type)
The durations are as above, i.e. (black) fixed nodes with 0 (a_i^0, small) resp. 1 (a_i^1, large), (white) flexible nodes with 2 if inserted (and 0 if not).
Given a graph G on n nodes, for each node v_i a gadget $Gd(v_i)$ with two new types $tp(b^0) = tp(b^1) = i$ and $tp(b^V) = n + i$ is introduced. In the same way as before these gadgets are linked according to the edges of G. Further we add a fixed activity with duration $2(n-k)-4$ preceding a_1^0 and finally a path $c_{n+1} \dots c_{2n}$ consisting of n nodes of types $n+1, \dots, 2n$ parallel to the remaining network:

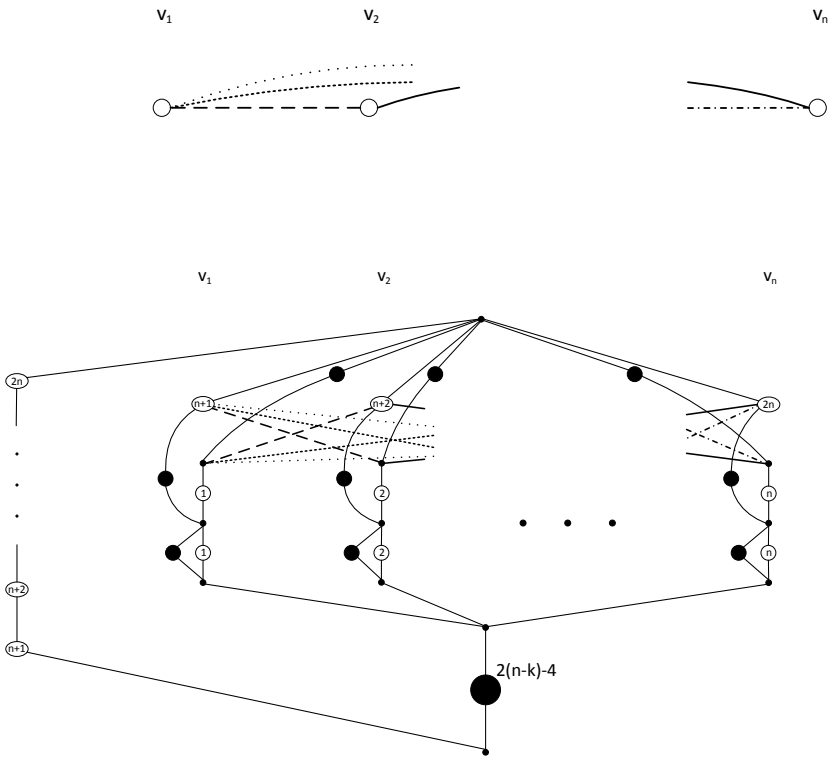


Figure 7: Different edges in G are dashed differently, corresponding edges in $\mathcal{A}^G$ are dashed with the same pattern. Edges in $\mathcal{A}^G$ are directed upwards.

Thus the network has $2n$ types each with two positions. Setting $r_\tau = 1$ for all $\tau = 1, \ldots, 2n$ we get a completely specified PAIP-instance $\mathcal{A}_1^G$. As before, it can be constructed in polynomial time and has polynomially bounded numerical values. Hence it remains to show:

Lemma 2. *G has an independent set of size k iff $\mathcal{A}_1^G$ has an insertion with makespan at most $2(n-k)$.*

Proof. '$\Rightarrow$': Let be $U \subseteq V$ an independent set of G of size k. Then we define an insertion $\mathcal{I}_U = (I_\tau)_{1 \leq \tau \leq 2n}$ of $\mathcal{A}_1^G$ as follows: If $v_i \in U$ then insert the V- and 1-level of $Gd(v_i)$, i.e. $I_i = \{b^1(v_i)\}$ and $I_{n+i} = \{b^V(v_i)\}$. If $v_i \in V \backslash U$ then insert the 0-level of $Gd(v_i)$ and the node of type $n+i$ in the chain, i.e. $I_i = \{b^0(v_i)\}$ and $I_{n+i} = \{c_{n+i}\}$
Analogously to the proof above we get a makespan of 4 for the sub-network between a_1^0 and a_5^0, i.e. without the fixed node of length $2(n-k)-4$ and the chain. Indeed, with that node we have a makespan of $2(n-k)$ which is equal to the length of the chain which contains $n-k$ inserted nodes. Thus the inserted instance has a makespan of $2(n-k)$.
'$\Leftarrow$': Let be $\mathcal{I} = (I_\tau)_{1 \leq \tau \leq 2n}$ an insertion with makespan $2(n-k)$. It follows that the sub-network between a_1^0 and a_5^0 has a makespan of at most 4 and the chain contains at most $n-k$ inserted nodes whence at least k nodes in the V-level are inserted. As $\mathcal{I}$ is an insertion of $\mathcal{A}_1^G$, precisely one node of each type $\tau = 1, \ldots, n$ is inserted whence for each gadget $Gd(v_i)$ either the 0- or the 1-level is inserted (as in the proof above, but there for a more complicated reason). In the same way as above we get that $U := \{v \in V | b^V \text{ in } Gd(v) \text{ inserted}\}$ is an independent set of G: Indeed, given $v, v' \in V$ with $\{v, v'\} \in E$ and $v \in U$. Then b^V in the gadget of v is inserted whence the 1-level in the gadget of v' is not inserted (otherwise the path between a_1^0 and a_5^0 through a_1^1, b^1 (both in gadget of v')and b^V (gadget of v) would have length 5), whence the 0-level in the gadget of v' is inserted, whence the

V-level in the gadget of v' is not inserted, whence $v' \notin U$. Thus U is an independent set of size $\geq k$.
□

3.4 $(1, -, -)$ with unit duration of flexible nodes

In the construction for establishing strong NP-completeness of $(1, -, -)$ the inserted activities got a duration of 2, i.e. $d_1^{tp} = 2$ for type 1. With respect to d_1^{tp} this is the lowest value for which the problem is hard since restricted to $d_1^{tp} = 1$ this case becomes solvable in polynomial time as will be shown in the following. For this we will apply so-called Sperner theory, more precisely the concept of (Sperner) $k-$families for which there exists a minimax characterization and an efficient algorithm. As preparation we recapitulate some essential notions and results from this area (for a comprehensive treatment see for example [Engel 1997]):
Let $(P, <)$ be a (finite) partially ordered set (short: poset), i.e. a binary relation which is irreflexive, asymmetric and transitive. A *chain* is a set of mutually comparable (and hence linearly ordered) elements, an antichain is a set of mutually non-comparable elements; a *k-chain* is a chain of k elements. A *k-family* is a subset of P which does not contain any $(k+1)$-chain. A weighted poset $(P, <, w)$ is additionally enhanced by a weight function $w : P \to \mathbb{N}_0$. For a set $S \subseteq P$ define the weight of S as $w(S) := \sum_{s \in S} w(s)$. Concerning the complexity to determine a maximal weighted family the following result is known ([Berenguer et al. 1982]):

Theorem 1. *Let $(P, <, w)$ be a weighted poset on n elements and $k \in \mathbb{N}$. Then a k-family of maximal weight can be computed in time $\mathcal{O}((kn)^3)$.*

As noted by Berenguer et al. the problem remains polynomial in n since the case $k > n$ need not to be considered as all chains have a length of at most n, such that P is the k-family of maximal weight for $k \geq n$. - Now we turn to our original setting again and show:

Proposition 6. $(1, -, -|d_1^{tp} = 1)$ *can be solved in polynomial time.*

Proof. The proof consists of three subsequent reductions: At first we reduce the problem to the case where each fixed activity has duration $\mathcal{O}(n^2)$. This allows the fixed activities to be replaced by chains which consist of unit time activities and have polynomial length. Finally, by assigning prohibitive weights to these fixed unit time activities, this case can be reformulated as problem finding a weighted k-family in a partially ordered set, a problem which is polynomially solvable according to the Theorem above.
For simplicity we assume that the PAIP instances do not contain fixed nodes of duration 0. This is no real restriction since we get an equivalent instance by changing start and end nodes to unit time activities (which increases makespan by a constant of 2) and by successively replacing further fixed nodes v of duration 0 by connecting each predeccessor u and each successor w of v by an arc (u, w). Obviously the number of nodes is not increased by this procedure, the number of edges at most quadratically, such that this transformation can be done in polynomial time.
So let $(\mathcal{A}, r_1)$ be a PAIP instance from $(1, -, -|d_1^{tp} = 1)$ with r_1 nodes to be inserted.

1. The first reduction aims to reduce the given problem to one where fixed activities have duration $\mathcal{O}(n^2)$, more precisely durations $\leq 2n^2$. If this is already the case, this first step is superfluous and we can continue immediately with the second step. So assume here that there are fixed activities with duration $> 2n^2$.
The idea is based on the observation that an inserted activity delays subsequent start and end points by at most 1, such that an arbitrary insertion delays any start or end point by not more than n. Thus the activity interval of long activities (with duration $\gg n$) may be affected at their start and their end but not in between: There is a large part where such an activity will be always active, independent of the actual insertion. To specify those parts which may be affected by the movements of start and

end points due to different insertions compared with those which are not, we define primary intervals containing those time points being at most n after some start or end point and secondary intervals as the remaining ones, i.e. those time ranges where activity intervals are not affected by movements of start and end point due to insertions. In other words, projected to primary intervals a schedule depends on the insertion whereas projected to secondary intervals each schedule looks the same whatever the insertion was. As a consequence, the content of these long secondary intervals is predictable such that the actual problem can be reduced to an instance with shorter durations by cutting these secondary intervals out. So this is the idea behind the more formal steps in the following.
Towards this end, using a forward pass, we compute earliest start and finish times ES_i^0/EF_i^0 for each $i = 0, \ldots, n+1$ without any insertion, i.e. where each flexible node gets duration 0 ("0-insertion"). Correspondingly, let ES_i^1/EF_i^1 be the earliest start and finish times if each flexible activity is inserted, i.e. gets duration 1 ("1-insertion"). Then the earliest start times for any insertion I satisfy $ES_i^0 \leq ES_i^I \leq ES_i^1 \leq ES_i^0 + n$ (and correspondingly for earliest finish times EF); the last inequality holds since the insertion of at most n unit time activities leads to a delay of at most n for each activity, as can be shown by a straightforward induction. Further it follows for each flexible i from $ES_i^0 = EF_i^0$ that $EF_i^I \leq EF_i^1 \leq EF_i^0 + n = ES_i^0 + n$.
This delay of at most n motivates the definition of the following intervals: Set $T := EF_{n+1}^0 + n$ as time horizon for the following considerations. The union of intervals $\bigcup_{1 \leq i \leq n}([ES_i^0, ES_i^0+n] \cup [EF_i^0, EF_i^0+n])$ forms $k \leq 2n$ disjoint closed intervals $PI_j = [a_j, b_j]$, $1 \leq j \leq k$ in $[0, T]$ (with $b_j < a_{j+1}$ and hence $a_1 = 0, b_k = T$) which we will call "primary intervals". Due to overlapping of the original intervals of length n, some of these primary intervals are possibly longer than n; however, following from our initial assumption, there is more than only one primary interval: Indeed, if $k = 1$ we would have $T \leq 2n^2$ and hence each fixed activity would also be of duration at most $2n^2$, which we explicitly excluded for this step. Thus we have $k > 1$. Note that the primary intervals are defined using only the generic 0-insertion and hence do not depend on any other special insertion, i.e. they are already determined by the PAIP instance alone.
From the definition of primary intervals it follows that if $ES_i^0 \in PI_j$ for some i, j then also $ES_i^0 + n \in PI_j$ (and the same for EF). From this we get that if $ES_i^0 \in P_j$ then also $ES_i^I \in P_j$ for each insertion I (and the same for EF). In other words, the earliest start (resp. finish) time of an activity remains in the same primary interval, independent of the insertion since the maximal delay of at most n due to an insertion does not suffice to leave the primary interval. Further, since $EF_i^I \leq ES_i^0 + n$ for flexible i, a flexible activity is active within the same primary interval also for each insertion and hence does not leave it, whereas for a fixed activity start and end time may be contained in different primary intervals. The following figure illustrates the position of primary intervals together with a schedule:

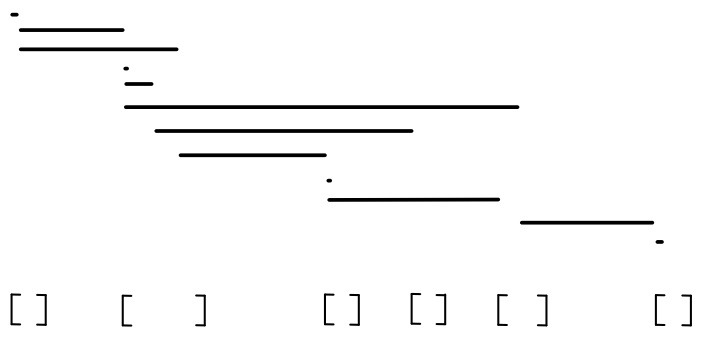


Figure 8: Primary intervals

Here bars indicate activity intervals corresponding to a Gantt chart. Each start and end point induces an interval of length n; due to overlapping some intervals may be longer, as indicated by the brackets

at the bottom of the figure.
The complement of the primary intervals in $[0,T]$ forms $k-1 \leq 2n-1$ disjoint (open) intervals $SI_j = (b_j, a_{j+1})$, $j = 1,\ldots,k-1$; we will refer to them as "secondary intervals". Clearly, like the primary intervals, they do not depend on any special insertion. As the start (resp. end) time of each activity remains in the same primary interval for all insertions and furthermore each flexible activity starts and ends in the same primary interval for all insertions, those secondary intervals have the following key property: For all insertions, the set of activities active in a given secondary interval is the same; it only consists of fixed activities which are active throughout the whole secondary interval. Thus, if restricted to secondary intervals, the earliest start schedules look the same for all insertions, for example like this:

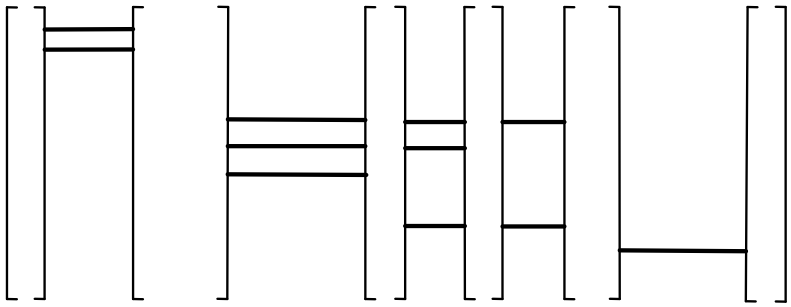


Figure 9: Primary intervals and activity intervals within secondary intervals

Here primary intervals are indicated by brackets (and the details of the schedule therein, which *do* depend on the insertion, are blanked out); the bars between them, i.e. throughout the secondary intervals, correspond to running fixed activities, some of which are active in several subsequent secondary intervals (as indicated by bars continuing at the same level).
Since the part of the schedule contained in the secondary intervals is not affected by the insertions, the idea is to cut them out and to generate a compact instance description with shorter task durations such that a polynomial translation is possible. The shorter task durations can be determined by left-shifting the parts of the schedule in all primary intervals after all secondary intervals have been removed.
Formally, if $ES_i^I \in [a_j, b_j]$ is the earliest start point and its primary interval for any insertion $I \subseteq A_1$, the new start point would be $\tilde{ES}_i^I := ES_i^I - \sum_{j'=1}^{j-1}(a_{j'+1} - b_{j'})$ (namely reduced by the lengths of preceding secondary intervals). The resulting schedule is obviously not feasible for the original instance since some fixed activities have shorter durations now. However, instead it is a feasible earliest start schedule for the same insertion I applied to a modified instance which one gets by shortening the durations of fixed activities by just the activity intervals which have been contracted and in which the corresponding activity was active: To specify that instance, let Act_j be the set of fixed activities which are active throughout $SI_j = (b_j, a_{j+1})$ (independent of the insertion). Then define new, shortened durations for each fixed activity i by $d'(i) := d(i) - \sum_{j=1,\ldots,k-1,\, i \in Act_j}(a_{j+1} - b_j)$, i.e. reduce its original duration by the length of the secondary intervals where it is active. Changing the durations in that way yields a new PAIP-instance $\mathcal{A}'$ where the duration of each fixed activity is at most the length of all primary intervals and hence not more than $2n^2$. Note further the following properties of that instance:
- Since the set of flexible activities remains unchanged we have $A_1 = A_1'$ whence an insertion of $\mathcal{A}$ is also an insertion of $\mathcal{A}'$ and vice versa; i.e. $\mathcal{A}$ and $\mathcal{A}'$ have the same set of (r_1-)insertions.
- The new instance does not contain fixed activities of duration 0: Given a fixed activity i, recall that its start (resp. end) point remains in the same primary interval for all insertions. Regarding the

0-insertion, recall that ES_i^0 and $ES_i^0 + n$ are contained in the same primary interval.
- Finally note that the new instance can be constructed from $\mathcal{A}$ in polynomial time; for this recall the steps: determine 0-insertion, primary intervals, secondary intervals, set Act_j for each secondary interval, reduce durations of fixed activities according to these sets.

It should be (intuitively) clear by construction that, given any insertion I, the schedule resulting from applying I to the original instance $\mathcal{A}$ and then contracting the secondary intervals agrees with the schedule resulting from applying the same I to the shortened instance $\mathcal{A}'$. We show this in somewhat more detail:
Formally the schedule resulting from contraction can be defined by reducing all start and end points by the total length of preceding secondary intervals, i.e. if $ES_i^I \in [a_j, b_j]$ is a start point contained in its primary interval, then the new start point after contraction is given by $ES_i^{I,c} := ES_i^I - \sum_{j'=1}^{j-1}(a_{j'+1} - b_{j'})$ (and correspondingly for the end points $EF_i^{I,c}$). Note that for given i the sum in the second term is the same for all insertions I. We check that this schedule actually agrees with that resulting from inserting the same I in the shortened instance $\mathcal{A}'$ (and that it is a feasible schedule at all). For this it suffices to show that $\mathfrak{s}^c(\mathcal{A}, I)$ is an earliest start schedule the durations of which agree with those in $\mathfrak{s}(\mathcal{A}', I)$.
At first note that if $t_1 < t_2$ are time points in primary intervals and t_1^c, t_2^c their counterparts after contraction, then it holds $t_1^c < t_2^c$. If further t_1, t_2 are contained in the same primary interval then also the difference is retained, i.e. $t_2^c - t_1^c = t_2 - t_1$. From the former observation follows feasibility of $\mathfrak{s}^c(\mathcal{A}, I)$ w.r.t. precedence since $EF_h^I \leq ES_i^I$ yields $EF_h^{I,c} \leq ES_i^{I,c}$ for $h \in P_i$ (where P_i is the set of predecessors of i). Furthermore, since $ES_i^I = max_{h \in P_i} EF_h^I$ for each activity i, we have equality for at least one predecessor h^*, i.e. $ES_i^I = EF_{h^*}^I$. Thus also $ES_i^{I,c} = EF_{h^*}^{I,c}$ and hence $ES_i^{I,c} = max_{h \in P_i} EF_h^{I,c}$, whence $\mathfrak{s}^c(\mathcal{A}, I)$ is an earliest start schedule as well.
Regarding durations next, the duration of each fixed activity is reduced by the total length of secondary intervals between its start and end point whence it is equal to the duration in $\mathcal{A}'$ (note here again, that the reduction is independent of insertion/the same for all insertions I). On the other hand, the duration of flexible activities remains the same since each of them is completely contained in some primary interval. Thus a flexible activity has duration 1 in $\mathfrak{s}^c(\mathcal{A}, I)$ iff it has duration 1 in $\mathcal{A}_I$ and also iff in $\mathcal{A}'_I$ (and otherwise 0). Thus, since durations of activities agree in $\mathfrak{s}^c(\mathcal{A}, I)$ with those in $\mathfrak{s}(\mathcal{A}', I)$, and further the former schedule is an earliest start schedule as is the latter, altogether $\mathfrak{s}^c(\mathcal{A}, I)$ is identical with $\mathfrak{s}(\mathcal{A}', I)$.
From $\mathfrak{s}^c(\mathcal{A}, I) = \mathfrak{s}(\mathcal{A}', I)$ it follows that $MS(\mathcal{A}_I) = MS(\mathcal{A}'_I) + \sum_{j=1}^{k-1}(a_{j+1} - b_j)$ for each insertion $I \subseteq A_1 = A'_1$ (of size r_1). Thus the difference of makespan is a constant independent of the insertion whence an optimal insertion for $\mathcal{A}$ is also optimal for $\mathcal{A}'$ and vice versa. Especially an insertion I is optimal for $\mathcal{A}$ (i.e. yielding a minimal makespan among all r_1-insertions) iff it is so for $\mathcal{A}'$. Thus to determine the optimal insertion of $\mathcal{A}$, it suffices to determine the optimal insertion for the shortened instance $\mathcal{A}'$; and since $\mathcal{A}'$ can be constructed from $\mathcal{A}$ in polynomial time (compute primary intervals, determine lengths of secondary intervals, reduce durations), the original problem is polynomial time reducible to instances with fixed activities of duration $\mathcal{O}(n^2)$.
Altogether we get the assertion:

Claim: For any insertion $I \subseteq A_1 = A'_1$ it holds $MS(\mathcal{A}'_I) = MS(\mathcal{A}_I) - \sum_{1 \leq j \leq k-1}(a_{j+1} - b_j)$.

Thus an insertion is optimal for the original instance iff it is so for the shortened instance, such that it suffices to solve the problem for the shortened instance $\mathcal{A}'$.

2. Having now an instance $\mathcal{A}'$ with fixed activities of duration $\mathcal{O}(n^2)$, next each fixed activity is replaced by a chain of fixed unit time activities of corresponding length. More precisely, each fixed activity i and incident arcs are replaced by new nodes $\{i\} \times \{1, \ldots, d'_i\}$ and arcs forming a path $(i,1) \ldots (i, d'_i)$ and linking the predecessors of i to $(i,1)$ as well as (i, d'_i) to the successors of i. Each of

the new nodes $(i,1),\dots(i,d'_i)$ gets duration 1, thus being new fixed activities. Since these chains are of length $\mathcal{O}(n^2)$, the size of the resulting instance $\mathcal{A}''$ is polynomially bounded in the size of $\mathcal{A}'$.
Since each fixed activity in $\mathcal{A}'$ is simply replaced by a chain of fixed unit time activities of the same length whereas flexible activties remain the same, it follows that $\mathcal{A}'$ and $\mathcal{A}''$ have the same insertions, for which holds $MS(\mathcal{A}'_I) = MS(\mathcal{A}''_I)$ for each r_1-insertion $I \subseteq A'_1 = A''_1$. Thus it suffices to solve instance $\mathcal{A}''$.

3. At first note that each acyclic digraph induces a poset via the reachability relation, i.e. $u < v \Leftrightarrow u \neq v \wedge v$ (in)direct successor of u. More precisely, the binary relation $<$ is irreflexive, since $u < v$ implies $u \neq v$ by definition; it is asymmetric since if v is reachable from $u \neq v$, u is not reachable from v as the digraph is acyclic; it is transitive since if w is reachable from v and v from u then also w from u.
So let $(A'', <)$ be the poset induced by the acyclic digraph of the precedence relation of $\mathcal{A}''$, i.e. with $i < j$ if j is direct or indirect successor of i. Due to the same underlying set A'', it will be convenient in the following to identify the elements of both structures; correspondingly, an element in the partial order may be simply called fixed or flexible activity if if is contained in A''_0 or A''_1, respectively. Moreover the poset is extended by a weight function $w : A'' \to \mathbb{N}$ with $w(i) = 1$ if $i \in A''_1$ and $w(i) = n+1$ if $i \in A''_0$. The large weight of fixed activities will ensure that they are really contained in a maximum-weight k-family, since removing one fixed activity cannot be outweighed by choosing all flexible activities instead. Note that the resulting weighted poset $\mathcal{P}'' = (A'', <, w)$ can be constructed from $\mathcal{A}''$ in polynomial time. Insertions in $\mathcal{A}''$ and $k-$families in $\mathcal{P}''$ are related as follows:

Claim: $\mathcal{A}''$ has an r_1-insertion with makespan $\leq k$ iff $\mathcal{P}''$ has a k-family of weight $r_1 + (n+1)|A''_0|$.

'$\Rightarrow$': Let I be an r_1-insertion of $\mathcal{A}''$ with makespan $\leq k$. Then $F := I \cup A''_0$ precisely contains the unit time activities in $\mathcal{A}''_I$ (all other activities, i.e. the uninserted flexible activities, are of duration 0). Considered as subset in $\mathcal{P}$, F has weight $r_1 + (n+1)|A''_0|$ such that it suffices to show that it is also a k-family. So let be $C \subseteq F$ any chain in $\mathcal{P}$. Then the nodes of C as elements of $\mathcal{A}''_I$ are contained in a path $\bar{P}$ in $\mathcal{A}''_I$ with length $\leq k$ due to the makespan assumption. Thus C contains at most k elements since as elements of F they have duration 1, from which we get that F is a k-family.

'$\Leftarrow$': Let be F a k-family with weight $r_1 + (n+1)|A''_0|$. Then A''_0 is contained in F: Indeed, the weight of all elements in $\mathcal{P}$ is $|A''_1| + (n+1)|A''_0|$ and if $i \notin F$ for some $i \in A''_0$ then we would have $w(F) \leq |A''_1| + (n+1)(|A''_0| - 1) = |A''_1| - n - 1 + (n+1)|A''_0| < (n+1)|A''_0|$, a contradiction (i.e. one fixed activity cannot be outweighed by all flexible ones). Thus $F = I \cup A''_0$ for some $I \subseteq A''_1$ of size r_1 whence I is an r_1-insertion of $\mathcal{A}''$. Further note that F is the set of unit time activities in $\mathcal{A}''_I$. It remains to show that (the earliest start schedule) $\mathfrak{s}(\mathcal{A}''_I)$ has makespan $\leq k$. So let be P a path of $\mathcal{A}''_I$ given as set of its nodes. Then P forms a chain in $\mathcal{P}''$ and it holds $|P \cap F| \leq k$ since F is a k-family. Thus P contains at most k unit time activities in $\mathcal{A}''_I$ whence it has length at most k, from which it follows $MS(\mathcal{A}''_I) \leq k$.

Altogether the original problem of determining an optimal r_1-insertion of $\mathcal{A}$ is polynomial-time reducible to the problem of determining a maximum k-family in an appropriate weighted poset. Since this latter problem can be solved in polynomial time, the former can be as well. Thus the PAIP-case $(1,-,-|d_1^{tp} = 1)$ can be solved in polynomial time. □

The following theorem summarizes the results:

Theorem 2. *The complexity of PAIP-cases according to the (T,S,R)-scheme is as follows:*
a) $(T,-,R)$ can be solved in polynomial time for all finite T,R (maximal polynomial-time case).
b) $(T,-,-)$ and $(-,S,R)$ are the minimal NP-hard cases. Both are already NP-hard in the strong sense for their minimal parameter setting ($T = 1$ and $S = 2, R = 1$, respectively).

Furthermore, $(1,-,-)$ has the following tractable subcase:

c) $(1, -, -|d_1^{tp} = 1)$ *can be solved in polynomial time.*

Whereas, as seen, some of these results are rather straightfoward, others are less trivial and even surprising. In particular the fact that the rather restricted case with only one type, repair duration 2 and a makespan bound of 5 turns out strongly NP-hard, while reducing repair time to one yields a polynomially solvable case by exploiting the relationship to Sperner theory, points to a rich combinatorial structure which might be fruitfully investigated in further research.

4 Conclusion

In this paper we have introduced a decision problem motivated by maintenance projects: Given a maintenance project, some activities correspond to repair tasks of defective components. A repair task becomes superfluous if the corresponding component is replaced by a spare component from stock; hence spare components can reduce working time. Since the number of spares is limited, one has to decide which defective components are to be replaced to minimize duration of the maintenance project. We have given a comprehensive complexity analysis for cases defined by a parameter scheme involving the number of component types, defective components, and spare parts. Somewhat surprisingly, even the case with only one component type turns out to be (strongly NP-)hard, also if restricted further considerably. However just this case becomes tractable, if duration of repair tasks is restricted to be equal to one, the proof of which highlights a structural link to so-called Sperner theory dealing with partially ordered sets. Further research could contain the development of efficient procedures for the general problem.
In our case, we will extend the problem in another direction, more motivated by further operations in maintenance projects: Instead of simple replacement from a spare parts inventory, we will regard the situation where no spare parts are given, but an exchange between components of the same type at different positions is allowed. In this way repair tasks for defective components and dummy tasks for already intact ones can be exchanged within the project such that repair tasks can be removed from critical positions and replaced by dummy tasks. The results given here can be applied to this exchange setting since the insertion operation considered here can be simulated by exchanges: Indeed, assuming flexible activities corresponding to intact components at the start of the project (corresponding to a number of available components in stock), replacing a defective component by an intact one can be represented by an exchange between stock and actual maintenance project. Thus the insertion problem can be reduced to this exchange problem, from which follows that the model studied in the subsequent paper is already NP-hard in the strong sense under rather restricted conditions, especially already if no resources are present.

5 References

Arora, S., and B. Barak (2009): Computational complexity: A modern approach. Cambridge University Press, Cambridge

Artigues, C., S. Demassey, and E. Néron (2008): Resource-Constrained Project Scheduling: Models, Algorithms, Extensions and Applications. Wiley, Hoboken

Banghart, M. (2017): Identification of Reverse Engineering Candidates utilizing Machine Learning and Aircraft Cannibalization Data. International Journal of Aviation, Aeronautics, and Aerospace 4(4) Art. 5

Berenguer, X., J. Díaz, and L.H. Harper (1982): A solution of the Sperner-Erdös Problem. Theoretical Computer Science 21, 99-103

Berthold, L. (2021a): Project Scheduling with Activity Exchange motivated by Maintenance Opera-

tions. Unpublished paper, Institute of Operations Management, University of Hamburg (part of this thesis)

Berthold, L. (2021b): Stochastic Activity Exchange Project Scheduling. Unpublished paper, Institute of Operations Management, University of Hamburg (part of this thesis)

Bock, S., D. Briskorn, and A. Horbach (2012): Scheduling flexible maintenance activities subject to job-dependent machine deterioration. Journal of Scheduling 15, 565-578

Brucker, P., A. Drexl, R. Möhring, K. Neumann, and E. Pesch (1999): Resource-constrained project scheduling: Notation, classification, models, and methods. European Journal of Operational Research 112, 3-41

Budai, G., R. Dekker, and R.P. Nicolai (2008): Maintenance and Production: A Review of Planning Models. In: K.A.H. Kobbacy, D.N.P. Murthy (eds.): Complex System Maintenance Handbook, pp. 321-344, Springer, London

Chen, V.Y.X. (1994): A 0-1 goal programming model for scheduling multiple maintenance projects at a copper mine. European Journal of Operational Research 76, 176-191

Chen, G., W. He, L.C. Leung, T. Lan, and Y. Han (2017): Assigning licenced technicians to maintenance tasks at aircraft maintenance base: a bi-objective approach and a Chinese airline application. International Journal of Production Research 55(19), 5550-5563

Cho, D.I., and M. Parlar (1991): A survey of maintenance models for multi-unit systems. European Journal of Operational Research 51, 1-23

De Bruecker, P., J. Van den Bergh, J. Belien, and E. Demeulemeester (2015): A model enhancement heuristic for building robust aircraft maintenance personnel rosters with stochastic constraints. European Journal of Operational Research 246, 661-673

Dilworth, R.P. (1950): A decomposition theorem for partially ordered sets. Annals of Mathematics 51, 161-166

Engel, K. (1997): Sperner Theory. Cambridge University Press

Garey, M.R., and D.S. Johnson (1979): Computers and Intractability: A Guide to the Theory of NP-Completeness. W.H. Freeman, San Francisco

Grigoriu, L. and D. Briskorn (2017): Scheduling jobs and maintenance activities subject to job-dependent machine deteriorations. Journal of Scheduling 20, 183-197

Hartmann, S., and D. Briskorn (2010): A survey of variants and extensions of the resource-constrained project scheduling problem. European Journal of Operational Research 207, 1-14

Li, H., and K. Womer (2009): Scheduling projects with multi-skilled personnel by a hybrid MILP/CP benders decomposition algorithm. Journal of Scheduling 12, 281-298

Masmoudi, M., and A. Hait (2012): Fuzzy uncertainty modelling for project planning: application to helicopter maintenance. International Journal of Production Research 50(13), 3594-3611

McKendall, A.R., J.S. Noble, and C.M. Klein (2008): Scheduling Maintenance Activities During Planned Outages At Nuclear Power Plants. International Journal of Industrial Engineering 15, 53-61

Megow, N., R.H. Möhring, and J. Schulz (2011): Decision Support and Optimization in Shutdown and Turnaround Scheduling. INFORMS Journal on Computing 23, 189-204

Papadimitriou, C.H. (1994): Computational complexity. Addison-Wesley, Amsterdam

Pierskalla, W.P., and J.A. Voelker (1976): A survey of maintenance models: The control and surveillance of deteriorating systems. Naval Research Logistics Quarterly 23, 353-388

Regattieri, A., M. Gamberi, R. Gamberini, R. Manzini (2005): Managing lumpy demand for aicraft spare parts. Journal of Air Transport Management 11, 426-431

Regattieri, A., A. Giazzi, M. Gamberi, and R. Gamberini (2015): An innovative method to optimize the maintenance policies in an aircraft: General framework and case study. Journal of Air Transport Management 44-45, 8-20

Rosales, L.J.S., J.-B. Yang, and Y.-W. Chen (2014): Analysing delays and disruptions in Aircraft Heavy Maintenance. Proceedings of the 32nd International Conference of the System Dynamics Society 20.-23.07.14, Delft, Netherlands

Safaei, N., and A.K.S. Jardine (2018): Aircraft routing with generalized maintenance constraints. Omega 80, 111-122

Salman, S., C.R. Cassady, E.A. Pohl, and S.W. Ormon (2007): Evaluating the Impact of Cannibalization on Fleet Performance. Quality and Reliability Engineering International 23, 445-457

Samaranayake, P. (2006): Current Practices and Problem Areas in Aircraft Maintenance Planning and Scheduling - Interfaced/Integrated System Perspective. Proceedings of the 7th Asia Pacific Industrial Engineering and Management Systems Conference 2006, 17-20 December 2006, Bangkok, Thailand

Samaranayake, P., and S. Kiridena (2012): Aircraft maintenance planning and scheduling: an integrated framework. Journal of Quality in Maintenance Engineering 18, 432-453

Schwindt, C., and J. Zimmermann (eds.) (2015): Handbook on project management and scheduling (2 volumes). Springer, Berlin

Sriram, C. and A. Haghani (2003): An optimization model for aircraft maintenance scheduling and re-assignment. Transportation Research Part A 37, 29-48

Wang, H. (2002): A survey of maintenance policies of deteriorating systems. European Journal of Operational Research 139, 469-489

Part II
Project Scheduling with Activity Exchange motivated by Maintenance Operations

Project Scheduling with Activity Exchange motivated by Maintenance Operations

Lukas Berthold

Abstract: In the following we consider a project scheduling problem which integrates component exchange operations as motivated by a practice in aircraft maintenance. Given a defective component at some time-critical position in a maintenance project and an intact one of the same type at some noncritical position, the idea is to remove both components from their original positions and replace the defective component with the intact one and vice versa, i.e. the components interchange their positions. By this exchange operation repair work associated with the defective component can be moved to a less time-critical position and thus saved at the original position where time is critical. As a consequence, project duration may be reduced, especially if such exchanges are done for several component pairs of different types. This problem is stated as an extension of the Resource Constrained Project Scheduling Problem (RCPSP) and solved by a Branch-and-Bound approach which is shown to be superior to a MIP model solved by a standard solver.

1 Introduction and motivation

Maintenance comprises all activities to keep a technical system operable. This contains preventive measures (keeping the system in a state of high reliability, i.e. where failures occur with sufficiently low probability) as well as corrective ones (restoring the system to full operational state, if failures have been occurred) [Pintelon and Gelders 1992, Wang 2002]. In aviation, where safety is of particular importance, the measures to ensure reliable aircraft are extensive and underly strict regulations by aviation authorities like EASA (European Aviation Safety Agency) and FAA (Federal Aviation Administration) [Regattieri et al. 2015]. Accordingly, an aircraft has to pass through maintenance cycles of increasing intensity and complexity, from routine pre-flight check to heavy-maintenance D-check, which is a deep structural check carried out every six to ten years, lasts one to two months and where the whole aircraft is minutely inspected and all its systems are thoroughly tested, for which major components must be disassembled [Vieira and Loures 2016]. Maintained in this way, a typical aircraft can be in service for up to thirty years [Lee et al. 2008].

But this care and effort has its price: On average, maintenance cost amounts to $ 3.6 million per aircraft and year, corresponding to $ 1089 per flight hour [IATA 2018]; one D-check alone (even if carried out only each several years) costs about 1-2 million dollar [Philips and Hsiang 2000], for an Airbus A380 also up to 3 million [Fabig and Winter 2018]. Correspondingly, maintenance is a major cost driver for airlines, among operating expenses the third largest after fuel and flight crews [Bazargan 2016]. In 2017, airlines spent $ 76 billion for maintenance, which is about 11% of their total operating cost [IATA 2018]. Given low margins and the competitive environment in commercial aviation [Doganis 2005], there is a strong pressure to keep these costs as low as possible, especially since those for fuel and personnel cannot be influenced very much [Rosales 2015]. Correspondingly, a number of measures are undertaken to reduce maintenance cost and maintenance duration, and hence to increase aircraft utilization.

One important cost driver in maintenance are high spare parts inventories: The average value of spare parts inventories amounts to more than $ 3 million per aircraft [Kilpi and Vepsäläinen 2004]. Such quantities of tied capital result from the need to satisfy demand timely since missing parts - one essential one may suffice - may prevent finishing maintenance and hence cause delays and even subsequent flight cancellations and dissatisfied customers, leading to costly downtimes. But since the demand is typically rather irregular and hence difficult to predict for most components and parts [Ghobbar and Friend 2002+2003, Regattieri et al. 2005], the inventories must be comprehensive enough to meet unforeseen failures which may occur anywhere in the whole aircraft, among its multitude of components. Therefore an effective spare parts management is a key issue to limit maintenance costs, and a number

of measures are undertaken to keep inventories as low as possible by maintaining both high quality and safety standards at the same time. This can or will include, for example, the use of rotable inventories [Gu et al. 2015] as well as setup and maintaining of an efficient logistic network which admits emergency transshipments [Liu et al. 2018, Kilpi 2004], and even inventory pooling also between different airlines [Kilpi and Vepsäläinen 2004] and other cooperative strategies [Kilpi et al. 2009].
Nevertheless, in spite of these and other measures, it may happen that an urgently needed spare component is not available and cannot be delivered in appopriate time. In such a situation a component exchange operation usually called "cannibalization" or "component swapping" may be applied [Banghart 2017, Salman et al. 2007] : For this, assume that besides aircraft A with defective component at a time-critical position there is another aircraft B in maintenance which contains an intact component of the same type at a non-critical position; for example, maintenance of B may have just started and/or the planned finish time of maintenance is only later such that even a defective component at that position in B would not cause a delay since there is enough time to react. Then, according to that practice, both components are removed from their original positions, each component is moved to the other position and installed there (the defective one after repair or it is simply replaced if a spare has become available); in other words, both components are interchanged between the aircraft. As a consequence, aircraft A gets an intact component such that its maintenance can be finished just in time, whereas maintenance procedure of B is not largely affected since having chosen an uncritical position. Altogether, the situation at A is largely improved whereas there is only a minor disadvantage at B which may even have no effect if one has enough time to react on that additional work (if repaired) or waiting time (if delivered).
Here the essential assumption of having two aircraft in maintenance containing the same component type is more common than not since about 80% of all commercial aircraft are built by only two manufacturers, and each of the Airbus A320 and Boeing 737 families alone makes up 28% of all aircraft [IATA2018]. Moreover, especially heavy maintenance is often not directly carried out by airlines themselves but by specialized MRO (Maintenance, Repair, and Overhaul) providers which typically maintain several aircraft of different airlines at each maintenance base in parallel [Al-kaabi et al. 2007, Vieira and Loures 2016]. This further increases the probability to find exchange opportunities if needed.
Altogether, component exchange may simultaneously reduce duration of maintenance procedure (hence increasing aircraft utilization) as well as spare part inventories (hence reducing maintenance cost). In our paper we will propose and study a model which extends project scheduling by just that described exchange operation and develop and apply a solution approach.

2 Literature review

The model and decision problem introduced and studied in this paper is motivated by aircraft maintenance, more precisely by the component exchange procedure described above. Structurally it is a project scheduling problem which extends the well-known Resource Constrained Project Scheduling Problem (RCPSP), and also our study will largely follow a methodology typical for investigating a scheduling problem. Correspondingly, in the following we will review more or less related contributions in aircraft maintenance and project scheduling.

Aircraft maintenance. As mentioned in the introduction, aircraft maintenance is organized in maintenance cycles called "checks" which have to be performed after some time or number of flight hours (the values of which somewhat depending on the aircraft type). Usually they include preflight check, daily check (overnight), transit check (at every airport with a maintenance base), A- (every 300-600 flight hours/2 months, takes half a day), B- (every 800-1200 hours/6 months, takes one day; in recent years this check is often splitted between A- and C-checks), C- (every 2500-5000 flight hours/18-20 months; takes 1-2 weeks) and D-check (every 25000 flight hours/6-10 years; takes 1-2 months) [Vieira

and Loures 2016, Lee et al. 2008, Satoglu et al. 2016, Kinnison 2012].
The checks up to B-check are also subsumed as "line maintenance", whereas C- and D-checks are referred to as "heavy maintenance". These latter ones involve deep structural checks which include detailed inspections and function tests, in case of a D-check also extensive disassembly of major components and even dismantling the whole interior of the aircraft down to its rough airframe in order to inspect it for signs of corrosion and cracks, after which the aircraft also gets a new painting [Vieira and Loures 2016]. For such extensive and intensive measures, the aircraft has to be taken out of service for up to several weeks and the work is carried out in a hangar (therefore also alternatively "base maintenance").
Due to its extent and complexity, a heavy maintenance check has many characteristics of a project: It is a unique endeavour with a defined goal (namely to bring the individual aircraft from its current state into prescribed state), which must be achieved with limited amounts of time, money and resources (skilled personnel and equipment) by observing high quality standards. Further it involves a large number of interdependent tasks (several hundred or thousand), integration of different planning areas like inventory management and resource planning, uncertainties due to a considerable amount of unplanned work following findings during inspection etc. [Samaranayake 2006, Rosales et al. 2014; for general project definitions see for example ISO 21500, PMI 2017, Schwindt and Zimmermann 2015]. As a consequence, planning and performing such a large maintenance project is a demanding task [Rosales et al. 2014].
Generally, decision problems in aviation have been studied since decades. This especially holds for those problems which are directly linked with the process of schedule preparation, such as flight scheduling (schedules of flight legs), fleet assignment (assigning aircraft types to flight legs) and aircraft routing (assigning individual aircrafts of given type), crew scheduling (assigning crews to individual aircraft) [Başdere and Bilge 2014]. Here more comprehensive and realistic models of aircraft routing take maintenance visits explicitly into account to fulfill maintenance cycles. This leads to the Aircraft Maintenance Routing Problem (AMRP), which has been studied extensively [Safaei and Jardine 2018]. Sometimes the ARMP is also simply called "aircraft maintenance scheduling", which, however, does not take into account other scheduling problems within aircraft maintenance, especially those which schedule the actual maintenance activities.
This more detailed planning of actual maintenance, i.e. what has to be done within those time windows, has been mainly studied only later and in some areas the interest has only significantly increased in the last decade. Detailed maintenance planning can be divided into three main areas [Dinis et al. 2019]:
1. spare parts management 2. capacity planning (workforce and equipment) 3. task scheduling
In reality, of course, all of these issues must be taken into account adequately; however, due to the complexity of each of them, most approaches in scientific literature only focus on one or two of these areas, such that the need for a more integrated treatment has been articulated [Samaranayake and Kiridena 2012].
Here we will mainly focus on task scheduling (and models for planning heavy maintenance), at least in some broader sense, since scheduling specifically dedicated to aircraft maintenance is still a relatively sparsely studied topic, especially for heavy maintenance. The relative dearth of models and studies had been already noted by some authors ten years ago [Samaranayake and Kiridena 2012, Papakostas et al. 2010] but in the meantime the scheduling of aircraft maintenance tasks has received more attention as it is the case for planning of aircraft maintenance in general.
In [Remenyi and Staudacher 2014] the authors regard scheduling rules in a job shop environment in engine maintenance and use simulation to test their performance. An early attempt in this direction can be also found in [Cobb 1995]. Simulation is also used in [Regattieri et al. 2015] to find an optimal mixture of maintenance policies (preventive and corrective) taking into account the failure rates of components. The objective is to minimize total costs which contain costs for spare parts stock,

maintenance crew and losses due to unplanned downtimes. A quantitative planning approach for line maintenance is proposed in [Papakostas et al. 2010]: Given an aircraft and a set of tasks, it has to be decided on which of the next airports which subset of tasks is to be performed. The resulting alternatives are evaluated according to different criteria like costs, delays, operational risk and remaining useful time of components.
Focusing more on heavy maintenance, [Rosales 2015, Rosales et al. 2014] use system dynamics to analyse disruptions in heavy maintenance especially due to unplanned tasks. Here dependencies between relevant factors are described by feedback loops, resulting in causal loop diagrams which are the basis for simulation models helping to understand the dynamics causing delays of maintenance procedures and to identify measures to prevent them.
To schedule an aircraft maintenance project, the Critical Chain Project Management (CCPM) developed by [Goldratt 1997] has been advocated by some authors [Srinivasan et al. 2007, Kulkarni et al. 2017, Junqueira et al. 2018]. CCPM is a project management tool which combines critical chain method (an extension of classical critical path method if resources are present) with a special buffer management: Instead of assigning a buffer to each activity, the activity durations are estimated tightly ("aggressively") and a common ("feeding") buffer for the whole project is used. The idea is to avoid that employees anticipate buffers for single activities and take them explicitly into account in their planning. However, this method has also received some critics [Herroelen and Leus 2001]; so, for example, its ability to cope with resources is only rather limited [Herroelen et al. 2002].
As soon as resources are fully taken into account in project scheduling, one gets an RCPSP or some variant. The RCPSP extends classical CPM project networks by equipping them with resources (for somewhat more details see below in the project scheduling part). Accordingly, RCPSP or extensions have been proposed by a few authors also for heavy aircraft maintenance.
A fuzzy set approach is proposed in [Masmoudi and Hait 2012]. Based on the experience that probability distributions for activity durations are usually not available such that one has to rely on inevitably vague estimations of experienced employees, the authors extend the RCPSP by describing activities durations with fuzzy sets. In fact two planning levels are considered: First, on tactical level, a Rough Cut Capacity Problem (RCCP) problem has to be solved, then, on operational level, an RCPSP.
Another extension of the RCPSP which is particularly useful in aircraft maintenance takes skills of workers into account. Indeed, due to the high safety standards, each task in aircraft maintenance is allowed to be performed only by licensed workers [Yadav 2010]. Additionally there may be different levels of experience due to which a task can be done with different pace. Such Multi-Skill RCPSPs have been suggested first in [Neron 2002] and studied in some variants since then [Li and Womer 2009, Firat and Hurkens 2012, and others]. The applicability on aircraft maintenance is emphasized in [Stadnicka et al. 2017]. Skills are also naturally taken into account in workforce planning models [De Bruecker et al. 2015], but there usually not taking into account any project structure, instead working with shifts to which workers are assigned.
Finally it should be remarked that models for maintenance planning and scheduling are also studied for other industries, involving systems like railway tracks [Budai et al. 2006], ships [Go et al. 2013], power plants [Froger et al. 2016], and highways [Wang et al. 2002]. Since this concerns large, complex systems as well, also a few project scheduling models can be found among the corresponding literature, so in [Chen 1994, McKendall et al. 2008, Megow et al. 2011]. Of course, beside more generic conditions these models contain more application-specific ones which cannot be translated to other application areas.
Another field where the term "maintenance scheduling" is used, are extensions of more classical machine scheduling models which take maintenance windows or durations after which maintenance must be performed into account; see for example [Grigoriu and Briskorn 2017].

However, the exchange procedure outlined in the introduction has not been considered in any of the approaches thus far, nor, to the best of our knowledge, has it found entrance in any scheduling model.

Instead, it has been mainly studied under the term "cannibalization" since the late 1960s in context of stochastic maintenance and reliability models for multi-component systems (often treated more general, but aircraft have been often quoted as a primary example and motivation). The first, most basic models proposed then are also most interesting for our setting whereas further development of this field took a different direction especially towards stochastic models for reliability considerations. The very first mathematical model has been studied in [Hirsch et al. 1968] and has been motivated by military practices (from which also the term "cannibalization" comes). Since some elements will also occur in our model in a similar way (albeit in a rather different setting and framework), we review this model in somewhat more detail, already using our slightly different terminology: A system is given by n positions ("loci" in Hirsch et al.) each of which contains a component ("part") of certain type which may be intact ("sound") or defective ("failed"). Different components may have the same type such that the set of positions is partitioned in subsets corresponding to the types. The system can attain performance states $\{0, \dots, M\}$, where state 0 corresponds to complete failure, state M to perfect functionality, and the remaining ones being intermediate and ordered by increasing functionality. This state of the system is determined by the components' states via a structure function $\phi : \{0,1\}^n \to \{0, \dots, M\}$, where a 1 at position k indicates an intact component, 0 a defective one. Further this structure function is assumed to be monotone, i.e. if $A \subseteq B$ then $\phi(1_A) \leq \phi(1_B)$ for their index functions, that means making defective components to intact ones by repair or replacement does not decrease the functionality of the system but possibly increase it. In other words, the monotonicity property captures the reasonable assumption that repairing or replacing a defective component cannot make the system worse. However, cannibalization is especially considered if just these possibilities - repair or replacement- are limited or currently not available. So in the basic model no spares are available, instead components of the same type can be interchanged, i.e. they can change their positions to increase system performance. Indeed, a real increase by such an exchange occurs if the same part type occurs at different positions some of which are more important than other ones for the system performance. This can be also expressed by the structure function, such that interchanges between components of same type may actually increase the state. In the further course of their study Hirsch at al. prove a representation theorem and study a stochastic model based on their initial assumptions. This basic model has been subsequently extended and modified, so further (intermediate) states were introduced also for the components, exchange between components of the same type has been restricted, cannibalization has been considered as only one possibility beneath repair and/or replacement by spares, costs have been considered, analytical solutions and simulation have been applied etc.; a systematic overview on different cannibalization models, their objectives and methods applied can be found in [Werbińska-Wojciechowska 2019, pp. 81-84].
For our model only the basic concepts of positions, components, component types and exchanges will be relevant, whereas structure function and reliability considerations have no counterpart in our scheduling setting; we will also not regard systems built from serially and parallely combined components, instead positions will correspond to activities the duration of which depends on the state of the component which is processed there; and whereas in the cannibalization models exchanges serve to increase reliability, component exchanges will reduce project duration in our case.
After this overview on literature on (aircraft) maintenance models we turn to project scheduling next.

Project scheduling. In the most basic project scheduling problem, projects are represented by acyclic directed graphs the nodes of which represent activities and the arcs the precedence relations between them; further each node is labelled with the duration of the corresponding activity (this is the so-called activity-on-node representation (AoN); alternatively, but less frequently used is the activity-on-arc representation (AoA), where activities are represented by arcs whereas nodes correspond to "events"). The aim is to find a schedule of start and end times which minimizes project duration (makespan). Corresponding models have been first suggested and also practically applied already in the late 1950s

[Kelley and Walker 1959, Malcolm et al. 1959, Roy 1959] together with efficient methods to determine project duration, critical paths of activities and further additional informations like earliest and latest start and end times and slacks of activities, altogether resulting in what is today known as the Critical Path Method (CPM). Since then CPM has become a standard tool in project management (today mostly used in commercial project management software) and it can be found in any introductory text to Operations Management and Operations Research (see for example [Domschke et al. 2015]). Moreover the model has been the basis for each project scheduling problem subsequently proposed until now.
The practically most important extension concerns resources, and correspondingly the need to take resources into account has been articulated only few years after formulation of the basic model [Wiest 1963, Moshman et al. 1963], which eventually led to the Resource Constrained Project Scheduling Problem (RCPSP). In the RCPSP, an instance is given by a standard project network extended by resource related information: Each activity has a certain resource consumption and each resource is available in a certain amount; in general there are several different resources. Resources are assumed to be renewable, i.e. their available amount (capacity) is the same for each time slot throughout project duration (as manpower, whereas a fixed budget for a project would be a non-renewable resource). The objective is to find a schedule, i.e. an assignment of start and end times to activities, which respects precedence and resource constraints; as far as the resources are concerned, this means that total resource consumption by activities does not exceed resource capacities for each time slot and each resource type.
Due to its practical relevance, its many extensions and the challenges it offers, the RCPSP has attracted many researchers as reflected by a number of surveys [Brucker et al. 1999, Hartmann and Briskorn 2010, Abdolshah 2014, Habibi et al. 2018] and books [Brucker and Knust 2006, Artigues et al. 2008, Schwindt et al. 2015] dedicated to the RCPSP and its variants.
However, in contrast to the efficient critical path methodology for standard project networks, the RCPSP has turned out to be a rather hard optimization problem, it is NP-hard in the strong sense [Blazewicz et al. 1983] and also difficult already for relatively small instances: From the currently most used set of benchmark instances, the PSPLIB (Project Scheduling Problem Library) [Kolisch and Sprecher 1996, online available under www.om-db.wi.tum.de/psplib/]), there are still some instances with 60 activities which have not been optimally solved yet, such that the optimal solution of real even medium-sized instances will be currently out of reach in many cases. The PSPLIB will be also the basis for generating instances for the problem studied in the following.
In spite of its practical relevance and the scientific interest it has attracted, also some limitations of the RCPSP have been noted and have become starting point for new variants and models. With respect to our model especially those will be of interest which try to make the in some sense rigid project structure more flexible, see below.

One variant which has some basic similarity with our model is the multi-mode extension of the RCPSP (MM-RCPSP, see for example [Weglarz et al. 2001, Van Peteghem and Vanhoucke 2014]). Here the idea is that an activity can be processed in different execution modes: For example, often an activity could be carried out within less time if more and/or other resources are used. Accordingly, each such combination of duration and consumed resources would correspond to a *mode* in which an activity can be executed. In our model some ("flexible") activities can be also performed as dummy task or as repair task and hence also in two "modes". However, the dummy task would be always preferable (since taking no time and consuming no resources) if there would not be further constraints limiting its availability. In fact, while in the case of MM-RCPSP all modes can be chosen independently for each activity, in our case only a certain number can be performed in the favourable dummy mode (namely corresponding to the limited number of intact components). There are also modifications which take certain dependencies into account, most notably the mode identity variant where activities of some subsets have to be executed in the same mode [Salewski et al. 1997]. At first sight, such subsets

are somewhat similar to what will be (component) types in our setting, but the dependency is rather different: In our case only a fixed number of activities (of given type) can be performed as dummy tasks, corresponding to the intact components, whereas the other ones have to be performed as repair tasks (corresponding to defective components); a corresponding model has been regarded in our preceding paper [Berthold and Fliedner 2021]. But also this is still a relaxation to what is considered in the following since now we will also take into account when a component actually becomes accessible (and hence, in a sense, when a "mode" becomes available). Thus the dependency considered here is a different and also more involved one than that one assumed in case of mode identity.

Further there is also a certain connection to approaches which aim to equip the RCPSP with more flexibility, especially with the capability to deal with alternative execution paths. Indeed, one point of criticism of the RCPSP is its relatively rigid project structure: All activities have to be executed and the precedence relations between them are also fixed. In contrast to this, often there is more than one way to achieve a certain (sub)goal, each path corresponding to a different set of activities. In some sense also the multi-mode extension is a variant which tries to make the RCPSP more flexible, but only a rather restricted one at the level of activities, not on a higher structural level which would allow to choose alternatives.
The most basic and straightforward tool to equip a project network with more structural flexibility is by use of logical relations. Indeed, in traditional project scheduling models precedence relations are interpreted to be associated with a logical AND: *All* successors of an activity must be carried out and an activity can only start if *all* predecessors have been finished; consequently also all activities have to be performed. However, in real applications one would often like to model that only one of several possible successors needs to be actually executed or that an activity can already start if only one preceding activity has been finished, i.e. corresponding to an (X)OR-interpretation of precedences. This would also admit to model alternative execution paths.
First project scheduling models with logical operators have been early proposed [Elmaghraby 1964], but in greater breadth and detail such approaches have been studied only later (and at first still without resources). So [Gillies and Liu 1995] give a complexity analysis of optimization for activity networks with logical relations which reveals that such problems become NP-hard already for rather simple and restricted cases. To solve related problems, [Beck and Fox 2000] present a constraint propagation approach based on a concept called probability of existence (referring to the probability that an activity will be actually contained in the final solution). [Capacho and Pastor 2005] consider an assembly line balancing problem with flexible precedence networks of tasks; and [Barták and Čepek 2008, Barták 2010] regard so-called P/A graphs (where P="parallel" essentially corresponds to AND and A="alternative" to XOR) and Temporal Activity Networks, also giving complexity results for their model. Further work in that direction can be also found in [Kis 2003].
Such models also formed the background for approaches aiming to extend the RCPSP by the capability to deal with alternatives - a line of research which has received increasing interest especially in the last decade. So, motivated by the disruption management of the aircraft turnaround process, [Kuster et al. 2010] presented a model called x-RCPSP (extended RCPSP), which incorporates XOR and mutual exclusion as logical operators. Intended for a more general applicability, [Kellenbrink and Helber 2015] proposed their "RCPSP-PS" (RCPSP with flexible project structure) and applied a genetic algorithm. Another model based on Petri Nets has been proposed in [Čapek et al. 2012], whereas in [Vanhoucke and Coelho 2016] a SAT solver approach is applied to the RCPSP equipped with logical operators (RCPSP-Log). [Tao and Dong 2017] formulated a version RCPSP-AC (RCPSP with activity chains) which is solved with a Simulated Annealing approach. Finally [Servranckx and Vanhoucke 2019] proposed a variant called RCPSP-AS (RCPSP with alternative subgraphs), which explicitly takes also the possibility of nested and linked alternative paths into account. Based on these two dimensions they also present a systematic generated benchmark set on which a tabu search approach has been tested and which they recommend for further use in that area.

Altogether one can note that some of this approaches are rather similar and seem to differ more in terminology and modeling features than in actual problem substance. However, even in case of principal (near)equivalence of some models, there may be some practically relevant difference; whereas some deal still with basic ("low level") logical constraints, other ones may be more convenient and intuitive to model (on a "higher level") alternatives as occuring in projects such that the advantage of a model compared with another one may be not completely reflected by logical equivalence alone. In any case, it can be also pointed out that still there is no established reference model for the RCPSP with alternatives and flexibility, though there is an increasing trend to become aware of the work and the models of other authors in this area.
Just as the listed approaches, our model also seeks to enhance the standard RCPSP with more flexibility. However, there are some important differences: So in our model, as in standard RCPSP, all activities are actually carried out and all precedence relations have to be satisfied. The flexibility, however, comes from the possibility to move work contents (as given by duration and resource consumption) between activities of the same type, reflecting the exchanges of intact and defective components between different positions. Thus our approach is not suitable (and also not intended) to model alternative project paths, but conversely the models mentioned above are also not appopriate to capture the type of flexibility necessary for the specific exchange situation described in the introduction (at least as soon as a larger number of components and different types are present); indeed, straightforward conversion using logical relations would lead to an exponential blow-up of the instance, hence turning out to be prohibitive. From this point of view, our model can be also seen as model to enhance the RCPSP with a restricted flexibility just sufficient for our specific problem, but at the same time also as an interesting example which is not covered by current flexible extensions of the RCPSP. Nevertheless it may be conceivable that the general idea to move work content across the project network, from one position to other one, may be applicable in other settings beyond the more special exchange procedure in aircraft maintenance. From this point of view, the model presented in the following may also encourage thinking on how flexibility of projects can actually look like.

3 Problem definition

3.1 Problem description and examples

The problem presented in the following can be understood as standard project scheduling problem extended by the capability to carry out exchange operations between components as described in the introduction. For this, we will use project networks with two kinds of activities: On the one hand activities related to components (or, rather, to component *positions*) which can be involved in exchange operations, and on the other hand the remaining activities. Due to exchange operations, i.e. replacing defective components by intact ones (or vice versa), activities of the first kind can alter their initial characteristics (duration, resource consumption) - whence we will call them "flexible" - whereas those of the second kind are simply modelled as standard activities with predetermined characteristics, whence we will call them "fixed". Correspondingly, the flexible activities will play a key role for our problem.
Before stating the problem in its full and formal generality, we regard two examples at first. There and throughout the whole paper we only use activity-on-node (AON) networks for representing the project structure. For simplicity, resources are omitted in the following examples. We start with a minimal example illustrating the general idea, namely to remove a defective component and the associated repair task from a time-critical position in the maintenance project network by moving the defective component to a less critical position and replace it by an intact component from that position (which does not require additional repair work). The initial situation looks as follows:

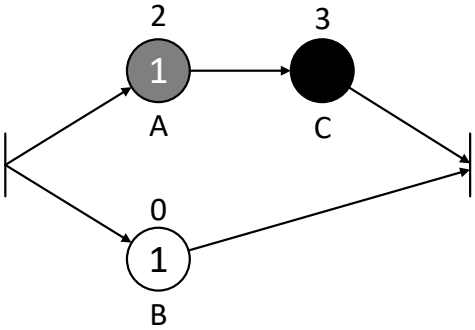


Figure 1: Project network without exchange

Here the numbers within the nodes of A and B indicate that these activities are related to components of type 1, each activity corresponding to one component. Such a component can be either defective (grey) or intact (white), and depending on this operational state the corresponding activity consists of repairing the component or having nothing to do. Correspondingly, activity A, associated with a defective component, is processed as repair task with some non-zero duration, whereas activity B, associated with an intact component for which there is nothing to do, is "processed" as dummy task of duration 0 (Although seeming superfluous at first sight, such dummy activities will be useful (and natural) to represent intact components which are possibly involved in component exchange). Activity C is any other activity not associated with any component (at least not in a manner relevant for our problem). Without any component exchange, i.e. seen as standard project network, the project duration is 5: Starting at $t = 0$, the component associated with A is repaired whereas there is nothing to do for the component associated with B, whence activity B immediately finishes. After A has finished at $t = 2$, activity C can start and ends at $t = 5$, and hence the whole project.
However, if component exchange is allowed (and the exchange procedure does not take too much time) then the project duration can be reduced: In this case components associated with A and B are removed from their original positions in the system, each is moved to the position which initially contained the other one, where they are finally installed (at B after repair for the component originally coming from A). So the component from A is moved to, processed and installed at B and vice versa. Here and in the following we assume that the whole exchange procedure takes only negligible time compared with activity durations, i.e. time zero. After that exchange we get the following situation:

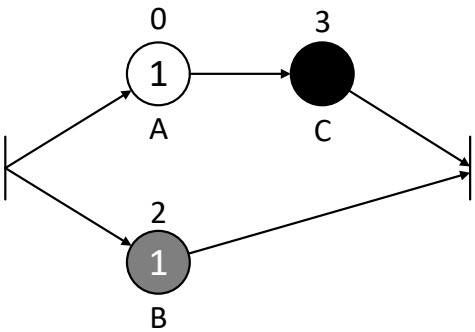


Figure 2: Project network after exchange

In this case, starting at zero, the exchange is immediately performed and finished at $t = 0$. Then the activities are processed, now activity A as dummy task and activity B as repair task. A finishes immediately whence C can start still at $t = 0$. Then B ends at $t = 2$ and C at $t = 3$ whence the whole

project ends at 3 (instead at 5 as before). Thus the exchange option may reduce project duration indeed.
Implicitly we have assumed that the exchange of components is also reflected by an exchange of work between activities within the project network. This is actually the case if flexible activities are associated with component positions rather than with individual components (above, we have related A and B simply to "components", still not making a clear distinction there but assuming the reader to have the "right" intuition). To illustrate the difference between these notions for an aircraft, the "right wheel of the landing gear" is a component position, which may contain the wheel with serial number, say, 123-4567 as individual component. Of course, component position and individual component must fit together, they must be of the same type (here the type "wheel", or possibly more precisely the type "wheel with specification xyz"). To illustrate the relationsship further, note that the individual component at some component position may be replaced by another one (of the same type) whereas the component position remains the same (unless the structure and construction of the aircraft is changed, which we will not assume). For convenience, we will often say simply "component" for individual component and "position" for component position.
Having clarified this distinction, in the following we will associate flexible activities with component positions (as already done above, albeit still implicitly) instead with individual components. This can be justified since in modelling a maintenance project as project network, many precedence relations will reflect spatial dependencies, i.e. sequences of activities at neighbouring, connected component positions. As a whole, the structure of the maintenance project will highly reflect the structure of the system under maintenance (aircraft) whence it seems natural to represent positions in the system by activities in the project. - Since a flexible activity represents a component position we will often identify them and refer to them by the same letter, hence speaking of activity A, position B, the duration of C, the component at D etc. This will not cause any difficulties since it will be clear from the context or simply by the corresponding attribute if an activity or a position is meant.
In reality a component position can be also temporarily empty, i.e. containing no component, for example during maintenance; however, in the operable system each essential component position contains a functioning individual component, and also for the system to be maintained each component position typically contains a component at the beginning, even though not necessarily an intact one. So we will assume that each component position initially contains and finally has to contain a component which will differ if the original component was replaced by another one of the same type, i.e. if the component and the position were involved in an exchange.
The distinction between component position and individual component corresponds to another distinction which we make, that between (flexible) *activity* and *task*: Here a task is the work associated with an individual component: If this component is defective, then it must be repaired, hence it is associated with a repair task. If it is intact, then there is nothing to do, and it is associated with a dummy task of duration 0. Essential for our scheduling problem is that if a component is moved to another position, the task is moved as well to be processed there. At that new position the task defines the work content of the associated flexible activity. Namely, the work associated with a flexible activity is defined by the task to be done for the component which is finally found at the corresponding position. Thus a flexible activity is processed as repair task, if the component initially found at the corresponding position is defective and remains there, or if this original component is replaced by another one which is defective. Correspondingly, a flexible activity is processed as dummy task and hence taking time zero, if the component initially found at the associated position is intact and remains there, or if this component is replaced by an intact component. For short, the task processed at some position defines the work content and hence the duration of the flexible activity representing that position. Concerning that duration, we assume that the duration of a repair task is only determined by its type (more precisely by the type of the component), i.e. the repair time for defective components of the same type is the same (but can and typically will differ for components of different

types). In the example above, flexible activity A is processed as dummy task if no exchange has been carried out, and as repair task, if the exchange has been performed.

Example illustrating availability

Next we regard an example which illustrates a further important aspect which has to be taken into account, namely the *availability* of components. Indeed, in reality often a component at some position is not immediately accessible and hence cannot be removed just at the beginning but only after other activities have been done before (like dismantling, for example). We will identify that point in time when a component becomes accessible and hence available for exchange operations with the point in time at which the flexible activitiy, which is associated with the position, where the component is initially contained, becomes eligible, i.e. at which all its predecessors have been finished. The following example clarifies this point:

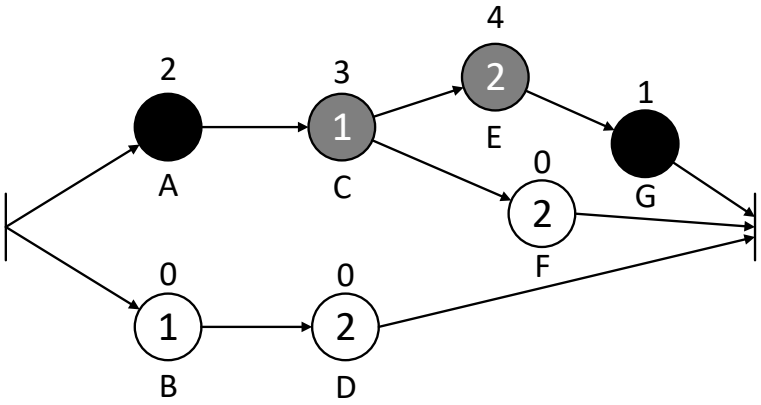


Figure 3: Project network with two types

Here activities B and C are associated with component positions for components of type 1, D, E and F with positions of type 2. The components initially found at B, D and F are intact, whereas those at C and E are defective (and repairable within time 3 and 4, respectively). Without any exchange the project duration is 10.
Again the duration can be reduced if component exchanges are allowed, in this case applying an exchange between B and C as well as between E and F. However, now we additionally have to take into account *when* the corresponding components actually become accessible. As mentioned above, we assume that a position and the component initially contained there become accessible if the corresponding flexible activity becomes eligible, i.e. if all its predecessors have been finished. Only then a component is available for exchange, i.e. to be removed from its original position and to be moved to another one (and to be replaced by another component). Thus it is not possible to exchange the components between B and C simply by exchanging the initial operational states - now defective at B, intact at C - such that B could start at 0 as repair task for a defective component and C at 2 as dummy task for an intact component; this is not possible since the intact component initially at C (as position) becomes accessible not before C (as activity) becomes eligible, i.e. not before $t = 2$, whence the defective component cannot be moved to B before $t = 2$. Thus, intuitively, the following initial part of a schedule would be *not* feasible:

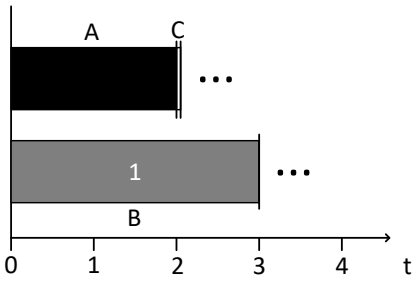


Figure 4: Infeasible schedule

Instead, a feasible exchange would be the following one: At time 0 activities A and B become eligible. A is started, but B is not since the only available component at 0 is that one which is already contained at B whence starting B would mean to process the intact component there and hence to finish B immediately. But then the intended exchange with C would not be possible anymore. Thus, instead, the start of activity B is delayed up to $t = 2$: Then A finishes and C becomes eligible whence the defective component contained there becomes accessible. Only now the components between positions B and C can be exchanged, each is removed from its original position, moved to the other one and processed there, whence B is processed as dummy task and C as repair task of duration 3:

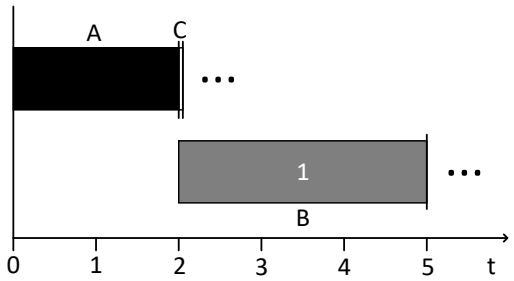


Figure 5: Feasible schedule

Having finished C already at $t = 2$, activities E and F become eligible and hence the the corresponding components accessible. Exchanging them moves the repair task of duration 4 to position F whereas E is processed as dummy task whence it finishes immediately such that G can start still at $t = 2$. Then the remaining schedule is as follows: At $t = 3$ activity G ends, at $t = 5$ activity B, whence D starts and ends immediately. Finally, at $t = 6$ activity F ends and hence the whole project. As a result, one gets a makespan of 6 instead of 10 without exchange.

Component counter functions

This feasibility w.r.t. exchanges can be also expressed using a "component counter function" which counts the number of available components at given time and thereby ensuring that no more components are consumed than having become available up to this point. The idea is to keep records on the number of components which become accessible (since their initial position has been reached, i.e. the corresponding flexible activity has become eligible) or unavailable (since their processing has been started at some position), from which we get how many components are available at given time, namely as the difference, the number of components having become accessible minus the number having become unavailable (or "consumed", as we will often say) up to a given point. This net value has to be non-negative, just meaning that not more components may be processed than having become available up to this point. This bookkeeping is done for each combination (τ, σ), where τ is a type and $\sigma \in \{0, 1\}$ the operational state. Contrary to common use, in our setting it will be more convenient

to label the intact state with "0" and the defective state with "1", as in this case state 0 is associated with a task of duration 0, and 1 with a task of duration >0. This will simplify some notation in the following.
Applied to the example above, the course of the counter functions $c_t(\tau, \sigma)$ (one for each (τ, σ)-combination) is as follows:

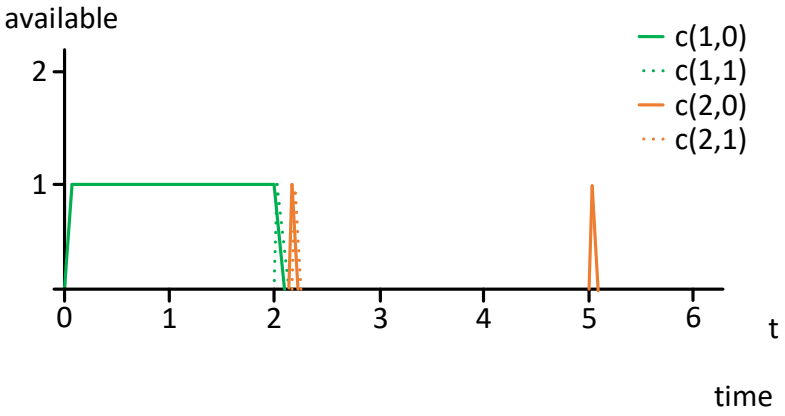


Figure 6: Example of counterfunctions

At the beginning, the value $c_t(\tau, \sigma)$ is zero for all combinations, but jumps to 1 for combination $(1, 0)$ as the intact component at B becomes accessible still at 0. If this component would be processed at B and B started at 0, then the counter would increase to 1, and immediately decrease to 0 again (component is not available any more, since it has been consumed just after having become accessible). However, for later exchange the component is not processed yet such that the (τ, σ)-counter stays at 1 until $t = 2$. Then the component initially at C becomes accessible whence the $(1, 1)$-counter also jumps to 1. However, still at $t = 2$, both components are moved to their new positions and their processing is started. Thus they become unavailable still at $t = 2$ and the counters for combinations $(1, 0)$ and $(1, 1)$ drop to 0 again. C, now (as position) with intact component, is (as activity) immediately processed and finished such that E and F become available whence both counters $c_t(2, 0)$ and $c_t(2, 1)$ jump to 1 due to intact and defective components at F and E, respectively. But since they are immediately exchanged and installed, the counters drop to 0 again still at $t = 2$. Finally, at $t = 5$, processing of the defective component at B finds an end such that D becomes eligible and the intact component there available, such that $c_t(2, 0)$ jumps to 1 again; but since the component remains there and is immediatly processed, the counter drops to 0 still at $t = 5$. Since until the end of the project at $t = 6$ (when the repair of the defective component at F finishes) no further flexible activity becomes eligible, the counters remain at 0.
Regarding the graph of these counter functions, one can see that such a function may actually take two values for the same t. To avoid such complications, we will use two counter functions: The *availability counter* $ac_t(\tau, \sigma)$ records the number of (τ, σ)-components available at t immediately after new components have become accessible at t and immediately before components are started to be processed at t, whereas the *component counter* $cc_t(\tau, \sigma)$ records the number after both events, i.e. after new component have become accessible *and* after components are started to be processed, hence becoming unavailable.
Using both these counter functions, we get $ac_0(1, 0) = ac_1(1, 0) = ac_2(1, 0) = 1$, $cc_0(1, 0) = cc_1(1, 0) = 1$ (but $cc_2(1, 0) = 0$), $ac_2(1, 1) = 1$ (but $cc_2(1, 1) = 0$); $ac_2(2, 0) = ac_2(2, 1) = 1$ (but $cc_2(2, 0) = cc_2(2, 1) = 0$); and $ac_5(2, 0) = 1$ (but $cc_5(2, 0) = 0$). For all other cases the counters are zero.

Compared with a list of explicit exchanges (like $A \leftrightarrow E$, $C \leftrightarrow H$, $K \leftrightarrow R$, ...), the advantage of

such counter functions will be their easier handling as soon as one tries to define feasibility w.r.t. exchanges for a formal definition of the scheduling problem and its representation as a mixed-integer program (MIP). Furthermore sometimes several different exchanges may lead to the same result: For example, let be given flexible activities A, B, C, D of type 1, positions A and B initially with intact components, C and D with defective ones; further assume that they become eligible at the same time t. Then exchanges $A \leftrightarrow C$ and $B \leftrightarrow D$ lead to the same result as exchanges $A \leftrightarrow D$ and $B \leftrightarrow C$ - both alternatives are equivalent for our problem. Instead, expressed via component counters, 2 intact and 2 defective components of type 1 become available at t and the same are immediately consumed. Thus using component counters may reduce degrees of freedom when trying to solve a problem instance and hence restricting the solution space. Thus in the following we will prefer to use those component counter functions to characterize feasibility, and correspondingly we will prefer to speak of "feasibility w.r.t. insertion" instead of "feasibility w.r.t. exchange" since this better reflects the character of the feasibility condition.

Before turning to the general problem definition and as preparation for it, we give a summary of the main elements and assumptions of our model:

- A problem instance basically consists of a project network with two kinds of activities called "fixed" and "flexible".
- The objective is to minimize project duration (makespan).
- Each flexible activity corresponds to a component position which initially contains a component of suitable type which is either intact or defective.
- For each component its operational state is known at the beginning.
- A defective component is associated with a repair task, an intact component with a dummy task of duration zero. Repair tasks associated with components of the same type have the same duration.
- A component can be removed from its initial position, moved to another one of the same type and installed there (after having been repaired if it was defective).
- Such exchange operations, i.e. removing, moving to another position and installing there, are assumed to take no time (negligible compared with repair times and other activity durations).
- If a component is moved to another position, the associated task is moved there as well and processed there.
- A position and the component initially found there become accessible if the corresponding flexible activity becomes eligible, i.e. its predecessors have been finished.
- A flexible activity is processed as the task associated with the component finally assigned to the position associated with that activity.
- The component actually processed at some position determines via its associated task the work content of the flexible activity representing that position.

Now we are prepared to state the problem generally.

3.2 Problem definition

An instance $\mathcal{E}$ of the Activity Exchange Project Scheduling Problem (AEPSP) is given by following data:

- a set of *activities* $A = \{0, \ldots, n+1\}$ where 0 will act as dummy start activity and $n+1$ as dummy end activity
  (in our examples we prefer to use letters for activities to avoid the use of too many numerical values, which could be a source of confusion)

- a binary *precedence* relation $P \subseteq A \times A$ such that (A, P) is an acyclic digraph and 0 (resp. $n+1$) is direct or indirect predecessor (resp. successor) of all other nodes. For an activity $i \in A$, let

P_i (resp. S_i) denote the set of direct predecessors (resp. successors) of i.

- a set $\mathcal{T} = \{1, \dots, \bar{\tau}\}$ of *types*
- a partition $A = \bigcup_{\tau=0}^{\bar{\tau}} A_\tau$ (i.e. with $A_\tau \cap A_{\tau'} = \emptyset$) where
  $A_0 =: A^{fx}$ is the set of *fixed* activities (with $0, n+1 \in A_0$),
  A_τ is the non-empty set of *activities of type* τ for $\tau = 1, \dots, \bar{\tau}$, the union of which is
  $A^{fl} := \bigcup_{\tau=1}^{\bar{\tau}} A_\tau = A \backslash A^{fx}$ the set of *flexible* activities
  $\tau(i)$ denotes the type of flexible activity i, i.e. it holds $i \in A_{\tau(i)}$
- *durations* of fixed activities $d_i \in \mathbb{N}_0$ for $i \in A^{fx}$ with $d_0 = d_{n+1} = 0$
- *durations of flexible types* $\tilde{d}_\tau \in \mathbb{N}$ for $\tau = 1, \dots, \bar{\tau}$
  (These durations for flexible types refer to the time needed for repair taks, i.e. to restore defective components of given type. The dummy tasks for intact components need no time.)
- a set of renewable *resources* $\mathcal{R} = \{1, \dots, \bar{\rho}\}$ with resource capacities $(R_\rho)_{\rho \in \mathcal{R}}$
- resource consumption values $(r_{i\rho})_{i \in A^{fx}}$ for fixed activities and $(\tilde{r}_{\tau\rho})_{\tau \in \mathcal{T}}$ for types, for each $\rho \in \mathcal{R}$.
  (As for the processing times, the values for flexible types refer to resources consumed by repair tasks, i.e. to restore defective components. The dummy tasks for intact components need no resources.)
- initial state value function for flexible activities $\sigma_0 : A^{fl} \to \{0, 1\}$
  (An $i \in A^{fl}$ with $\sigma_0(i) = 1$ corresponds to additional work (repair task) associated with a defect component initially at that position whereas $\sigma_0(i) = 0$ corresponds to no additional work (zero task) associated with an intact component.)

Applying the notation to the second example above, we have $\mathcal{T} = \{1, 2\}$, $A_1 = \{B, C\}$, $A_2 = \{D, E, F\}$, $\tilde{d}_1 = 3$, $\tilde{d}_2 = 4$, no resources, $\sigma_0(C) = \sigma_0(E) = 1$ and equal to 0 for the other activities B, D, F in A^{fl}.

3.2.1 Schedules, decision points, associated activity sets

Having formally specified instances in this way, we turn to the actual scheduling problem now. In order to state the scheduling problem rigorously and clearly, we have to define (partial) schedules and feasibility conditions for them at first, and this will especially mean to capture the right feasibility condition for exchange operations: Since, as seen before, subsequent decision points can fall onto the same time due to activities of duration zero and the number of available components can change from one to the next decision point and hence within the same point of time, it will be necessary to introduce finer auxiliary time steps to avoid some undesirable effects contradicting a logical sequence of events. But before we state what can be straightforwardly translated from other (project) scheduling problems, especially from the RCPSP.

The bare definition of a (partial) schedule, i.e. still without feasibility conditions, is straightforward:

A *(partial) schedule* $\mathfrak{p} = (S, s, f, \sigma)$ is given by following data:

- a set of *scheduled activities* $S \subseteq A$ which is closed under precedence, i.e. if $j \in S, i \in P_j$ then $i \in S$. If $S = A$ then the schedule is *complete*.
- start $s(i)$ and finish times $f(i)$ for each $i \in S$ (with values in $\mathbb{N}_0$)

- a state $\sigma(i) \in \{0,1\}$ for each $i \in S \cap A^{fl}$.
  (This state function indicates at which positions defective components are actually repaired (value 1), i.e. which flexible activities are processed as repair tasks, whereas the other positions get intact components, such that the corresponding activities are processed as dummy tasks of duration 0 (value 0). In general it differs from the initial function σ_0 due to later exchanges.)

Where necessary, we will specify the schedule an object is associated with by a superscript like $S^{\mathfrak{p}}$, $f^{\mathfrak{p}}$, etc.

As the notion of a schedule, that of a decision point was already used above as well, now we define it formally:

A *decision point* is given by a pair $\delta = (\mathfrak{p}, t)$ where $\mathfrak{p}$ is a partial schedule and t a point in time such that $s(i) \leq t$ for each $i \in S^{\mathfrak{p}}$. If the schedule is clear, we will often refer just to the mere time t as "decision point".
Further, given a schedule $\mathfrak{p}$ and a time t, let $\mathfrak{p}|_t$ denote the schedule $\mathfrak{p}$ restricted to activities starting strictly before t, i.e. with $S^{\mathfrak{p}|_t} = \{i \in S^{\mathfrak{p}} | s(i) < t\}$ and agreeing with $\mathfrak{p}$ on s, f, σ. Accordingly, t induces on $\mathfrak{p}$ the decision point $\delta = (\mathfrak{p}|_t, t)$.
As the name suggests, a decision point describes a situation where one stands at some point t at which several activities have been already started before (forming the partial schedule $\mathfrak{p}$) and one has to decide which (and if any) activities are to be started now, just at t.

Associated with a schedule $\mathfrak{p}$ are the following activity sets: An activity $i \in S$ is *active* in time slot $[t, t+1]$ or simply $[t]$ if $s(i) \leq t$ and $t+1 \leq f(i)$. The corresponding activity set is denoted by $A_{[t,t+1]}$ or short $A_{[t]}$. If t extends $\mathfrak{p}$ to a decision point $\delta = (\mathfrak{p}, t)$, then we also write A_δ for this set of activities which are active at δ (as opposed to those which are altready finished or not scheduled yet). An activity i is *finished* at δ if $f(i) \leq t$ (corresponding set denoted by F_δ); and an activity i is *eligible* at δ (set E_δ) if all its predecessors have been scheduled and finished but i has not been scheduled yet, i.e. $j \in S^{\mathfrak{p}} \cap F_\delta$ for all $j \in P_i$ and $i \notin S^{\mathfrak{p}}$, i.e. if it could be started at δ. An activity is *newly eligible* if it has become eligible just at δ, i.e. if $f(j) = t$ for at least one $j \in P_i$ (corresponding set denoted by E^{nw}_δ). Further we define $el(i) = max\{f(j) | j \in P_i\}$, i.e. the time at which an activity becomes eligible.

3.2.2 Feasibility conditions for schedules

Beyond the basic data above, a schedule must satisfy several feasibility conditions the first three of which correspond to those for the RCPSP:

Precedence: An activity can be started only if all its predecessors have been finished, i.e. $f(i) \leq s(j)$ for all activities i, j with $i \in P_j$.

Duration: The difference between start and finish times must correspond to activity duration, i.e. $f(i) - s(i) = d_i$ for fixed activities $i \in A^{fx}$. For flexible activities the duration is $\tilde{d}_{\tau(i)}$ in case of repair, i.e. if a defective component is processed at the corresponding position ($\sigma(i) = 1$) and 0 for an intact component ($\sigma(i) = 0$), whence $f(i) - s(i) = \sigma(i) \cdot \tilde{d}_{\tau(i)}$ has to be satisfied for each $i \in A^{fl}$.

Resources: For each time slot $[t, t+1] = [t]$, the resources consumed by fixed and flexible activities which are active in that slot must not exceed the capacities, i.e. $\sum_{i \in A_{[t]}} r'_{i\rho} \leq R_\rho$ where $r'_{i\rho} = r_{i\rho}$ for fixed and $r'_{i\rho} = r_{\tau(i)\rho}$ for flexible activities i.
(Note that a flexible activity which is active in $[t]$ has non-zero duration whence it is processed as repair task and consumes resources according to its type.)

Whereas the previous conditions are well-known and only adapted to the new setting, the following last condition is specific for our scheduling problem. In order to state it, we must formally define the

counter functions which were already, albeit more informally, introduced in the second example above. As already seen there, a component counter function logs how many components of given type and state are available at some point. More precisely, for given time t and combination (τ, σ), the *availability counter* counts the components of type τ and operational state σ which are available at t, just after new components have become accessible and just before new activities have been started and thereby consuming components. On the other hand, the *component counter* counts the components just after new components have become accessible and just after new activities have been started. Formally, given a schedule $\mathfrak{p}$, these counters can be inductively defined for each combination (τ, σ) as follows:

- $ac^{\mathfrak{p}}_{-1}(\tau, \sigma) = cc^{\mathfrak{p}}_{-1}(\tau, \sigma) = 0$
  (initial condition)
- $ac^{\mathfrak{p}}_{t+1}(\tau, \sigma) = cc^{\mathfrak{p}}_{t}(\tau, \sigma) + |\{i \in El^{nw}_{t+1} \cap A_\tau | \sigma_0(i) = \sigma\}|$
  (the number of (τ, σ)-components available after t and hence up to just before $t+1$, is increased by the number of (τ, σ)-components which become newly available at $t+1$, i.e. the number of activities becoming newly eligible at $t+1$)
- $cc^{\mathfrak{p}}_{t+1}(\tau, \sigma) = ac^{\mathfrak{p}}_{t+1}(\tau, \sigma) - |\{i \in A_\tau | s(i) = t+1\,,\, \sigma(i) = \sigma\}|$
  (the number of components available at $t+1$ is decreased by the number which become unavailable at $t+1$, i.e. which are consumed by activities starting at $t+1$)

From this follows that the number of components available at t is the number of components having become available up to t minus the number of components having been consumed up to t, i.e.

- $ac^{\mathfrak{p}}_{t}(\tau, \sigma) = |\{i \in A_\tau | el(i) \leq t\,,\, \sigma_0(i) = \sigma\}| - |\{i \in A_\tau | s(i) < t\,,\, \sigma(i) = \sigma\}|$
- $cc^{\mathfrak{p}}_{t}(\tau, \sigma) = |\{i \in A_\tau | el(i) \leq t\,,\, \sigma_0(i) = \sigma\}| - |\{i \in A_\tau | s(i) \leq t, \sigma(i) = \sigma\}|$

(the only difference is $\leq$ instead of $<$ in the second term for $cc_t(\tau, \sigma)$ since, compared with ac_t, cc_t is additionally reduced by components consumed by activities starting just at t)

Whereas the first, inductive representation will be used in the MIP model, the second will be convenient for proofs. Having defined counter functions one can state insertion feasibility as follows:

Insertion: For all t and combinations (τ, σ), the number of components consumed up to t must not be greater than the number of components which have become available up to t whence the net value of available components must not be negative. This can be expressed using the component counter function just defined by $cc_t(\tau, \sigma) \geq 0$ for all t and combinations (τ, σ) (ac is not needed here).

3.2.3 Insertion feasibility via auxiliary time steps

However, a closer look reveals that this condition alone is not yet completely sufficient to ensure in any case schedules with exchanges as intended. Indeed, difficulties can stem from the assumption that intact components are associated with tasks of duration of exactly zero since this can lead to the situation that events which logically follow one another may fall onto the same point in time whence the actual order of events may be hidden. As a consequence, it may also happen that a sequence of events which is logically inconsistent is not detected and excluded by the condition such that a schedule satisfying all conditions (as given above) is in fact not feasible as intended. This can really happen as the following example will show (again without resources):

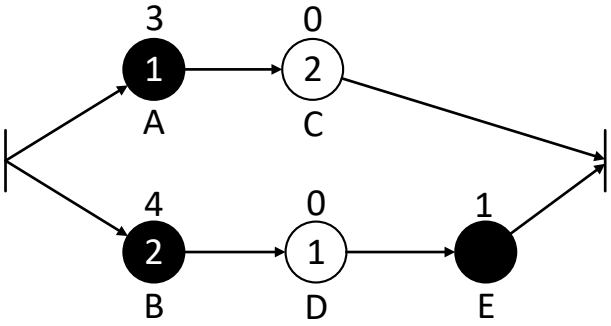


Figure 7: Example for crosswise exchange

Here activities A and D are flexible of type 1, B and C of type 2. Positions A and B contain initially defective components, C and D intact components. Repair tasks of type 1 and 2 have duration 3 and 4, respectively, i.e. $\tilde{d}_1 = 3$ and $\tilde{d}_2 = 4$. E is a fixed activity of duration 1. Given this instance, the following schedule is feasible w.r.t. the conditions above:

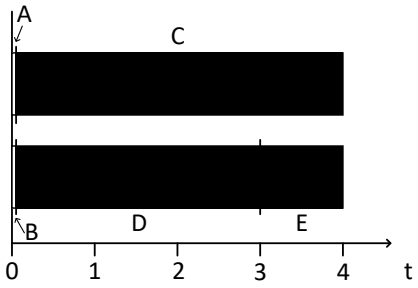


Figure 8: Schedule with crosswise exchange

i.e. the schedule defined by

$s(A) = f(A) = s(B) = f(B) = s(C) = f(D) = 0$, $f(D) = 3$, $f(C) = f(E) = 4$ and
$\sigma(A) = \sigma(B) = 0$, $\sigma(C) = \sigma(D) = 1$.

Feasibility w.r.t. precedence and duration is immediately checked, the resource condition is not relevant since there are no resource restrictions. Finally, the insertion condition is satisfied as all flexible activities become eligible at $t = 0$, whence for each combination $(\tau, \sigma) \in \{1, 2\} \times \{0, 1\}$ a component becomes available at that point of time, i.e. $ac_0(\tau, \sigma) = 1$ for all combinations. Still at $t = 0$ each of these components is also consumed by starting all flexible activities, whence $cc_t(\tau, \sigma) = 0$. Thus each component becomes available at $t = 0$ and is immediately consumed again. Altogether we get a schedule with duration 4, which is feasible w.r.t. to the conditions above.
However, this schedule must be considered to be actually infeasible for the following reason: The exchange between A and D at 0 requires that B has been finished at 0 and hence that the exchange between B and C has already happened; but this, in turn, requires that the exchange A and B has happened before. For short, the exchange between A and D must have been performed before that between B and C - and vice versa. Thus the crosswise exchange causes the problem and leads to an logically impossible circular situation which is not detected since in fact these exchanges happen at the same time at $t = 0$; more precisely, since "before" can be done immediately, i.e. 0 time units before and hence at the same point of time whence the logical precedence is not reflected by a temporal one. In some sense, the logical inconsistency is contracted to a single point of time and becomes invisible for the condition. - Thus, at most one exchange is possible and in fact it is not worse (and, if the additional effort would be counted, even better) here to do without any exchange, in which case only

a project duration of 5 can be achieved instead of 4.
Alternatively, one could also simply argue that no intact component becomes accessible before $t = 3$ whence no exchange is possible before and no dummy task can be processed already at $t = 0$. However, this argument gives no further insight as to how that situation arises at all and hence what to do in order to avoid such behaviour.
Altogether, normal time units are not fine-grained enough to state the insertion condition properly. Accordingly, this undesirable behaviour can be overcome by virtually assigning small durations to dummy tasks and thereby avoiding that their contribution has any further, unintended effect on the actual project duration. For this, we will introduce auxiliary marginal time steps such that each normal time unit is subdivided by a constant, sufficiently large number M of such marginal time steps. Then each task of actual duration 0 gets a duration of 1 marginal time unit whereas each duration $d > 0$ is translated to dM marginal time units. Since each dummy task causes a small delay by this translation, one has to make sure that the cumulated delays do not have any effect on the regular time units, i.e. they must not sum up to a common time unit. For this, M has to be chosen sufficiently large ($M := n + 1$ would suffice, but usually smaller values will do as well).
This leads to an equivalent instance with durations simply measured in marginal time steps. Correspondingly, one gets extended schedules $\mathfrak{p} = (S, \underline{s}, \underline{f}, \sigma)$ where $\underline{s}$, $\underline{f}$ are functions $A \to \mathbb{N}_0 \times \{0, \ldots, M-1\}$, i.e. start and end times have values (t_1, t_2) with $t_1 \in \mathbb{N}_0$ and $t_2 \in \{0, \ldots, M-1\}$. Each such value (t_1, t_2) can be identified with $t_1 M + t_2$, the time counted in marginal time steps whence we get a schedule with start and end times in $\mathbb{N}_0$ again, for which the feasibility condition is directly applicable. And this we also do: A schedule $\mathfrak{p} = (S, \underline{s}, \underline{f}, \sigma)$ is feasible if its translation with values in $\mathbb{N}_0$ satisfies the four feasibility conditions introduced before.
Altogether, an instance $\mathcal{E}$ is translated to an instance $\tilde{\mathcal{E}}$ by multiplying all durations by M. A schedule $\tilde{\mathfrak{p}}$ is feasible for this translated instance $\tilde{\mathcal{E}}$ if it satisfies the four conditions above and assigns duration 1 to dummy tasks. Projection to standard time units, i.e. forgetting the marginal part, gives a feasible schedule for the original instance which is really feasible as intended since proper sequence of events is guaranteed by the extended schedule $\tilde{\mathfrak{p}}$.
Thus feasibility of a normal schedule $\mathfrak{p}$ is defined via the existence of a feasible extension $\tilde{\mathfrak{p}}$; only if a feasibile extension exists, a schedule is feasible. Conversely, a schedule for which no such feasible extension exists is not considered feasible even if it satisfies all conditions, the insertion condition directly applied to normal time units only (as in the example above). However, the marginal time steps have only an auxiliary function to avoid anomalies as seen above, and at the end we are interested only in the real times measured in normal time units.
Thus having specified the feasibility of schedules, the objective of the AEPSP is to find a feasible complete schedule with minimal duration, i.e. determining the minimal makespan:

Objective: $min\{f^{\mathfrak{p}}(n+1) | \mathfrak{p}$ is a complete and feasible schedule$\}$

Thus having stated the objective completes the formal definition of the AEPSP. A formulation as mixed-integer program (MIP) can be found in the Appendix (8.1).

4 A Branch-and-Bound approach for the AEPSP

4.1 Motivation: Branching with delaying alternatives for the RCPSP

Having defined the decision problem formally, next we present a solution procedure for it. The solution approach introduced and applied in the following is an extension of the Branch-and-Bound (in the following short "B&B") procedure for the RCPSP based on so-called delaying alternatives. This approach has been introduced by [Demeulemeester and Herroelen 1992] and has turned out one of the most efficient branching schemes for the RCPSP [Artigues et al. 2008]. Correspondingly,

we start with a short review of this branching scheme, illustrating it by the following RCPSP instance:

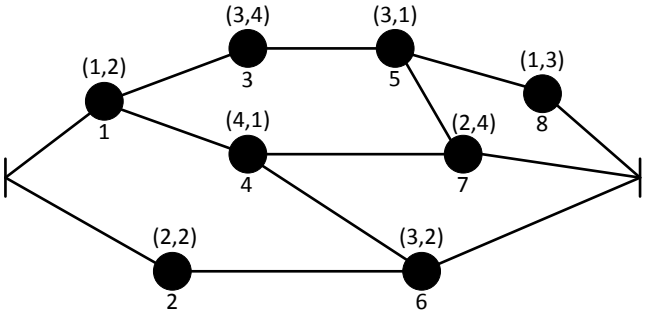


Figure 9: RCPSP instance

Here we assume one resource type with capacity $R_1 = 5$. For each node, the value below indicates the activity number, the first of the values above represents the duration, the second resource consumption.

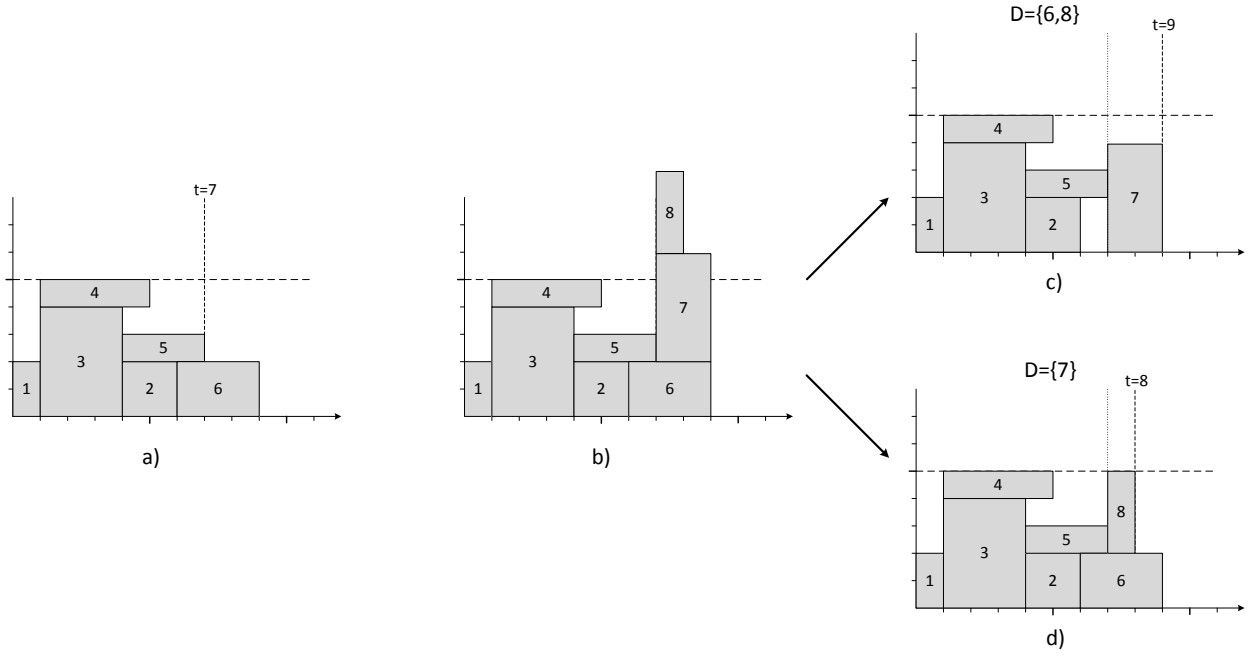


Figure 10: Branching with delaying alternatives

The nodes in the B&B tree correspond to decision points $\delta = (\mathfrak{p}, t)$ consisting of a feasible partial schedule and a point in time, both resulting from decisions made before, along the path through the B&B-tree which leads to that node. Arrived at such a decision point (a), the idea is to schedule at first all eligible activities (here: 7,8), starting them at the current time. In general, this will lead to a resource conflict and hence to an infeasible schedule (b). This infeasibility can be resolved by delaying some of the activities contributing to the capacity excess. This concerns not only the eligible

activities just scheduled (namely 7,8) but also activities scheduled earlier but still running (here: 6). The schedule of the delayed activities, i.e. their start and end times, are cancelled and one gets a new feasible schedule. Next time point is given by the earliest end time among the activities which have not finished yet. Both schedule and time point form the next decision point and hence a new B&B node. Above each of the delaying sets $\{6, 8\}$ and $\{7\}$ leads to a successor in the B&B-tree. It can be shown that it suffices to regard only (inclusion-)minimal delaying sets to find an optimal solution whence delaying sets $\{6, 7\}$ and $\{7, 8\}$ need not to be taken into account since they already contain $\{7\}$. If there is no infeasibility, the delaying set is empty and the new feasible decision point is the only successor of the original branch-and-bound node. In this way finally a complete schedule is reached which corresponds to a leaf of the B&B tree.
Thus, altogether, the first step consists of starting all eligible activities, the second step of delaying a minimal set of activities in order to resolve a possible resource conflict (if there is none, no activity needs to be delayed). Here it is important, that each still running activity can be delayed, even if it has been started earlier. Thus the schedule of an activity remains possibly preliminary up to the point when its end time has been reached.

4.2 Minimal alternatives for the AEPSP

After this exemplary outline of the branching scheme with delaying alternatives for the RCPSP, we turn to the AEPSP again and show, now in detail, how to adapt the procedure sketched above to the new setting. The main differences will stem from the fact that we now have two sorts of activities which must be treated differently: Whereas fixed activities at some decision point can be treated in the same way as for the RCPSP (decision on delay or not delay), for flexible activities delay and insertion decisions must be made simultaneously. Accordingly, this will make the situation more complicated.
As for the RCPSP, a node in the B&B-tree corresponds to a decision point $\delta = (\mathfrak{p}, t)$. The set of activities on which decisions will be made at the current decision point δ is called *decision set* here, denoted by Dc_δ. It consists of all eligible activities and the active fixed activities, i.e. $Dc_\delta = E_\delta \cup A_\delta^{fx}$. Note that active flexible activities are not contained: In contrast to their fixed counterparts, they will not change their schedule as soon as they have been scheduled once.
For each activity $i \in Dc_\delta$ precisely one of the following decisions has to be made, depending on the set it belongs to:

- If i is fixed and active ($i \in A_\delta^{fx}$) then it can be delayed or it is kept scheduled as before (2 possibilities: d, s).
- If i is fixed and eligible ($i \in E_\delta^{fx}$) then it can be delayed or scheduled, starting at t (2 possibilities: d, s).
- If i is flexible ($i \in E_\delta^{fl}$) then it can be either delayed without assignment of a state or scheduled with state 0 or scheduled with state 1 (3 possibilities: $d, 0, 1$).

A combination $\bar{c} \in \{d, s\}^{Dc_\delta^{fx}} \times \{d, 0, 1\}^{Dc_\delta^{fl}}$ of one of those decisions for each activity $i \in Dc_\delta$ can be uniquely represented by a pair of sets (Dl, St^0) where $Dl \subseteq Dc_\delta$ is the set of activities to be delayed (fixed or flexible) and St^0 the set of flexible activities to be scheduled with state 0 (for further use, let St^1 be the corresponding set for state 1). Conversely, each such pair of sets with $Dl \subseteq Dc_\delta$, $St^0 \subseteq Dc_\delta^{fl} \backslash Dl$ forms a possible alternative from which the decision made for each $i \in Dc_\delta$ can be uniquely reconstructed.
The decisions associated with an alternative (Dl, St^0) lead to a new decision point $\delta' = (\mathfrak{p}', t')$ as follows:

- $S^{\mathfrak{p}'} = S^{\mathfrak{p}} \cup Dc_\delta \backslash Dl$

- $s(i) = t$ for each $i \in E_\delta \backslash Dl$ (for other $i \in S^{\mathfrak{p}'}$ unchanged)
- finish times given by

$$f(i) = \begin{cases} t + d_i & for\ i \in E_\delta^{fx} \backslash Dl \\ t & for\ i \in St^0 \\ t + \tilde{d}_{\tau(i)} & for\ i \in St^1 = Dc_\delta^{fl} \backslash (Dl \cup St^0) \end{cases}$$

  (for other $i \in S^{\mathfrak{p}'}$ unchanged)
- $t' = min\{f(i) | (i \in S^{\mathfrak{p}'} \cap S^{\mathfrak{p}} \wedge f(i) > t) \vee i \in S^{\mathfrak{p}'} \backslash S^{\mathfrak{p}}\}$

Such a schedule can be infeasible due to resource and insertion conditions (whereas precedence and duration conditions are already satisfied by construction). Thus we exclude all alternatives which lead to an infeasible schedule and refer to the other ones as feasible alternatives. Accordingly, an alternative (Dl, St^0) is *feasible* if it satisfies

- $\sum_{i \in Dc_\delta^{fx} \backslash Dl^{fx}} r_{i\rho} + \sum_{i \in A_\delta^{fl}} \tilde{r}_{\tau(i)\rho} + \sum_{i \in St^1} \tilde{r}_{\tau(i)\rho} \leq R_\rho$ for each resource $\rho \in \mathcal{R}$

  (resource-feasibility: for each resource type, the resource consumption of non-delayed fixed activities; earlier started, still running flexible activities; and flexible activities just started in state 1 must not exceed the resource capacity; i.e. the resource consumption in the time slot following t must not be exceed resource capacity)
- $|St^\sigma \cap A_\tau| \leq ac_\delta(\tau, \sigma)$ for each type $\tau \in \mathcal{T}$ and state $\sigma \in \{0, 1\}$

  (insertion-feasibility: for each combination (τ, σ), the number of (τ, σ)-components just inserted must not exceed the number which have been available at δ)

Further a feasible alternative (Dl_1, St_1^0) is *minimal* if there is no other feasible alternative (Dl_2, St_2^0) which agrees with (Dl_1, St_1^0) on the decisions on flexible activities and delays a proper subset of the fixed activities, i.e. $Dl_2^{fl} = Dl_1^{fl}$, $St_2^0 = St_1^0$, and $Dl_2^{fx} \subsetneq Dl_1^{fx}$. Given a decision point δ, we denote the set of minimal alternatives by $\mathcal{MA}_\delta$.
For the AEPSP, these minimal alternatives will be the equivalent to the delaying alternatives for the RCPSP, i.e. for a B&B-node given by a decision point δ, each minimal alternative $(Dl, St^0) \in \mathcal{MA}_\delta$ corresponds to a successor of δ in the B&B-tree. Correspondingly, in the same way as a branching tree using delaying alternatives contains an optimal solution for an RCPSP instance, it suffices to regard minimal alternatives (they are indeed "complete"), i.e. that an optimal solution can be found by using branching with them:

Theorem 1. *Let $\mathcal{E}$ be an instance of the AEPSP. Then the search tree using branching with minimal alternatives contains a node corresponding to a an optimal schedule.*

The proof is somewhat technical and can be found in the Appendix (8.2).

Correspondingly, the generation of $\mathcal{MA}_\delta$ will be a subroutine called for each B&B-node δ, and to generate minimal alternatives in a systematical and efficient way we will use so called "numerical combinations". Given the different decisions to be made for fixed and flexible activities respectively, the idea is to generate the decisions for both classes of activities as separately as possible at first and then combine them to complete decisions, i.e. minimal alternatives. However, the decisions cannot be made completely independently since decisions for fixed and flexible activities must fit together to form a minimal alternative indeed (i.e. which is feasible and minimal). The link between both will be what we call "numerical combinations": Given the current decision point δ and its decision set Dc_δ, a *numerical combination* is a tuple $\overline{nc} = (nc_\tau)_{\tau \in \mathcal{T}} \in \{0, \ldots, |Dc_\delta \cap A_1|\} \times ... \times \{0, \ldots, |Dc_\delta \cap A_{\bar{\tau}}|\}$ which

will indicate how many flexible activities of some type are to be scheduled with state 1 at the current decision point, i.e. it reflects the intended cardinalities $|St^1 \cap A_\tau|$ for each type τ. For feasibility, a numerical combination must satisfy insertion and resource conditions which are implicitly required for each $St^1 \cap A_\tau$ as well, i.e.

- $nc_\tau \leq ac_\delta(\tau, 1)$ for each $\tau \in \mathcal{T}$
  (start at δ no more activities of type τ in state 1 than available $(\tau, 1)$-components) and
- $\sum_{\tau=1}^{\bar{\tau}} nc_\tau \cdot \tilde{r}_{\tau\rho} \leq R_\rho - \sum_{\tau=1}^{\bar{\tau}} |A_\delta \cap A_\tau| \cdot \tilde{r}_{\tau\rho}$ for each $\tau \in \mathcal{T}$
  (from $\sum_{\tau=1}^{\bar{\tau}} (nc_\tau + |A_\delta \cap A_\tau|) \cdot \tilde{r}_{\tau\rho} \leq R_\rho$: resource consumption of flexible activities, which either start at δ in state 1 or which are still active in state 1 at δ, must not exceed resource capacity).

Choosing a feasible numerical combination, i.e. choosing a feasible sequence of values for $(|St^1 \cap A_\tau|)_{\tau \in \mathcal{T}}$ is the first decision made towards the complete decision of a minimal alternative (Dl, St^0). In order to get such a one, one has to make further decisions for fixed and flexible activities, respectively. For flexible activities one has to choose a subset $St^1 \subseteq Dc_\delta^{fl}$ with $|St^1 \cap A_\tau| = nc_\tau$ for each $\tau \in \mathcal{T}$. For the remaining $i \in Dc_\delta \backslash St^1$ one has to decide if i is either started in state 0 ($i \in St^0$) or if it is delayed ($i \in Dl$). For feasibility one has also to guarantee $|St^0 \cap A_\tau| \leq ac_\delta(\tau, 0)$.
As far as fixed activities are concerned, a numerical combination $\overline{nc}$ determines the resources left for those, since the residual capacity is $R^{res}_{\overline{nc}\rho} := R_\rho - \sum_{\tau=1}^{\bar{\tau}} (nc_\tau + |Ac_\delta \cap A_\tau|) \cdot \tilde{r}_{\tau\rho}$. Now all fixed activities in Dc_δ^{fx} are virtually started at δ and then all minimal subsets Dl_δ^{fx} are determined which have to be delayed in order to avoid that the residual capacity $R^{res}_{\overline{nc}\rho}$ is exceeded.
For given numerical combination $\overline{nc}$, the sets generated for fixed activities can be combined with those for flexible activities, and each of this pairwise combinations corresponds to a minimal alternative associated with $\overline{nc}$, leading to a set $\mathcal{MA}_{\overline{nc}}$ containing all these minimal alternatives. Finally the union of these sets is taken, resulting in the set of all minimal alternatives for δ, $\mathcal{MA}_\delta := \bigcup_{\overline{nc}} \mathcal{MA}_{\overline{nc}}$.

4.3 Dominance rule with cut vectors

In order to get an applicable, i.e. sufficiently efficient procedure, the introduced branching rule must be supplemented by some method which reduces the otherwise exponentially growing search tree (or, rather, to slow down the generally unavoidable exponential growth). In the corresponding case of RCPSP a number of lower bounds and dominance rules have been presented and applied [Klein 2000, Brucker and Knust 2006]. For the AEPSP, we will propose and apply a dominance rule extending the so-called "cutset rule" for the RCPSP introduced by [Demeulemeester and Herroelen 1992], adapted to the exchange setting.
For a decision point $\delta = (\mathfrak{p}, t)$, the cut information consists of the pair $(S^{\mathfrak{p}}, f^{\geq t})$ where $S^{\mathfrak{p}}$ is as usual the set of scheduled activities and $f^{\geq t}$ with $f^{\geq t}(i) := max\{f(i), t\}$ the function assigning to each $i \in S^{\mathfrak{p}}$ its end time if it ends after t and t else. Then, given two decision points δ and δ', δ dominates δ' according to the cutset rule if $S^{\mathfrak{p}} \supseteq S^{\mathfrak{p}'}$ and $f^{\geq t}(i) \leq f^{\geq t'}(i)$ for all $i \in S^{\mathfrak{p}'}$. In other words, comparing two decision points, it is advantageous standing at an earlier time, having scheduled more (more precisely: a superset of) activities and finishing them earlier; and further, as indicated by the use of $f^{\geq t}$, the particular scheduling decision for any activity that ends prior to t can be ignored (figuratively spoken, a cut is made at the current decision point by which earlier details of the schedule are simply forgotten). Here dominating and "advantageous" means that the corresponding decision point has a continuation (a complete schedule extending the current partial schedule) the makespan of which is the same or lower than any continuation of the other decision point.
Adapted to our setting, instead of cut sets and end times we will use "cut vectors" defined as follows: For a decision point $\delta = (\mathfrak{p}, t)$ define its *cut vector* as $cv_\delta := (t, S^{\mathfrak{p}}, f^{\geq t}, (n_\tau^0)_{\tau \in \mathcal{T}})$ where

- $S^{\mathfrak{p}}$ is (as usual) the set of activities scheduled in $\mathfrak{p}$

- $f^{\geq t} : S^{\mathfrak{p}} \to \mathbb{N}_0$ with $f^{\geq t}(i) := max\{f^{\mathfrak{p}}(i), t\}$
- $n^0_\tau := |\{i \in S^{\mathfrak{p}} \cap A^{fl} | \sigma^{\mathfrak{p}}(i) = 0\}|$ for each $\tau \in \mathcal{T}$
  (i.e. the number of intact components consumed to realize that schedule)

While the first two informations are as for the RCPSP, the third, the number of already consumed intact components, is specific for the exchange setting. Cut vectors describe a situation by which decision points can be compared w.r.t. their extendability, and which one is better in this case is captured by the following notion of "dominance":
Given two cut vectors $cv = (t, S, f^{\geq t}, (n^0_\tau)_{\tau\in\mathcal{T}})$ and $\tilde{cv} = (\tilde{t}, \tilde{S}, \tilde{f}^{\geq t}, (\tilde{n}^0_\tau)_{\tau\in\mathcal{T}})$, then *cv dominates* $\tilde{cv}$ if

- $t \leq \tilde{t}$
- $S^{fl} = \tilde{S}^{fl}$
- $S^{fx} \supseteq \tilde{S}^{fx}$
- $f^{\geq t}(i) \leq \tilde{f}^{\geq \tilde{t}}(i)$ for each $i \in S$
- $n^0_\tau \leq \tilde{n}^0_\tau$ for each $\tau \in \mathcal{T}$

As before, it is advantageous to stand at an earlier time, having scheduled more activities and finishing still running activities (i.e. ending after t) earlier; additionally it is also advantageous having consumed fewer intact components to reach this. Requiring equality for the scheduled flexible activities may be at first somewhat counter-intuitive and unnecessarily restricting. However, it is not always better having scheduled more flexible activities (more precisely, a strict superset of such ones): The reason for this is that in our branching rule a flexible activity, scheduled once, cannot be delayed and hence will not change its schedule (whereas fixed activities can be still delayed); therefore a flexible activity scheduled with state 1 (as repair task) which is still running may prevent starting some activities according to an optimal solution due to resource constraints, in contrast to a decision point where this activity has not been prematurely scheduled yet.
The following assertion captures the intuition and formally states that cut vectors and the dominance relation can be used for a dominance rule for the branching with minimal alternatives indeed:

Theorem 2. *Let $\delta = (\mathfrak{p}, t)$ and $\tilde{\delta} = (\tilde{\mathfrak{p}}, \tilde{t})$ be decision points with cut vectors cv and $\tilde{cv}$ such that cv dominates $\tilde{cv}$. Further let $\Delta = (\mathfrak{q}, MS(\mathfrak{q}))$ be a decision point corresponding to a complete schedule with minimal makespan among those which are (direct or indirect) successors of δ in the branching tree with minimal alternatives. Let $\tilde{\Delta} = (\tilde{\mathfrak{q}}, MS(\tilde{\mathfrak{q}}))$ be a corresponding decision point for $\tilde{\delta}$, respectively. Then $MS(\mathfrak{q}) \leq MS(\tilde{\mathfrak{q}})$.*

Proof. see Appendix (8.3) □

As a consequence, being at $\tilde{\delta}$ in the search tree and having found a stored cut vector cv of a node visited earlier which dominates the current cut vector $\tilde{cv}$, an optimal successor of $\tilde{\delta}$ is not better than that of the node δ from which cv has been derived. Correspondingly, if the sub-tree rooted in δ has been already searched, that one rooted in $\tilde{\delta}$ need not to be searched anymore but can be pruned.

The branching scheme using minimal alternatives together with the dominance rule using cut vectors form the Branch-and-Bound approach which has been applied to solve the AEPSP. The results are presented in the following.

5 Computational study

5.1 Experimental setting

5.1.1 Instance generation

To test and demonstrate the performance of the proposed Branch-and-Bound solution approach, we carried out a comprehensive computational study. This included the comparison with a standard solver (GAMS/CPLEX) applied to the the mixed integer program; further a thorough analysis how computation time is influenced by instance parameter settings; and finally a closer investigation of the distribution of computation times around the mean for fixed parameter combinations.
For this we generated problem instances derived from RCPSP benchmark instances from the PSPLIB (Project Scheduling Problem Library) [Kolisch and Sprecher 1996, online available under www.om-db.wi.tum.de/psplib/] , a widely used test instance library for that problem. The PSPLIB contains test instances for the RCPSP and some variants (like multi-mode). For standard RCPSP, it contains instance sets for 30, 60, 90 and 120 (non-dummy) activities, each of which consists of several hundred instances. In spite of much research of more than two decades, up to now only the 30-activity set (J30) is completely closed, i.e. all its instances have been solved to optimality. For the 60-activity set (J60), currently 28 out of 480 instances are still open [www.om-db.wi.tum.de/psplib/files/j60lb.sm], which underlines the difficulty of the RCPSP and its occasionally uttered estimation as one of "really hard optimization problems" [Koné et al. 2011] which has attracted many researchers, as already seen in the literature review. Due to this difficulty and expecting that incorporating the exchange option will further increase the practical complexity (even if remaining strongly NP-complete as the RCPSP), we restricted our study to AEPSP instances with 30 activities derived from that RCPSP set J30.
This set J30 consists of 480 RCPSP instances generated according to a full factorial design by varying the following three parameters:

- *Network Complexity (NC):* Network complexity is the average number of non-redundant precedence relations per node, or, alternatively, the average number of (direct) successors (if there are no redundant arcs), i.e. $NC = |P|/(n+2) = (\sum_{i=0}^{n+1} |S_i|)/(n+2)$.

- *Resource Factor (RF):* The resource factor is the average number of resources (more precisely: resource types) which are consumed by an activity related to the number of all available resources (or, alternatively, the average proportion of resources consumed by each activity), i.e. $RF = \sum_{i=1}^{n} |\{\rho \in \mathcal{R} | r_{i\rho} > 0\}|/(n\bar{\rho})$. For example, if 4 resource types are available, from which, on average, each activity actually consumes 3 resource types, then $RF = 0.75$. Thus the resource factor still does not reflect to which extent the resources used by an activity are actually consumed, i.e. whether, on average, an activity needs 10 or 50% of the resource capacity for each resource it actually consumes; this is captured by the following parameter:

- *Resource Strength (RS):* The resource strength measures resource consumption by an activities in relation to the available resource capacity. The precise definition is somewhat technical since it also takes into account how many activities are scheduled parallel at given time in a resource-feasible earliest start schedule, see [Kolisch et al. 1995] for details. It takes values between 0 and 1, where the latter value indicates maximal consumption of the resources which are actually consumed by activities.

Given these parameters, for the full-factorial design the following values were considered:

$NC \in \{1.5, 1.8, 2.1\}$
$RF \in \{0.25, 0.5, 0.75, 1\}$
$RS \in \{0.2, 0.5, 0.7, 1\}$

This yields $3 \cdot 4 \cdot 4 = 48$ parameter combinations, for each of which 10 instances are contained in the set (J30), hence resulting in just those 480 RCPSP instances mentioned above, which are also the starting point from which instances for our problem have been derived.

To get a (non-trivial) AEPSP instance from an RCPSP instance, one has to specify flexible activities of different types. For this we we have modified RCPSP instances according to so-called "type combinations". Here a *type combination* is given by a double sequence $b_1 \dots b_{\bar{\tau}}/a_1 \dots a_{\bar{\tau}}$ specifying, for each type τ, the number of positions a_τ (flexible activities) and, among these, the number b_τ of positions with initial state 1 (initially containing a defective component), i.e. $a_\tau = |A_\tau|$ and $b_\tau = |\{i \in A_\tau | \sigma_0(i) = 1\}|$. We will regard the following five type combinations, each for 5 types:

11111/22222, 11111/33333, 11111/44444, 11122/22333, 11212/23344

Note that since $\sum_{\tau=1}^{5} a_\tau \leq 30$, all these type combinations are feasible for instances with $n = 30$ non-dummy activities.
Now, given an RCPSP instance and a type combination $\bar{b}/\bar{a}$, an AEPSP instance is generated by randomly choosing sets according to the type combination: For this, choose disjoint sets $A_\tau \subseteq \{1, \dots n\}$ with $|A_\tau| = a_\tau$; for each of these these choose subsets $B_\tau \subseteq A_\tau$ with $|B_\tau| = b_\tau$. Finally an activity $i_\tau \in B_\tau$ is chosen to determine the remaining type-specific characteristics, i.e. duration and resource consumptions for a repair task of type τ, by setting $\tilde{d}_\tau := d'_{i_\tau}$ and $\tilde{r}_{\tau\rho} := r'_{i_\tau \rho}$ where the $'$ed values are those from the underlying RCPSP instance. Individual durations and resource consumption values of other activities chosen to be flexible are simply omitted. Altogether we get a fully specified AEPSP instance.
In this way for each pair of RCPSP instance and type combination one AEPSP instance is generated, which yields $480 \cdot 5 = 2400$ AEPSP instances representing $48 \cdot 5 = 240$ parameter combinations for four parameters (we will treat "type combination" as a single parameter which can attain the five "values" given above). Thus each parameter combination is represented by 10 instances.
For better readability of the figures below, we will uniformly use the following order for the parameter combinations: Primary they are ordered according to type combination, followed by NC, RF and finally RS. Accordingly, the 240 parameter combinations are numbered such that
- type combination is changed after 48 (according to the order above)
- NC is increased after 16
- RF is increased after 4
- RS is increased after 1 step(s).

Thus, for example, parameter combination $163 = 144 + 16 + 0 + 2 + 1 = (4-1) \cdot 48 + (2-1) \cdot 16 + (1-1) \cdot 4 + (3-1) \cdot 1 + 1$ corresponds to the 4^{th} type combination (11122/22333), 2^{nd} NC-value (NC=1.8), 1^{st} RF-value (RF=0.25) and the 3^{rd} RS-value (RS=0.7). Where results are averaged over type combinations, the remaining 48 $NC/RF/RS$-combinations are numbered correspondingly, i.e. like for a fixed type combination.

5.1.2 Realization of the study

To solve the generated instances, the presented Branch-and-Bound approach (in the following short "B&B") as well GAMS/CPLEX applied to the mixed integer programm (MIP) have been applied. The B&B approach has been implemented in C++, whereas on the other hand GAMS 24.6 with solver CPLEX 12.6 has been used for the MIP. Both solution approaches were run on a computer with Intel Xeon 3.1 GHz with 64 GB RAM. For our study a time limit of 3600 s has been chosen (which has been later increased for the time distribution study). Further a computation has been broken if 180000 cut vectors were reached; the corresponding instance has been counted as timeout (however, this latter case only concerns a negligible proportion of less than 1% (17 out of 2400 instances) which are finished for this reason.
Having described the experimental setting, the results are presented next.

5.2 Results

5.2.1 Comparison of B&B and MIP with aggregated data

Already a first look at the most basic performance measures shows that the B&B procedure outperforms GAMS/CPLEX applied to the MIP:

	average computation time (s)	instances solved	timeouts	timeouts for both	better than alternative
B&B	237.9	2324 (96.8%)	76 (3.2%)	64 (2.7%)	2272 (94.7%)
MIP	1846.2	1430 (59.6%)	970 (40.4%)	64 (2.7%)	64 (2.7%)

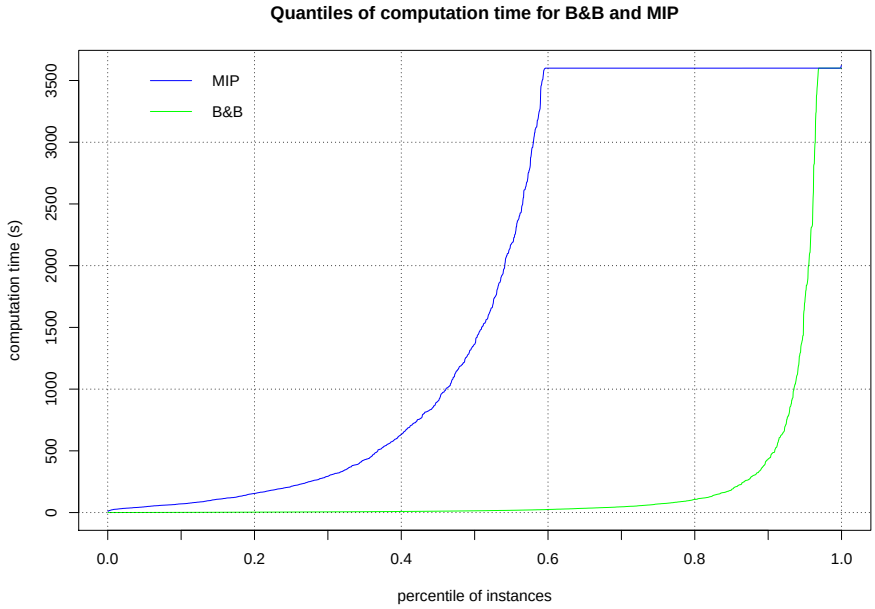
On average, B&B is about 8 times faster than MIP; for 95% of all instances, B&B needs less time compared with only 3% for MIP (remaining 3% with timeout for both solution approaches). Correspondingly, B&B could solve nearly all instances (except for 3%) within one hour, whereas about 40% remained unsolved by MIP. Given this high proportion of timeout instances for MIP, the actual discrepancy between computation times is even higher than suggested by the averages, which will be confirmed below.
These first observations are further corrobated by a more detailed view at the proportion of instances which were solved within given time bounds:

time (s)	1	10	60	300	1800	3600
B&B	7.3	42.9	73.4	87.9	95.1	96.8
MIP	0	0	7.8	30.2	53.0	59.6

Using B&B, nearly three of four instances have been solved within a minute whereas for MIP this ratio is not even reached after one hour. The complementary data are given in the following table, showing the percentiles:

percentile (%)	10	25	50	75	90	95
B&B	1.4	4.3	14.2	69.7	434.5	1704.4
MIP	70.6	212.7	1363.8	>3600	>3600	>3600

Note that the median (50%) of computation time for MIP is about 100 times larger than that for B&B, and also for other percentiles the ratio is at about 50 or more. The complete continuous picture is presented by the following graph:



The quantitative relationsship is still better visible in logarithmic scale (see Appendix 8.4.1). There both curves have similar shape (at least up to the time limit), but are distinguished by a roughly constant difference, which, since in logarithmic scale, corresponds to a quotient of at least one-and-a-half magnitudes of order. Closer inspection confirms that this quotient reaches a minimum of about 50 at [0.1,0.25] and is significantly increasing afterwards (Appendix 8.4.2).
Altogether these quantile consideration suggests (since largely dismissing the effect of the time limit, which systematically favours MIP due to its high proportion of timeouts) that the actual ratio of computation times is at least about 50 over a large range (instead "only" about 8 as seen above where the effect of time limit was still present). However, in the following we will see that this ratio also depends on the parameter combination (even if the results so far already imply that there cannot be more than very few, if any, parameter combinations where MIP is comparable to or even better than B&B). But since already these first results clearly suggest its better performance, we will focus more on B&B and complement the results with those for MIP where such a comparison is especially instructive.

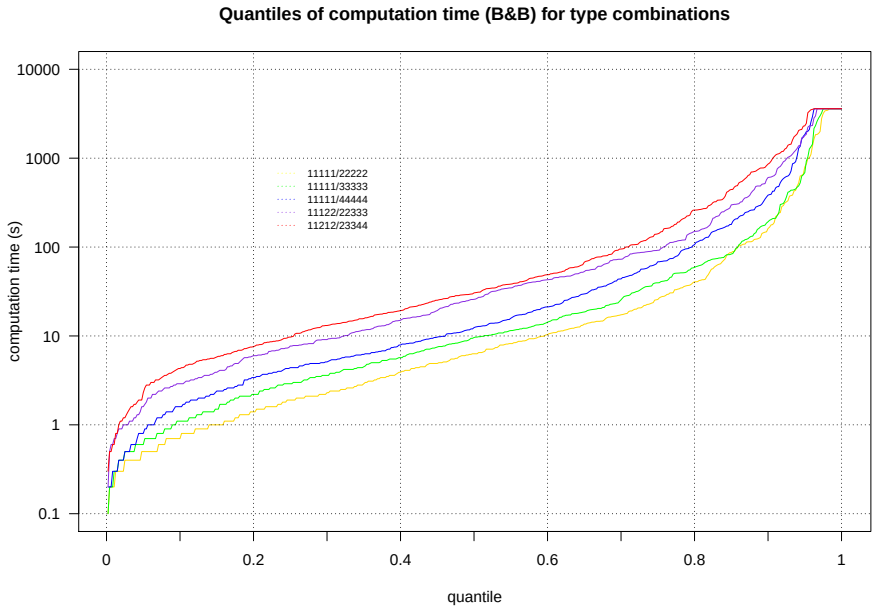
5.2.2 Influence of parameters settings on solution time

a) Fixing one parameter value

Clear dependencies of computation times on instance parameters become already visible by fixing one parameter value and taking the average over times for all instances having this value:

NC	1.5	1.8	2.1		
	584.5	*117.4*	*11.9*		
RF	0.25	0.5	0.75	1	
	85.1	*123.1*	*247.3*	*496.3*	
RS	0.2	0.5	0.7	1	
	516.0	*251.7*	*120.2*	*63.7*	
TC	11111/22222	11111/33333	11111/44444	11122/22333	11212/23344
	165.2	*182.0*	*241.9*	*269.7*	*330.8*

For example, the computation time for all (2400/3=800) instances with $NC = 1.5$ (whereas other parameters can be varied arbitrarily) is 584.5 s, that for all (2400/4=600) instances with $RS = 0.7$ is 120.2 s, etc.
Regarding the first three parameters, the ratio of computation times between the easiest and hardest parameter setting is about 50 for NC, 6 for RF and 8 for RS. Given the similar size of ranges, changing NC has a larger effect than RF and RS also if measured with same units: So it can be read out that increasing NC by 0.3 reduces average computation time by a factor 5-10 for the parameter range regarded. On the other hand, increasing RF as well as decreasing RS by (about) 0.25 each increases average computation time by a factor about 2 (except for the somewhat smaller step between $RF = 0.25$ and 0.5). This also suggests a more exponential dependency on these three parameters, at least for the regarded ranges; this will be also one reason why we will also work with logarithmated times later.
The dependency on the type combination is more moderate, at least for the regarded ones, but still significant. As expected, the difficulty increases with more positions and/or initially defective components, such that the used order of the type combinations actually represents the order w.r.t. to their hardness. The type combination can be also distinguished by their percentile curves:

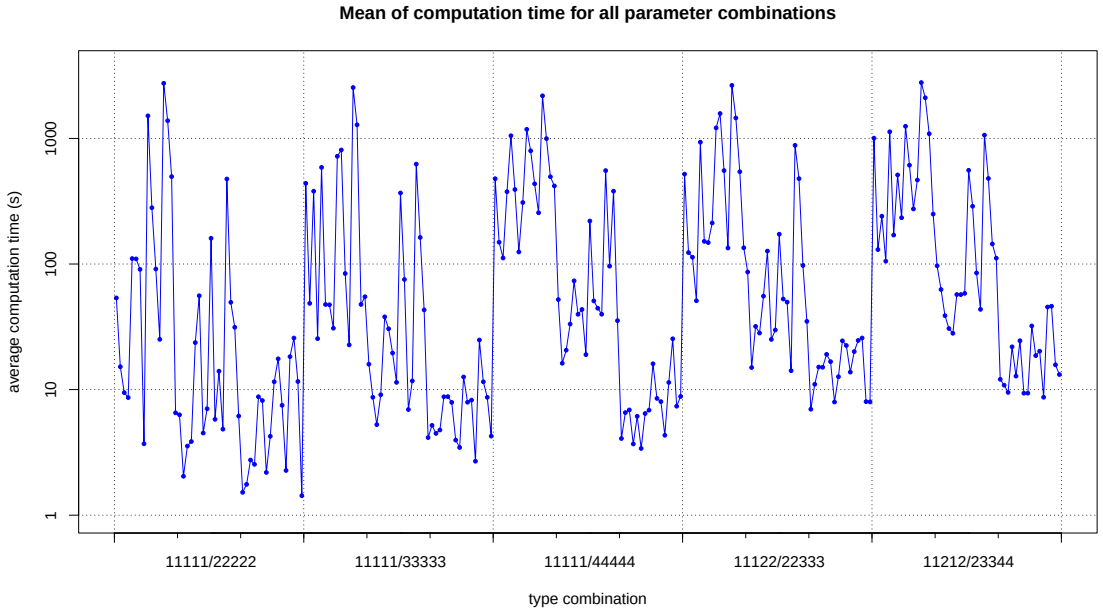


Here the ratio between the easiest and hardest type combination is 5-6 for the large part of the range such that the discrepancy of only about 2 as above may be partially an effect of the timeouts; on the other hand towards the ends the ratio actually becomes somewhat lower.

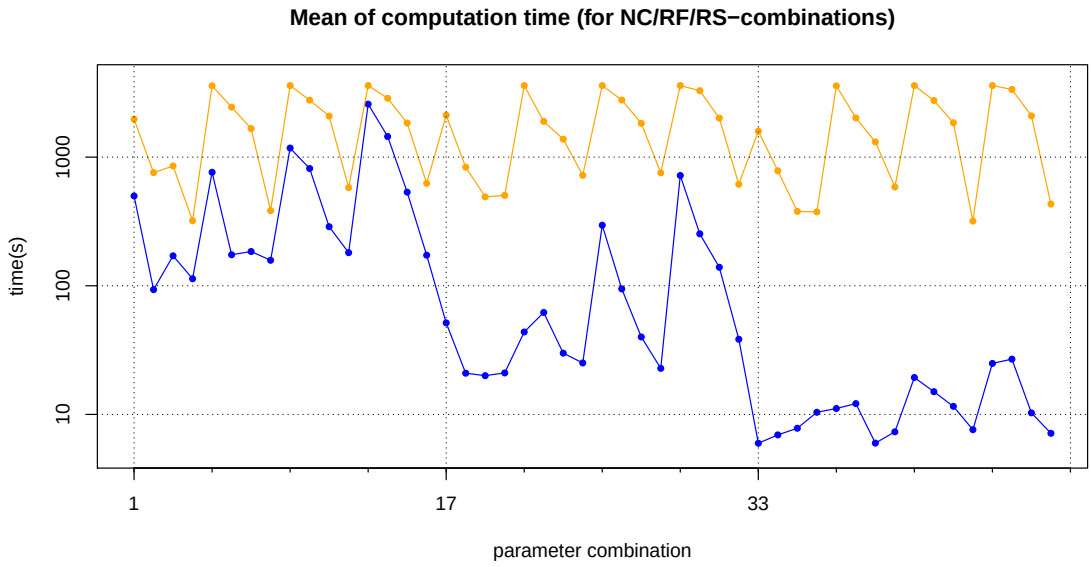
This gives a first picture which already shows some strong influences. In the following we will consider them in some more detail.

b) Regarding all parameter combinations

The average computation times for all 240 parameter combinations for the B&B approach are given by the following graph (combinations numbered as described earlier):



Generally the influence of the parameters is recognizable by the "multiperiodic behaviour" w.r.t. to the chosen ordering. To get a clearer picture, we note at first some differences between type combinations: Besides the slight general increasing as already seen above (visible especially for $NC = 2.1$, i.e. the last 16 values for each type combination), type combinations 11111/2222 and, weaker, 11111/33333 show higher sensitivity w.r.t. the other parameters (as visible by greater differences between following points and peaks). The three remaining type combinations have rather similar curves. But altogether the behaviour of all type combinations is similar enough (pairwise correlation coefficients around 0.9 for each two sequences of 48 means) to take averages over type combinations and focus at the remaining $NC/RF/RS$-combinations. This yields the following more compact picture, now also taking into account the results for MIP:



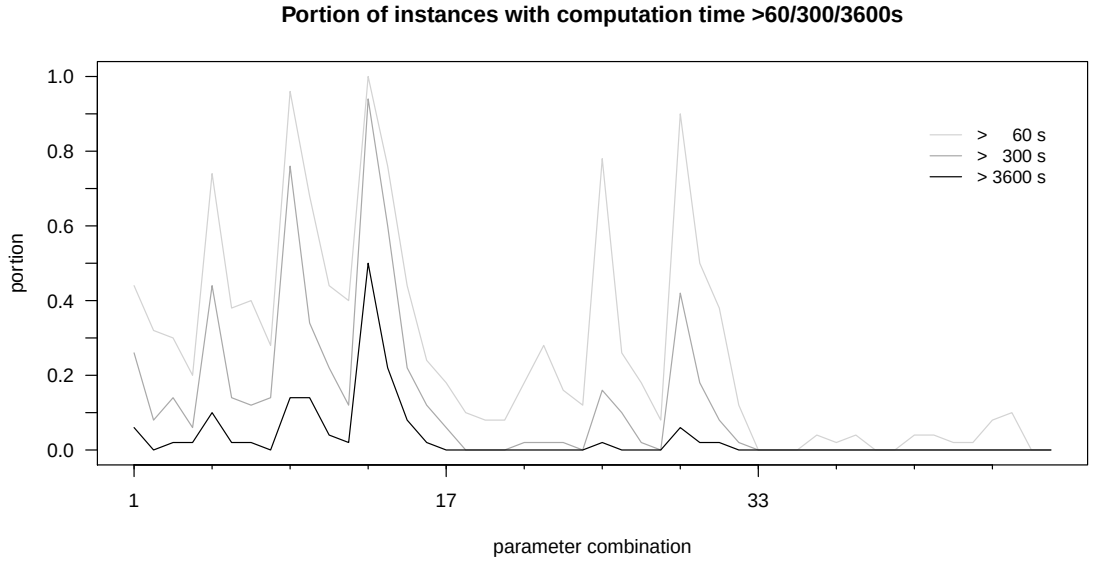
Averaged and hence also the uncertainties reduced, the mentioned multiperiodic behaviour in our ordering becomes better visible for B&B: In general, the values confirm the observations already made

above by fixing one parameter value, but here not only on average, but largely also for most individual parameter combinations: As can be seen by the periodicities in the graph, in most cases computation time decreases by increasing NC or RS (forward +16 and +1), or (to a lesser extent) by decreasing RF (backwards -4). This pattern is especially visible for higher computation times, whereas for smaller ones (especially for $NC = 2.1$) the general variability is smaller as well and the periodicities become blurred, possibly also due to statistical fluctuations of the means. As a consequence from this largely multi-periodic behaviour in our ordering, the effects of the parameters on computation time seem to be relatively independent, more complex interactions between them seem to be limited. Furthermore, since the dependencies seem to be relatively linear in logarithmic scale (corresponding to the roughly expontial behaviour observed by fixing one parameter value as seen above) the comparison with a linear model regression model could be interesting. The interested reader can find the results in the Appendix (8.4.3). - As a consequence of these dependencies, maximal computation time for each type combination is attained for $NC = 1.5, RF = 1, RS = 0.2$; the next difficult combinations are got by changing from this one to $RF = 0.75$ or $RS = 0.5$ (whereas changing to $NC = 1.8$ would reduce computation time somewhat more).

Qualitatively these dependencies can be explained as follows: Regarding network complexity, a higher NC means more precedence relations whence for more activity pairs the order in the schedule is determined, which in turn makes the scheduling problem easier. Considering resource factor RF, the situation becomes more difficult if more resource types are used by an activity on average, also since this increases the probability that two activities consume the same resource type. Finally, as far as resource strength RS is regarded, instances become easier if activities consume resources up to their capacities since in this case the probability increases that two activities cannot be scheduled parallel whence we get additional constraints. Generally lower NC, higher RF and lower RS increase the probability for resource conflicts which makes an instance more difficult, and this is just what can be seen in computation times for B&B.

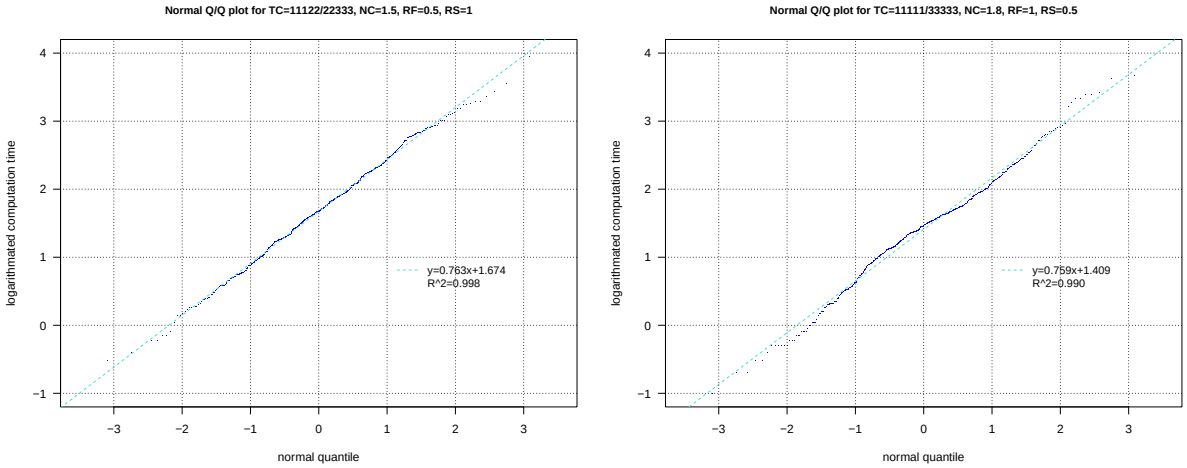
For MIP, dependency on RS and, less visible, on RF is qualitatively the same as for B&B; quantitatively their effect may be also similar as there, the exact values, however, are obviously biased by the high proportion of timeout instances. As far as NC is regarded, it is interesting to note that there is nearly no influence of NC, which is especially remarkable since for B&B this parameter has the largest effect. Consequently, the discrepancy between B&B and MIP increases especially by increasing NC, starting from typically less than a magnitude of order for $NC = 1.5$ to nearly two magnitudes for $NC = 2.1$. Thus, compared with B&B, MIP still perfoms best for $NC = 1.5$ and here especially for $RS = 1$, where the quotient between MIP and B&B value is around 3 relatively independent of the RF value. That NC has no significant effect shows that MIP cannot take advantage of a more favourable network structure.

Back again to B&B, the pattern observed for the averages is also reflected by the proportion of instances solved within given time bounds:



Again combination $NC = 1.5$, $RF = 1$, $RS = 0.2$ is clearly the hardest one (nearly 50% timeout instances), followed by that one changing to $RS = 0.5$ (about 20% timeouts). Generally it can be seen how the proportion of more difficult instances sharply decreases with increasing NC and also with increasing RS and decreasing RF. For example, the proportions of instances with $>3600\,\mathrm{s}$ for $NC = 1.5$ nearly correspond to those of instances with $>300\,\mathrm{s}$ for $NC = 1.8$ (and similar with $>300\,\mathrm{s}$ and $>60\,\mathrm{s}$, respectively), which means that by changing from $NC = 1.5$ to $NC = 1.8$ the harder instances become easier by about one magnitude of order. This continues to $NC = 2.1$, where almost all instances are solved within a minute even for the otherwise more difficult RF/RS-combinations. Generally there seems to be an approximately exponential decrease of the proportions of hard instances by changing to easier parameter combinations, also consistent with the similar behaviour already observed for the means.
This figure also makes another point partially visible, namely the high variability of computation times among instances generated for the same parameter combination. So it can be seen that parameter combinations with timeout instances also contain instances solvable within a minute (the only exception is the hardest combination, for which, however, one may assume that there would be still considerable variation beyond the time limit, given the high proportion of these instances). Indeed, for virtually each of the 240 parameter combinations it can be observed that already the 10 corresponding computation times vary considerably, in most cases over 1-3 magnitudes of order (powers of ten), with median 64.6 over all parameter combinations.

Motivated by this considerable variability even for the same parameter combination, we also regarded the distribution of computation times for several parameter combinations in somewhat more detail for the B&B. For this, for each chosen parameter combination 500 instances were generated and run with a time limit of 18000 s. In most cases the results strongly suggest a lognormal distribution, here illustrated for two parameter combinations:



For further results see 8.5 in the Appendix. So even if the computation times for the same parameter combination vary heavily, they do so according to a relatively simple distribution. Given the special structure of the algorithm, it seems quite remarkable to find such a smooth dependency.
Since this distribution conveys an impression of the computations times within a given parameter setting, it also complements in some sense the dependencies on parameter settings investigated before.

6 Conclusion

Motivated by a component exchange procedure in aircraft maintenance which admits to reduce duration of a maintenance project, a model called AEPSP (Activity Exchange Problem Scheduling Problem) has been defined. To solve this problem, a Branch-and-Bound algorithm has been developed, extending a corresponding approach for the RCPSP. Performance of this procedure has been demonstrated by comparison with a standard solver applied to the corresponding MIP. Also the dependency on parameter settings has been regarded in some detail, further showing that the B&B is able to use instance structure better.
Further research could be continued in different directions. For example, even if the Branch-and-Bound procedure shows reasonable performance, solution times are higher especially for some parameter settings, such that considerably larger instances would be out of reach to be solved exactly. Here it would be interesting to develop heuristics, for which one could start with those which already perform well for the RCPSP and try to modify them in a suitable way. Moreover, in a maintenance context one has often to deal with uncertainties w.r.t. components' states. This suggests to drop deterministic assumptions and to regard stochastic extensions, which is actually done in a subsequent paper.
Finally the model studied here may not only motivate to be developed further itself, but also to have a closer look at real projects in order to identify further operations (like here the exchange option in maintenance projects) which could lead to alternative models of flexible project scheduling. On the other hand a model can find new application areas; and in this sense, even if the model considered here comes from a more special issue in aircraft maintenance, we take it for conceivable that the idea to move work within project networks may be of more general interest for flexible project scheduling.

7 References

Abdolshah, M. (2014): A Review of Resource-Constrained Project Scheduling Problems (RCPSP) Approaches and Solutions. International Transaction Journal of Engineering, Management & Applied Sciences & Technologies 5, 253-286

Al-kaabi, H., A. Potter, and M. Nain (2007): An outsourcing decision model for airlines' MRO activities. Journal of Quality in Maintenance Engineering 13, 217-227

Artigues, C., S. Demassey, and E. Néron (2008): Resource-Constrained Project Scheduling: Models, Algorithms, Extensions and Applications. Wiley, Hoboken

Banghart, M. (2017): Identification of Reverse Engineering Candidates utilizing Machine Learning and Aircraft Cannibalization Data. International Journal of Aviation, Aeronautics, and Aerospace 4(4) Art. 5

Barták, R., and O. Čepek (2008): Nested Temporal Networks with Alternatives: Recognition, Tractability, and Models. Proceedings of the 13th International Conference on Artifical Intelligence: Methodology, Systems, and Applications, Varna, Bulgaria (AIMSA 2008)

Barták, R. (2011): Optimizing Alternatives in Precedence Networks. Proceedings of the 9th Mexican International Conference on Artificial Intelligence, Pachuca, Mexico (MICAI 2010)

Başdere, M., and Ü. Bilge (2014): Operational aircraft maintenance routing problem with remaining time consideration. European Journal of Operational Research 235, 315-328

Bazargan, M. (2016): Airline maintenance strategies- in-house vs. outsourced- an optimization approach. Journal of Quality in Maintenance Engineering 22, 114-129

Beck, J.C., and M.S. Fox (2000): Constraint-directed techniques for scheduling alternative activities. Artificial Intelligence 121, 211-250

Berthold, L., and M. Fliedner (2021): Optimal spare part utilization in maintenance projects and its complexity. Unpublished paper, Institute of Operations Management, University of Hamburg (part of this thesis)

Blazewicz, J., J.K. Lenstra, and A.H.G. Rinnooy Kan (1983): Scheduling subject to resource constraints: Classification and complexity. Discrete Applied Mathematics 5, 11-24

Brucker, P., and S. Knust (2012): Complex Scheduling. Springer Berlin, Heidelberg

Brucker, P., A. Drexl, R. Möhring, K. Neumann, and E. Pesch (1999): Resource-constrained project scheduling: Notation, classification, models, and methods. European Journal of Operational Research 112, 3-41

Budai, G., R. Dekker, and R.P. Nicolai (2006): A Review of Planning Models for Maintenance & Production. Technical Report, Econometric Institute, Erasmus University Rotterdam

Capacho, L., and R. Pastor (2008): ASALBP: the Alternative Subgraphs Assembly Line Balancing Problem. International Journal of Production Research 46(13), 3503-3516

Čapek, R., P. Šůcha, and Z. Hanzálek (2012): Production scheduling with alternative process plans. European Journal of Operational Research 217, 300-311

Chen, V.Y.X. (1994): A 0-1 goal programming model for scheduling multiple maintenance projects at a copper mine. European Journal of Operational Research 76, 176-191

Cobb, R. (1995): Modeling aircraft repair turntime. Journal of Air Transport Management 2, 25-32

De Bruecker, P., J. Van den Bergh, J. Belien, and E. Demeulemeester (2015): A model enhancement

heuristic for building robust aircraft maintenance personnel rosters with stochastic constraints. European Journal of Operational Research 246, 661-673

Demeulemeester, E., and W. Herroelen (1992): A Branch-and-Bound Procedure for the Multiple Resource-Constrained Project Scheduling Problem. Management Science 38(12), 1803-1818

Dinis, D., A. Barbosa-Póvoa, A.P. Teixeira (2019): A supporting framework for maintenance capacity planning and scheduling: Development and application in the aircraft MRO industry. International Journal of Production Economics 218, 1-15

Doganis, R. (2005): The Airline Business in the Twenty-first Century. Taylor & Francis, London

Domschke, W., A. Drexl, R. Klein, and A. Scholl (2015): Einführung in Operations Research (9. ed.). Springer Gabler, Wiesbaden

Elmaghraby, S.E. (1964): An Algebra for the Analysis of Generalized Activity Networks. Management Science 10, 494-514

Fabig, C., and E. Winter (2018): A multi-level modeling approach for simulation-based capacity planning and scheduling of aircraft maintenance projects. In: Proceedings of the 2018 Winter Simulation Conference, Gothenburg, Sweden

Firat, M., and C.A.J. Hurkens (2012): An improved MIP-based approach for a multi-skill workforce scheduling problem. Journal of Scheduling 15, 363-380

Froger, A., M. Gendreau, J.E. Mendoza, E. Pinson, and L.-M. Rousseau (2016): Maintenance scheduling in the electricity industry: A literature review. European Journal of Operational Research 251, 695-706

Ghobbar, A.A., and C.H. Friend (2002): Sources of intermittent demand for aircraft spare parts within airline operations. Journal of Air Transport Management 8, 221-231

Ghobbar, A.A., and C.H. Friend (2003): Evaluation of forecasting methods for intermittent parts demand in the field of aviation: a predictive model. Computers & Operations Research 30(14), 2097-2114

Gillies, D.W., and J.W.-S. Liu (1995): Scheduling Tasks with AND/OR Precedence Constraints. SIAM Journal on Computing 24, 797-810

Go, H., J.-S. Kim, and D.-H. Lee (2013): Operation and preventive maintenance scheduling for containerships: Mathematical model and solution algorithm. European Journal of Operational Research 229, 626-636

Goldratt, E.M. (1997): Critical Chain. North River Press, New York

Grigoriu, L. and D. Briskorn (2017): Scheduling jobs and maintenance activities subject to job-dependent machine deteriorations. Journal of Scheduling 20, 183-197

Gu, J., G. Zhang, and K.W. Li (2015): Efficient aircraft spare parts inventory management under demand uncertainty. Journal of Air Transport Management 42, 101-109

Habibi, F., F. Barzinpour, and S.J. Sadjadi (2018): Resource-constrained project scheduling problem: review of past and recent developments. Journal of Project Management 3, 55-88

Hartmann, S., and D. Briskorn (2010): A survey of variants and extensions of the resource-constrained project scheduling problem. European Journal of Operational Research 207, 1-14

Herroelen, W., and R. Leus (2001): On the merits and pitfalls of critical chain scheduling. Journal of Operations Management 19, 559-577

Herroelen, W., R. Leus, and E. Demeulemeester (2002): Critical Chain Project Scheduling: Do not oversimplify. Project Management Journal 33, 48-60

Hirsch, W.M., M. Meisner, and C. Boll (1968): Cannibalization in multicomponent systems and the theory of reliability. Naval Research Logistics Quarterly 15, 331-360

IATA (2018) Airline Maintenance Cost Executive Commentary, November 2018.

ISO 21500: Guidance on Project Management, 2012

Junqueira, V.S.V., M.S. Nagano, and H.H. Miyata (2018): Procedure structuring for programming aircraft maintenance activities. Revista de Gestão 27, 2-20

Kellenbrink, C., and S. Helber (2015): Scheduling resource-constrained projects with a flexible project structure. European Journal of Operational Research 246, 379-391

Kelley, J.E., and M.R. Walker (1959): Critical-Path Planning and Scheduling. Proceedings of the Eastern Joint Computer Conference 1959

Kilpi, J., and A.P.J. Vepsäläinen (2004): Pooling of spare components between airlines. Journal of Air Transport Management 10, 137-146

Kilpi, J., J. Töyli, and A. Vepsäläinen (2009): Cooperative strategies for the availability service of repairable aircraft components. International Journal of Production Economics 117, 360-370

Kinnison, H.A., and T. Siddiqui (2013): Aviation Maintenance Management. McGraw-Hill, New York

Kis, T. (2003): Job-shop scheduling with processing alternatives. European Journal of Operational Research 151, 307-332

Klein, R. (2000): Scheduling of Resource-Constrained Projects. Springer, Boston

Kolisch, R., and A. Sprecher (1996): PSPLIB - A project scheduling problem library. European Journal of Operational Research 96, 205-216

Kulkarni, A., D. Yadav, and H. Nikraz (2017): Aircraft maintenance checks using critical chain project path. Aircraft Engineering and Aerospace Technology 89(6), 879-892

Kuster, J., D. Jannach, and G. Friedrich (2009): Extending the RCPSP for modeling and solving disruption management problems. Applied Intelligence 31, 234-253

Kuster, J., D. Jannach, and G. Friedrich (2010): Applying Local Rescheduling in response to schedule disruptions. Annals of Operations Research 180, 265-282

Lee, S.G., Y.-S. Ma, G.L. Thimm, and J. Verstraeten (2008): Product lifecycle management in aviation maintenance, repair and overhaul. Computers in Industry 59, 296-303

Li, H., and K. Womer (2009): Scheduling projects with multi-skilled personnel by a hybrid MILP/CP benders decomposition algorithm. Journal of Scheduling 12, 281-298

Liu, Y., Y. Feng, X. Xue, and C. Lu (2018): Research on Multi-Echelon Inventory System for Civil Aircraft Spare Parts with Lateral Transshipments and Importance Degree. 12th International Conference on Reliability, Maintainability, and Safety (ICRMS) 2018, Shanghai, China

Malcolm, D.G., J.H. Roseboom, C.E. Clark, and W. Fazar (1959): Application of a Technique for Research and Development Program Evaluation. Operations Research 7, 646-669

Masmoudi, M., and A. Hait (2012): Fuzzy uncertainty modelling for project planning: application to helicopter maintenance. International Journal of Production Research 50(13), 3594-3611

McKendall, A.R., J.S. Noble, and C.M. Klein (2008): Scheduling Maintenance Activities During

Planned Outages At Nuclear Power Plants. International Journal of Industrial Engineering 15, 53-61

Megow, N., R.H. Möhring, and J. Schulz (2011): Decision Support and Optimization in Shutdown and Turnaround Scheduling. INFORMS Journal on Computing 23, 189-204

Moshman, J., J. Johnson, and M. Larsen (1963): RAMPS - A Technique for Resource Allocation and Multi-Project Scheduling. Proceedings of the Spring Joint Computer Conference 1963

Néron, E. (2002): Lower Bounds for the Multi-Skill Project Scheduling Problem. Proceeding of the 8th International Workshop on Project Management and Scheduling, Spain, 274-277

Papakostas, N., P. Papachatzakis, V. Xanthakis, D. Mourtzis, and G. Chryssolouris (2010): An approach to operational aircraft maintenance planning. Decision Support Systems 48, 604-612

Philips, J., and O. Hsiang (2000): Aircraft asset-backed securities. In: F.J. Fabozzi (ed.): Investing in Asset-Backed Securities, 151-168. Wiley, Hoboken

Pintelon, L.M., and L.F. Gelders (1992): Maintenance management decision making. European Journal of Operational Research 58, 301-317

PMI 2017: A Guide to the Project Management Body of Knowledge. Project Management Institute

Regattieri, A., M. Gamberi, R. Gamberini, R. Manzini (2005): Managing lumpy demand for aicraft spare parts. Journal of Air Transport Management 11, 426-431

Regattieri, A., A. Giazzi, M. Gamberi, and R. Gamberini (2015): An innovative method to optimize the maintenance policies in an aircraft: General framework and case study. Journal of Air Transport Management 44-45, 8-20

Reményi, C., and S. Staudacher (2014): Systematic simulation based approach for the identification and implementation of a scheduling rule in the aircraft engine maintenance. International Journal of Production Economics 147, 94-107

Rosales, L.J.S., J.-B. Yang, and Y.-W. Chen (2014): Analysing delays and disruptions in Aircraft Heavy Maintenance. Proceedings of the 32nd International Conference of the System Dynamics Society 20.-23.07.14, Delft, Netherlands

Rosales, L.J.S. (2015): Analysing uncertainty and delays in Aircraft Heavy Maintenance. Ph.D. Thesis, University of Manchester, Manchester, UK

Safaei, N., and A.K.S. Jardine (2018): Aircraft routing with generalized maintenance constraints. Omega 80, 111-122

Salewski, F., A. Schirmer, and A. Drexl (1997): Project scheduling under resource and mode identity constraints: Model, complexity, methods, and application. European Journal of Operational Research 102, 88-110

Salman, S., C.R. Cassady, E.A. Pohl, and S.W. Ormon (2007): Evaluating the Impact of Cannibalization on Fleet Performance. Quality and Reliability Engineering International 23, 445-457

Samaranayake, P. (2006): Current Practices and Problem Areas in Aircraft Maintenance Planning and Scheduling - Interfaced/Integrated System Perspective. Proceedings of the 7th Asia Pacific Industrial Engineering and Management Systems Conference 2006, Bangkok, Thailand

Samaranayake, P., and S. Kiridena (2012): Aircraft maintenance planning and scheduling: an integrated framework. Journal of Quality in Maintenance Engineering 18, 432-453

Satoglu, R., N. Humaira, and G. Inalhan (2016): Aircraft Scheduled Airframe Maintenance and Downtime Integrated Cost Model. Advances in Operations Research, Volume 2016

Schwindt, C., and J. Zimmermann (eds.) (2015): Handbook on project management and scheduling. Springer, Berlin

Servranckx, T., and M. Vanhoucke (2019): A tabu search procedure for the resource-constrained project scheduling problem with alternative subgraphs. European Journal of Operational Research 273, 841-860

Srinivasan, M.M., W.D. Best, and S. Chandrasekaran (2007): Warner Robins Air Logistics Center Streamlines Aircraft Repair and Overhaul. INFORMS Journal on Applied Analytics 37, 7-21

Stadnicka, D., D. Arkhipov, O. Battaia, and R.M.C. Ratnayake (2017): Skills management in the optimization of aircraft maintenance processes. International Federation of Automatic Control Papers Online 50(1), 6912-6917

Tao, S., and Z.S. Dong (2017): Scheduling resource-constrained project problem with alternative activity chains. Computers & Industrial Engineering 114, 288-296

Van Peteghem, V., and M. Vanhoucke (2014): An experimental investigation of metaheuristics for the multi-mode resource-constrained project scheduling problem on new dataset instances. European Journal of Operational Research 235, 62-72

Vanhoucke, M., and J. Coelho (2016): An approach using SAT solvers for the RCPSP with logical constraints. European Journal of Operational Research 249, 577-591

Vieira, D.R., and P.L. Loures (2016): Maintenance, Repair and Overhaul (MRO) Fundamentals and Strategies: An Aeronautical Industry Overview. International Journal of Computer Applications 135(12), 21-29

Wang, H. (2002): A survey of maintenance policies of deteriorating systems. European Journal of Operational Research 139, 469-489

Wang, Y., R.L. Cheu, and T.F. Fwa (2002): Highway Maintenance Scheduling using Genetic Algorithm with Microscopic Traffic Simulation. Proceedings of the 81st Annual Meeting of the Transportation Research Board, 13-17

Węglarz, J., J. Józefowska, M. Mika, and G. Waligóra (2011): Project scheduling with finite or infinite number of activity processing modes - A survey. European Journal of Operational Research 208, 177-205

Werbińska-Wojciechowska, S. (2019): Technical System Maintenance. Springer Series in Reliability Engineering, Springer

Wiest, J.D. (1963): The Scheduling of Large Projects with Limited Resources. Ph.D. thesis, Carnegie Institute of Technology, Pittsburgh

Yadav, D.K. (2010): Licensing and recognition of the aircraft maintenance engineers - A comparative study. Journal of Air Transport Management 16, 272-278

8 Appendix

8.1 MIP

Sets and indices:

$A/A^{fx}/A^{fl}$: set of activities/fixed activities/flexible activities
i, j: activities
d_i: duration of fixed activity i
d_τ: duration associated with type τ (for repair task for component of defective component of type τ)
P_i/S_i: set of predecessors/successors of i
R: set of resources
r: resource
$\mathcal{T}$: set of types
τ: type
$\tau(i)$: type of (flexible) activity i
T: time horizon
t: time
$[t]$: time slot $[t, t+1]$
M, M': sufficiently high numbers
M: number of marginal time steps corresponding to one normal time step ($M > |A^{fl}|$ suffices)
M': sufficiently high number ($M' > max_{r \in \mathcal{R}} cp_r$ suffices)

Variables:

continuous:

dv_i: actual duration of activitiy i
csv_{ir}: actual consumption of activity i of resource r
$csvt_{irt}$: consumption of resource r by activity i in time slot t
f_i: end time of activity i
$ac^1_{\tau t}$: number of available components of type τ in time slot t
$ac^0_{\tau t}$: number of available components of type τ in time slot t
el_i: time at which activity i becomes eligible

binary:

af_{it}: 1 if i finishes at some $t' \leq t$ ("already finished")
as_{it}: 1 if i starts at some $t' \leq t$ ("already started")
ael_{it}: 1 if i becomes eligible at some $t' \leq t$
act_{it}: 1 if i is active in time slot $[t] = [t, t+1]$
y_i: 1 if i is processed in state 1
u_{it}: 1 if i is processed in state 0 at t
w_{it}: auxiliary variable equal to $min\{as_{it} - as_{i,t-1}, y_i\}$

Model:

(0) $min\, MS = f_{n+1}$
s.t.
(1) $f_0 = 0$
(2) $f_i \leq el_j$ f.a. $j \in A$, $i \in P_j$
(3) $el_i + dv_i \leq f_i$ f.a. $i \in A$
(4) $as_{i,t} \leq as_{i,t+1}$ f.a. $t = 0, \ldots, T-1\}$
(5) $f_i - dv_i = T + 1 - \sum_{t=0}^{T} as_{it}$ f.a. $i \in A$

(6) $af_{it} \leq af_{i,t+1}$ f.a. $0 \leq t \leq T-1$
(7) $f_i = T + 1 - \sum_{t=0}^{T} af_{it}$ f.a. $i \in A$
(8) $ael_{it} \leq ael_{it+1}$ f.a. $i \in A$, $t = 0, \ldots T-1$
(9) $el_i = T + 1 - \sum_{t=0}^{T} ael_{it}$ f.a. $i \in A$
(10) $dv_i = M \cdot d_i$ f.a. $i \in A^{fx}$
(11) $dv_i \geq M \cdot d_\tau \cdot y_i$ f.a. $i \in A^{fl}$, $\tau = \tau(i)$
(12) $dv_i \geq 1$ f.a. $i \in A^{fl}$
(13) $y_i = 1$ f.a. $i \in A^{fx}$
(14) $\sum_{t=}^{T} u_{it} = 1 - y_i$ f.a. $i \in A$
(15) $u_{i,t+1} \geq as_{i,t+1} - as_{it} - y_i$ f.a. $i \in A$
(16) $w_{i,t+1} \leq as_{i,t+1} - as_{i,t}$ f.a. $i \in A$
(17) $w_{it} \leq y_i$ f.a. $i \in A$, $t = 0, \ldots T$
(18) $w_{i,t+1} \geq as_{i,t+1} - as_{it} + y_i - 1$ f.a. $i \in A$, $t = 0, \ldots, T-1$
(19) $act_{it} = as_{it} - af_{it}$ f.a. $i \in A$, $t = 0, \ldots, T-1$
(20) $\sum_{t=0}^{T-1} act_{it} = dv_i$ f.a. $i \in A$
(21) $\sum_{i \in A} csvt_{irt} \leq cp_r$ f.a. $r \in \mathcal{R}, t$
(22) $csv_{ir} = cs_{ir}$ f.a. $i \in A^{fx}$
(23) $csv_{ir} = cs_{\tau r} \cdot y_i$ f.a. $i \in A^{fl}$, $\tau = \tau(i)$
(24) $csvt_{irt} \geq csv_{ir} - M' \cdot (1 - act_{it})$ f.a. $i \in A$, $r \in \mathcal{R}$, $t = 0, \ldots T$
(25) $ac^1_{\tau 0} = 0$ f.a. $\tau \in \mathcal{T}$
(26) $ac^0_{\tau 0} = 0$ f.a. $\tau \in \mathcal{T}$
(27) $ac^1_{\tau,t+1} = ac^1_{\tau} + \sum_{i \in A_\tau, \sigma_0(i)=1} (ael_{i,t+1} - ael_{it}) - \sum_{i \in A_\tau} w_{i,t+1}$ f.a. $\tau \in \mathcal{T}$, $t = 0, \ldots, T-1$
(28) $ac^0_{\tau,t+1} = ac^0_{\tau t} + \sum_{i \in A_\tau, \sigma_0(i)=0} (ael_{i,t+1} - ael_{it}) - \sum_{i \in A_\tau} u_{i,t+1}$ f.a. $\tau \in \mathcal{T}$, $t = 0, \ldots, T-1$
(29) $ac^1_{\tau t} \geq 0$ f.a. $\tau \in \mathcal{T}$, $t = 0, \ldots, T$
(30) $ac^0_{\tau t} \geq 0$ f.a. $\tau \in \mathcal{T}$, $t = 0, \ldots, T$

Comments :

(0) Objective is makespan, i.e. project duration, which is given by the end time of the end activity.
(1) Start activity ends at 0
(2) guarantees that an activity becomes eligible only if all its predecessors have been finished.
(3) states that an activity can start only if it has become eligible.
(4), (6) and (8) capture monotonicity of the "already started/finished" variables
(5) and (7) link finish time with the variables,
(9) links time of becoming eligible with the "already eligible" variables
(10) translates duration in normal time units into marginal time steps for fixed activities
(11) as (10) for flexible activities, depending on their insertion state
(12) ensure that each flexible activity takes at least one marginal time unit
(13) sets the state variable to 1 for fixed activities
(14)+(15) define auxiliary variable u_{it}
(16)-(18) define auxiliary variable $w_{it} = min\{as_{it} - as_{i,t-1}, y_i\}$
(19) reflects that an activity is active in a time slot if it has been already started but not finished yet
(20) states that the duration of an activity corresponds to the number of time slots where it is active
(21) ensure that capacity is not exceeded in each time slot
(22) states that the actual resource consumption for a fixed activity is just as given for that activity, whereas for a flexible activity it is determined by its state and the type-specific consumption
(23) ensures that flexible activities only consume resources if they are processed as repair task
(24) lets an activity consume resources in a time slot an if it is active there
(25)-(30) define the availability counter where:
(25)+(26) are the initial conditions,

(27)+(28) define changes from one time step to the next one and
(29)+(30) ensure non-negativity

8.2 Proof of Theorem 1 (Completeness of branching with minimal alternatives)

Theorem 1. *Let $\mathcal{E}$ be an instance of the AEPSP. Then the search tree using branching with minimal alternatives contains a node corresponding to an optimal schedule.*

To prove this result we show the more general assertion that this also holds locally, i.e. from each node in the branching tree there is a path leading to an optimal extension of the corresponding decision point, i.e. it can be reached by a sequence of minimal alternatives. To state this more rigorously we need the following notion of a continuation:

Given a decision point $\delta = (\mathfrak{p}, \underline{t})$, a *continuation* of $\mathfrak{p}$ at $\underline{t}$ (or simply of δ) is a (feasible) complete schedule $\mathfrak{q}$
- which agrees with $\mathfrak{p}$ on $S^{\mathfrak{p}}$ with the possible exception that some fixed activities $i \in S^{\mathfrak{p}}$ with $\underline{f}^{\mathfrak{p}}(i) > \underline{t}$ may start in $\mathfrak{q}$ at $\underline{t}$ or later instead as in $\mathfrak{p}$; and
- in which all activities $i \notin S^{\mathfrak{p}}$ start at $\underline{t}$ or later.

A continuation $\mathfrak{q}$ is *optimal* if $\underline{f}^{\mathfrak{q}}(n+1) \leq \underline{f}^{\mathfrak{q}'}(n+1)$ for each other continuation $\mathfrak{q}'$.

Main motivation for this definition is the fact that a complete schedule $\mathfrak{q}$ which arises from $\mathfrak{p}$ at $\underline{t}$ by a sequence of minimal alternatives is a continuation of $\mathfrak{p}$ at $\underline{t}$: For this regard any activity i; further assume $i \in S^{\mathfrak{p}}$ at first. If i is flexible or if $\underline{f}^{\mathfrak{p}}(i) \leq \underline{t}$ then the schedule of i cannot be changed anymore by a minimal alternative whence i is scheduled in $\mathfrak{q}$ as in $\mathfrak{p}$. If i is fixed and finishes after $\underline{t}$ in $\mathfrak{p}$ then either it changes its schedule by some minimal alternative(s) or it is kept scheduled as before. In the first case it will start at $\underline{t}$ or later in $\mathfrak{q}$, in the second case it will be be scheduled in $\mathfrak{q}$ as in $\mathfrak{p}$. On the other hand, if we assume $i \notin S^{\mathfrak{p}}$ then it is started by a minimal alternative at some point $\underline{t} \geq \underline{t}'$ and then possibly delayed further whence it starts in $\mathfrak{q}$ at $\underline{t}$ or later. Altogether i satisfies the corresponding condition necessary for $\mathfrak{q}$ being a continuation of $\mathfrak{p}$ at $\underline{t}$.

Now we can state the assertion to be shown as follows:

Theorem 1 (extended)**.** *Let $\mathcal{E}$ be any instance of the AEPSP and $\delta = (\mathfrak{p}, \underline{t})$ a decision point corresponding to a node of the search tree using branching with minimal alternatives. Then there is a sequence of minimal alternatives which leads to an optimal continuation of δ. In other words, from each node of the search tree leads a path to an optimal continuation (of that node).*

Before starting with the actual proof for Theorem 1, we still need the following useful notions:

We say that two schedules *agree* on a set of activities if they agree for fixed activities w.r.t. $\underline{s}$ and (hence) $\underline{f}$ and for flexible activities w.r.t. $\underline{s}, \underline{f}$ (and hence) σ.

For a schedule $\mathfrak{p}$ and a time point $\underline{t}$, we denote by $\mathfrak{p}|_{\underline{t}}$ the schedule $\mathfrak{p}$ restricted to activities starting before $\underline{t}$, i.e. $S^{\mathfrak{p}|_{\underline{t}}} = \{i \in S^{\mathfrak{p}} | \underline{s}^{\mathfrak{p}}(i) < \underline{t}\}$ with values for $\underline{s}, \underline{f}$ and σ as in $\mathfrak{p}$. Note that this is a feasible schedule.

Now we can turn to the proof of Theorem 1:

Proof of Theorem 1. 0. (outline of the following proof)
Since, worked out in detail, the following proof is somewhat complicated and lengthy, we first give an outline of its general structure: At first we choose an optimal continuation of $\mathfrak{p}$ at $\underline{t}$, denoted by $\mathfrak{q}$, with a "dominance property", essentially meaning that it has minimal start times. For such a dominant schedule we show that each start point coincides with some end point. The rest of the proof

consists of showing that $\mathfrak{q}$ can be reached via a sequence of minimal alternatives and decision points as intermediate steps. This sequence will be constructed by induction, using a bundle of properties which an intermediate decision point δ_l must have to be able to construct the next decision point δ_{l+1}. Being at such δ_l, the induction step does not only consist of giving its sucessor but one has also to take care that it can be reached by application of a suitable minimal alternative. After that one has also to check that the new $\mathfrak{q}_{l+1}$ satisfies again all properties to be itself a starting point for the next step of the induction. This will involve some closer look to exclude undesired configurations. At several points auxiliary schedules are constructed, in most cases to lead an assumption to a contradiction to establish the right property. Showing that these schedules are actually feasible will often make more work than simply describing their construction.

1. (dominance; fixing a dominant continuation; sequence of end times)
Let $\mathcal{E}$ and $\delta = (\mathfrak{p}, \underline{t})$ be given as in the theorem. For the purpose of the following proof we will say that, for two continuations $\mathfrak{q}$ and $\mathfrak{q}'$ of $\mathfrak{p}$ at $\underline{t}$, continuation $\mathfrak{q}$ *dominates* $\mathfrak{q}'$ if $\underline{s}^{\mathfrak{q}}(i) \leq \underline{s}^{\mathfrak{q}'}(i)$ for all $i \in A$ and $\underline{s}^{\mathfrak{q}}(i) < \underline{s}^{\mathfrak{q}'}(i)$ for at least one $i \in A$. Note that different continuations will agree on fixed nodes ending up to $\underline{t}$ and on flexible nodes starting before $\underline{t}$, i.e. on $\{i \in S^{fx,\mathfrak{p}} | \underline{f}^{\mathfrak{p}}(i) \leq \underline{t}_l\} \cup \{i \in S^{fl,\mathfrak{p}} | \underline{s}^{\mathfrak{p}}(i) < \underline{t}_l\}$ anyway whence this dominance concerns the remaining activities - fixed activities scheduled in $\mathfrak{p}$ ending after $\underline{t}$ or activities not scheduled in $\mathfrak{p}$ at all, i.e. on $\{i \in S^{fx,\mathfrak{p}} | \underline{f}^{\mathfrak{p}}(i) > \underline{t}\} \cup (A \backslash S^{\mathfrak{p}}))$. We call a continuation $\mathfrak{q}$ *dominant* if there is no other continuation $\mathfrak{q}'$ which dominates $\mathfrak{q}$ (Note that such a dominant continuation really exists albeit not being unique in general). For the rest of the proof fix such a dominant continuation $\mathfrak{q}$ of $\mathfrak{p}$ at $\underline{t}$. In the following we will show that there is a sequence of minimal alternatives leading from $\mathfrak{p}$ at $\underline{t}$ to $\mathfrak{q}$.
Let $\underline{t}_0 = \underline{t}$ and $\underline{t}_1 < \ldots < \underline{t}_k$ the ordered sequence of time points greater than $\underline{t}$ which actually appear as end points of $\mathfrak{q}$, i.e. which are contained in the set $\{\underline{f}^{\mathfrak{q}}(i) | \underline{f}^{\mathfrak{q}}(i) > \underline{t}, i \in A\}$ (every point appearing once, i.e. identical $\underline{f}^{\mathfrak{q}}(i) = \underline{f}^{\mathfrak{q}}(j)$ are represented by only one $\underline{t}_l$).

2. (each start point falls onto some end point)
As preliminary step we will show that each start point of $\mathfrak{q}$ later than $\underline{t}$ coincides with some end point, i.e. $\{\underline{s}^{\mathfrak{p}}(i) | \underline{s}^{\mathfrak{p}}(i) \geq \underline{t}, i \in A\} \subseteq \{\underline{t}_l | 0 \leq l \leq k\}$. This property follows from the dominancy of $\mathfrak{q}$, since if it would not hold then one could prepone one of those activities which does not start at some end point and would get a continuation which dominates $\mathfrak{q}$, contradiction. In the following we will show this in some more detail, especially making sure that that preponement leads to a schedule which is feasible indeed (and here especially w.r.t. the novel insertion condition).
Assume to the contrary that there is some $i \in A$ with $\underline{s}^{\mathfrak{q}}(i) \geq \underline{t}$ and $\underline{s}^{\mathfrak{q}}(i) \notin \{\underline{t}_0, \ldots, \underline{t}_k\}$, from which we get $\underline{t}_l < \underline{s}^{\mathfrak{q}}(i) < \underline{t}_{l+1}$ for some $0 \leq l < k$. Regard the schedule $\mathfrak{q}'$ resulting from $\mathfrak{q}$ by starting i already at $\underline{t}_l$ and ending correspondingly and leaving everything else unchanged. To show that $\mathfrak{q}'$ is feasible, we check the feasibility conditions for $\mathfrak{q}'$:

- *precedence*: Let i_1, i_2 be activities with i_1 predecessor of i_2. If $i \notin \{i_1, i_2\}$ then i_1, i_2 are scheduled as in $\mathfrak{q}$ and precedence is satisfied. If $i_1 = i$ then i_1 finishes earlier than in $\mathfrak{q}$ and precedence is satisfied as well. For $i_2 = i$ note that each predecessor i_1' of i does not finish later than $\underline{s}^{\mathfrak{q}}(i)$ in $\mathfrak{q}$ and hence not later than the last end point up to $\underline{s}^{\mathfrak{q}}(i)$, which is $\underline{t}_l = \underline{s}^{\mathfrak{q}'}(i)$. Thus, in $\mathfrak{q}'$, i_1 does not finish later than $\underline{s}^{\mathfrak{q}'}(i) = \underline{s}^{\mathfrak{q}'}(i_2)$ as well, whence precedence is satisfied for all cases.

- *duration*: satisfied since satisfied in feasible $\mathfrak{q}$ and additional preponement of i does not change its duration

- *resources*: For each resource ρ, the resource consumption is decreased only if some activity ends and resources are set free again. Thus, since no activity ends in $(\underline{t}_l, \underline{s}^{\mathfrak{q}}(i)]$ in $\mathfrak{q}$, the resource consumption in $\mathfrak{q}$ is monotonically increasing from slot $[\underline{t}_l]$ to slot $[\underline{s}^{\mathfrak{q}}(i)]$. Starting now i at $\underline{t}_l$ for $\mathfrak{q}'$, i ends in $\mathfrak{q}'$ before $\underline{f}^{\mathfrak{q}}(i)$. In $[\underline{t}_l, \underline{s}^{\mathfrak{q}}(i)]$ for $\mathfrak{q}'$, the resources suffice since their consumption is at most as high as in $[\underline{s}^{\mathfrak{q}}(i)]$ in $\mathfrak{q}$ due to monotonicity. In $[\underline{s}^{\mathfrak{q}}(i), \underline{f}^{\mathfrak{q}'}(i)]$ for $\mathfrak{q}'$ (if $\underline{f}^{\mathfrak{q}'}(i) > \underline{s}^{\mathfrak{q}}(i)$, otherwise this case need no to be

considered) the resources suffice since $\mathfrak{q}$ is (resource-)feasible due to assumption. Thus the resources suffice for $\mathfrak{q}'$ on $[\underline{s}^{\mathfrak{q}'}(i), \underline{f}^{\mathfrak{q}'}(i)] \subseteq [\underline{t}_l, \underline{f}^{\mathfrak{q}}(i)]$. Beyond the first interval, the resource consumption remains unchanged compared with $\mathfrak{q}$ whence $\mathfrak{q}'$ is resource-feasible as a whole.

- *insertion*: One has to show that $cc^{\mathfrak{q}'}_{\underline{t}'}(\tau,\sigma) \geq 0$ for each combination (τ,σ) and each time $\underline{t}'$. For this fix any combination (τ,σ) and recall that
$cc^{\mathfrak{q}'}_{\underline{t}'}(\tau,\sigma) = |\{i' \in A_\tau | \sigma_0(i') = \sigma \ \wedge \ \underline{el}^{\mathfrak{q}'}(i) \leq \underline{t}'\}| - |\{i' \in A_\tau | \sigma^{\mathfrak{q}'}(i') = \sigma \ \wedge \ \underline{s}^{\mathfrak{q}'}(i') \leq \underline{t}'\}|$
i.e. the number of (τ,σ)-components which have become available up to $\underline{t}'$ (since the activities associated with the positions, where those components were initially contained, have become eligible) minus the number of (τ,σ)-components which have been consumed (since their processing has started at their final positions, reflected by the start of associated activities) is nonnegative. This already holds for $\mathfrak{q}$ due to its feasibility, so check how this expression is changed by the preponement of i: For this, denote the first term of the difference above by $cc^{\mathfrak{q}'+}_{\underline{t}'}(\tau,\sigma)$, the second (without negative sign) by $cc^{\mathfrak{q}'-}_{\underline{t}'}(\tau,\sigma)$ such that $cc^{\mathfrak{q}'}_{\underline{t}'}(\tau,\sigma) = cc^{\mathfrak{q}'+}_{\underline{t}'}(\tau,\sigma) - cc^{\mathfrak{q}'-}_{\underline{t}'}(\tau,\sigma)$ (and correspondingly for $\mathfrak{q}$). Note that both cc^+- and cc^--part are monotonically increasing over time.
On the one hand, concerning the cc^+-part, preponing i means finishing i earlier whence no activity becomes eligible later and some activities may become eligible earlier in $\mathfrak{q}'$ compared with $\mathfrak{q}$, from which follows that the number of activities having become eligible up to given time remains the same or increases, i.e. $cc^{\mathfrak{q}'+}_{\underline{t}'}(\tau,\sigma) \geq cc^{\mathfrak{q}+}_{\underline{t}'}(\tau,\sigma)$ for each $\underline{t}'$. More precisely, preponing may lead to a strict inequality only for $\underline{t}' = \underline{f}^{\mathfrak{q}'}(i), \ldots, \underline{f}^{\mathfrak{q}}(i) - 1$ whereas for other times both values agree.
Further, since in $\mathfrak{q}$ no activity ends at $\underline{t}_l < \underline{t}' < \underline{t}_{l+1}$, also no activity becomes newly eligible at these time points whence $cc^{\mathfrak{q}}_{\underline{t}'}(\tau,\sigma)$ has the same value for $\underline{t}' = \underline{t}_l, \ldots, \underline{t}_{l+1} - \underline{1}$, which is especially equal to $cc^{\mathfrak{q}}_{\underline{s}^{\mathfrak{q}}(i)}(\tau,\sigma)$.
On the other hand, concerning the cc^--part, preponing i means to consume a (τ,σ)-component already at $\underline{s}^{\mathfrak{q}'}(i) = \underline{t}_l$ instead at $\underline{s}^{\mathfrak{q}}(i)$, but only if i does actually consume a (τ,σ)-component, i.e. i is an activity of type τ and is processed in state σ. In this case one has $cc^{\mathfrak{q}'-}_{\underline{t}'}(\tau,\sigma) = cc^{\mathfrak{q}-}_{\underline{t}'}(\tau,\sigma) + 1$ for $\underline{t}' = \underline{t}_l, \ldots, \underline{s}^{\mathfrak{q}}(i) - \underline{1}$ and equality of the counter values for other times (otherwise, if i does not consume any (τ,σ)-component the values for $\mathfrak{q}$ and $\mathfrak{q}'$ are the same for all $\underline{t}'$). Especially one has $cc^{\mathfrak{q}'-}_{\underline{t}'}(\tau,\sigma) \leq cc^{\mathfrak{q}'-}_{\underline{s}^{\mathfrak{q}}(i)}(\tau,\sigma) = cc^{\mathfrak{q}-}_{\underline{s}^{\mathfrak{q}}(i)}(\tau,\sigma)$ for each $\underline{t}' \leq \underline{s}^{\mathfrak{q}}(i)$ due to monotonicity.
So if i does not consume any (τ,σ)-component we have
$cc^{\mathfrak{q}'}_{\underline{t}'}(\tau,\sigma) = cc^{\mathfrak{q}'+}_{\underline{t}'}(\tau,\sigma) - cc^{\mathfrak{q}'-}_{\underline{t}'}(\tau,\sigma) \geq cc^{\mathfrak{q}+}_{\underline{t}'}(\tau,\sigma) - cc^{\mathfrak{q}-}_{\underline{t}'}(\tau,\sigma) = cc^{\mathfrak{q}}_{\underline{t}'}(\tau,\sigma) \geq 0$ for all $\underline{t}'$
due to the $\geq$-inequality for the cc^+- and equality for the cc^--part.
If i does consume a (τ,σ)-component the same applies for $\underline{t}' < \underline{t}_l$ and $\underline{t}' \geq \underline{s}^{\mathfrak{q}}(i)$. For the remaining times we have
$cc^{\mathfrak{q}'-}_{\underline{t}'}(\tau,\sigma) = cc^{\mathfrak{q}-}_{\underline{t}'}(\tau,\sigma) + 1 \leq cc^{\mathfrak{q}-}_{\underline{s}^{\mathfrak{q}}(i)-\underline{1}}(\tau,\sigma) + 1 = cc^{\mathfrak{q}-}_{\underline{s}^{\mathfrak{q}}(i)}(\tau,\sigma) \leq cc^{\mathfrak{q}+}_{\underline{s}^{\mathfrak{q}}(i)}(\tau,\sigma) = cc^{\mathfrak{q}+}_{\underline{t}'}(\tau,\sigma) \leq cc^{\mathfrak{q}'+}_{\underline{t}'}(\tau,\sigma)$
and hence $cc^{\mathfrak{q}'}_{\underline{t}'}(\tau,\sigma) = cc^{\mathfrak{q}'+}_{\underline{t}'}(\tau,\sigma) - cc^{\mathfrak{q}'-}_{\underline{t}'}(\tau,\sigma) \geq 0$ for each $\underline{t}' = \underline{t}_l, \ldots, \underline{s}^{\mathfrak{q}}(i) - \underline{1}$.
This reflects the fact that it does not matter to consume the component already at $\underline{t}_l$ since this is the last time up to $\underline{s}^{\mathfrak{q}}(i)$ at which new (τ,σ)-components may have become available; i.e. a component consumed at $\underline{s}^{\mathfrak{q}}(i)$ was already available at $\underline{t}_l$.
Altogether the component counter for $\mathfrak{q}'$ is non-negative, whence $\mathfrak{q}'$ is insertion-feasible.

($\mathfrak{q}'$ is actually a continuation)
Having thus a feasible complete schedule, check finally that $\mathfrak{q}'$ is actually a continuation of $\mathfrak{p}$ at $\underline{t}$: Since each activity $i' \neq i$ is scheduled in $\mathfrak{q}'$ as in the continuation $\mathfrak{q}$, each activity $i' \neq i$ further satisfies the corresponding condition necessary for a continuation (for $i' \in S^{\mathfrak{p}} \wedge (i' \in A^{fl} \vee \underline{f}^{\mathfrak{p}}(i') \leq \underline{t})$, for $i' \in S^{fx,\mathfrak{p}} \wedge \underline{f}^{\mathfrak{p}}(i') > \underline{t}$ as well for $i' \notin S^{\mathfrak{p}}$). Regarding i, note that it starts not before $\underline{t}$ in $\mathfrak{q}$ and after the preponement in $\mathfrak{q}'$ as well since $\underline{t}_l \geq \underline{t}$. Thus the condition necessary for i is kept satisfied in $\mathfrak{q}'$.

Thus $\mathfrak{q}'$ is a continuation of $\mathfrak{p}$ at $\underline{t}$, and since i starts earlier while other activities remain unchanged compared with $\mathfrak{q}$, $\mathfrak{q}'$ would dominate $\mathfrak{q}$, contradiction. Thus the initial assumption of some start point not coinciding with any end point has turned out to be false whence all start points of the dominant continuation $\mathfrak{q}$ greater than or equal to $\underline{t}$ are contained in $\{\underline{t}_0, \ldots, \underline{t}_k\}$ indeed.

3. In the rest of the proof we will show that $\mathfrak{q}$ is contained in the branching tree, i.e. that it (or, more precisely, the corresponding decision point $\delta_\mathfrak{q} = (\mathfrak{q}, \underline{f}^\mathfrak{q}(n+1))$) can be reached by a sequence of appropriately chosen minimal alternatives. This will be shown by induction, and to get a suitable induction assumption, we have to show the following more general assertion:

Claim: There is a sequence of minimal alternatives $(\mu_l)_{1\le l\le k} = (Dl_l, St_l^0)_{1\le l\le k}$ which, starting from $\delta = (\mathfrak{p}, \underline{t})$, results in $\delta_k = (\mathfrak{q}, \underline{f}^\mathfrak{q}(n+1))$. Each of its initial subsequences $(\mu_{l'})_{1\le l'\le l}$ (with $0 \le l \le k$) results in a decision point $\delta_l = (\mathfrak{q}_l, \underline{t}_l)$ having the following properties

(i) $S^{fx,\mathfrak{q}_l} \supseteq \{i \in A^{fx} | \underline{s}^\mathfrak{q}(i) < \underline{t}_l\}$

(ii) $S^{fl,\mathfrak{q}_l} = \{i \in A^{fl} | \underline{s}^\mathfrak{q}(i) < \underline{t}_l\}$

(iii) $\underline{s}^{\mathfrak{q}_l}(i) \le \underline{s}^\mathfrak{q}(i)$ for each $i \in S^{fx,\mathfrak{q}_l}$ (and hence the same for $\underline{f}$), with

(iv) equality if additionally $\underline{s}^\mathfrak{q}(i) < \underline{t}_l$ or $\underline{f}^{\mathfrak{q}_l}(i) \le \underline{t}_l$

(v) $\underline{s}^{\mathfrak{q}_l}(i) = \underline{s}^\mathfrak{q}(i)$ for each $i \in S_l^{fl}$, and the same for $\underline{f}$; from this also follows:

(vi) $\sigma^{\mathfrak{q}_l}(i) = \sigma^\mathfrak{q}(i)$ for each $i \in S_l^{fl}$

For short, $\mathfrak{q}_l$ agrees with $\mathfrak{q}|_{\underline{t}_l}$ except that there may be fixed activities scheduled in $\mathfrak{q}_l$ with $\underline{f}^{\mathfrak{q}_l}(i) > \underline{t}_l$ which are not scheduled in $\mathfrak{q}|_{\underline{t}_l}$ (since starting in $\mathfrak{q}$ at $\underline{t}_l$ or later). Thus, if (fixed) activities which start in $\mathfrak{q}$ at or later than $\underline{t}_l$ are removed from $\mathfrak{q}_l$ one gets precisely $\mathfrak{q}|_{\underline{t}_l}$. From this also follows that in $\mathfrak{q}_l$ and $\mathfrak{q}$ the same activities have been finished up to $\underline{t}_l$, i.e. $F_{\delta_l} = F_{(\mathfrak{q},\underline{t}_l)}$, which we will need for specifying the decision set below.

4. The claim is proved by inductively constructing μ_l and $\mathfrak{q}_l$ and checking that they satisfy the desired properties.

4.1 For $l = 0$, after an empty sequence, we simply have $\delta_0 = (\mathfrak{q}_0, \underline{t}_0) = (\mathfrak{p}, \underline{t})$. The properties (i)-(vi) immediately follow from $\mathfrak{q}$ being a continuation of $\mathfrak{p}$ at $\underline{t}$:
(i) $S^{fx,\mathfrak{p}} \supseteq \{i \in A^{fx} | \underline{s}^\mathfrak{q}(i) < \underline{t}\}$ since all activities not scheduled in $\mathfrak{p}$ start in $\mathfrak{q}$ at $\underline{t}$ or later
(ii) $S^{fl,\mathfrak{p}} = \{i \in A^{fl} | \underline{s}^\mathfrak{q}(i) < \underline{t}\}$ since flexible activities in $\mathfrak{p}$ are scheduled in $\mathfrak{q}$ in the same way and those not in $\mathfrak{p}$ start not before $\underline{t}$ in $\mathfrak{q}$
(iii) $\underline{s}^\mathfrak{p}(i) \le \underline{s}^\mathfrak{q}(i)$ for all $i \in S^{fx,\mathfrak{p}}$, since either i is scheduled in $\mathfrak{q}$ as in $\mathfrak{p}$ or it starts in $\mathfrak{q}$ not before $\underline{t}$ and hence later than in $\mathfrak{p}$
(iv) equality if $\underline{s}^\mathfrak{q}(i) < \underline{t}$, since in this case i is also scheduled in $\mathfrak{p}$ and, since i starts before $\underline{t}$, its schedule agrees with that in $\mathfrak{p}$
(v)+(vi) since continuation $\mathfrak{q}$ agrees with $\mathfrak{p}$ on all flexible activities scheduled in $\mathfrak{p}$

4.2 Now let $(\mu_{l'})_{1\le l'\le l}$ and $\mathfrak{q}_l$ be constructed for some $l \ge 0$ with $\mathfrak{q}_l, \underline{t}_l$ having the properties (i)-(vi) listed above in the claim.
For constructing the new schedule $\mathfrak{q}_{l+1}$ with the desired properties and the corresponding minimal alternative, we make decisions for flexible and fixed activities. Starting point for the construction is $\mathfrak{q}_l$ at $\underline{t}_l$. Besides only stating the decisions, we must also take care that they can be actually represented by a well-defined minimal alternative. For this, recall that the decision set of $\delta_l = (\mathfrak{q}_l, \underline{t}_l)$ consists of activities which have become eligible up to $\underline{t}_l$, i.e. for which all predecessors have finished not later

than $\underline{t}_l$ in $\mathfrak{q}_l$, and additionally
- for flexible activities: .. which do not have been started yet (i.e. which are not scheduled in $\mathfrak{q}_l$).
- for fixed activities: .. which do not have been finished yet (or, more precisely, which either have not been scheduled in $\mathfrak{q}_l$ or which have been scheduled there, but have not been finished up to $\underline{t}_l$).
Having this in mind, we give the decisions for flexible and fixed activities now:

4.2.2 (schedule flexible activities)
Regarding flexible activities at first, each flexible i which starts at $\underline{t}_l$ in $\mathfrak{q}$ is also started at $\underline{t}_l$ with the same state $\sigma^{\mathfrak{q}}(i)$, yielding a schedule $\mathfrak{q}'_l$. Note that such an activity i is contained in the decision set Dc_{δ_l}: Since i starts at $\underline{t}_l$ in $\mathfrak{q}$, all its predecessors finish not later than $\underline{t}_l$ in $\mathfrak{q}$ and hence in $\mathfrak{q}_l$ as well due to $F_{\delta_l} = F_{(\mathfrak{q}|_{\underline{t}_l}, \underline{t}_l)}$; since additionally i is not scheduled in $\mathfrak{q}_l$ due to (ii), it is eligible and hence contained in the decision set for δ_l.
Further, starting of these activities is feasible for the flexible part of a minimal alternative: Having started them with their states, the flexible activities running then at $[\underline{t}_l]$ (including earlier started ones) do not cause any resource conflict since they do not in (resource-)feasible $\mathfrak{q}$ where they are scheduled the same due to (v) and (vi) (for those started before) and due to the choice above (for those just started). Further the start of them with their states does not exceed the number of available components at that point since it does not in (insertion-)feasible $\mathfrak{q}$: Namely, for each combination (τ, σ), the number of (τ, σ)-components having become available up to $\underline{t}_{l+1}$ is the same as in $\mathfrak{q}$ (since the same activities as in $\mathfrak{q}$ finish up to $\underline{t}_l$ and they do at the same times; thus the same activities become eligible and hence the same components available as in $\mathfrak{q}$); and so is also the number of (τ, σ)-components consumed up to $\underline{t}_l$ the same as in $\mathfrak{p}$ (since the same flexible activities, scheduled the same, are started up to $\underline{t}_l$).
So the choice to start at $\underline{t}_l$ the same flexible activities with same states as in $\mathfrak{q}$ is feasible for the flexible part of a minimal alternative at decision point δ_l and can be specified by setting $St_l^{\sigma} := \{i \in A^{fl} | \underline{s}^{\mathfrak{q}}(i) = \underline{t}_l \wedge \sigma^{\mathfrak{q}}(i) = \sigma\}$ for $\sigma = 0, 1$ and, for the remaining flexible activities, $Dl_l^{fl} := Dc_{\delta_l}^{fl} \backslash \{i \in A^{fl} | \underline{s}^{\mathfrak{q}}(i) = \underline{t}_l\}$.

4.2.3 (schedule fixed activities)
Regarding fixed activities next, initially remove from $\mathfrak{q}'_l$ each fixed i starting strictly later than $\underline{t}_l$ in $\mathfrak{q}$. Such an i starts before $\underline{t}_l$ (since scheduled in $\mathfrak{q}_l$) and ends strictly later than $\underline{t}_l$ due to (iv). Such an activity is contained in the current decision set since its predecessors have been finished in $\mathfrak{q}_l$ up to $\underline{t}_l$ (otherwise it would not be scheduled in $\mathfrak{q}_l$) but itself finishes only after that point. Denote the set of these initially removed activities by $B_0 := \{i \in S^{\mathfrak{q}_l, fx} | \underline{s}^{\mathfrak{q}}(i) > \underline{t}_l\}$.
Now start each fixed i starting at $\underline{t}_l$ in $\mathfrak{q}$ unless it is already scheduled in $\mathfrak{q}_l$. Such an activity i is contained in the decision set since its predecessors have been finished up to $\underline{t}_l$ in $\mathfrak{q}$ and hence in $\mathfrak{q}_l$ as well whereas i is has not been started up to that point, hence being eligible at δ_l. Note that the resulting schedule $\mathfrak{q}''_l$ is identical with $\mathfrak{q}|_{\underline{t}_{l+1}}$ except that some fixed activities ending after $\underline{t}_l$ in $\mathfrak{q}_l$ may start in $\mathfrak{q}_l$ earlier than in $\mathfrak{q}$.
This schedule is resource-feasible since up to $\underline{t}_l$ it is identical with $\mathfrak{q}_l$ except for the possibly removed fixed activities, which, however, only reduce resource consumption; on the other hand, from $\underline{t}_l$ onwards, one has the following: Each activity i active in $\mathfrak{q}'_l$ beyond $\underline{t}_l$, i.e. in $[\underline{t}_l, \underline{f}^{\mathfrak{q}'_l}(i)]$, starts in $\mathfrak{q}$ not later than $\underline{t}_l$ and not earlier than in $\mathfrak{q}'_l$ whence it is active there in $[\underline{t}_l, \underline{f}^{\mathfrak{q}}(i)]$ with $\underline{f}^{\mathfrak{q}}(i) \geq \underline{f}^{\mathfrak{q}_l}(i)$. Thus the resource consumption for each time slot after $\underline{t}_l$ is in $\mathfrak{q}'_l$ is not more than in that in $\mathfrak{q}$ such that $\mathfrak{q}'_l$ is resource-feasible since $\mathfrak{q}$ is. Altogether $\mathfrak{q}'_l$ is resource-feasible as a whole.
Now schedule the fixed activities initially removed (as given by B_0) with their original start and end points again and additionally start the (remaining) eligible fixed activities which have not been started yet (since starting in $\mathfrak{q}$ later than $\underline{t}_l$). These two classes of activities form the set $B = \{i \in S^{fx, \mathfrak{q}_l} | \underline{s}^{\mathfrak{q}}(i) > \underline{t}_l\} \cup \{i \in A^{fx} \backslash S^{fx, \mathfrak{q}_l} | P_i \subseteq F_{\delta_l}\}$ which is contained in the current decision set - for B_0 as seen above, for the second class simply since they are eligible at δ_l. By scheduling these activities

(for B_0: again), one possibly gets a resource conflict from $\underline{t}_l$ onwards, with maximum exceedance at $[\underline{t}_l]$ (in this slot, since all activities in the new schedule start not later than $\underline{t}_l$ whence the resource consumption is monotonically decreasing from $[\underline{t}_l]$ onwards). If there is such a resource conflict indeed then remove some (setwise) minimal subset $C \subseteq B$ of activities which resolves the resource conflict at $[\underline{t}_l]$ (and hence for the whole schedule). Recall that removing a subset of B suffices since removing whole B yields $\mathfrak{q}'_l$ which has been shown resource-feasible above. If there is no resource conflict, set $C := \emptyset$. Now delaying $C =: Dl^{fx}_{l+1}$ we get a resource-feasible schedule $\mathfrak{q}'' =: \mathfrak{q}_{l+1}$.

Combining the choices for flexible and fixed activities we get a minimal alternative $\mu_{l+1} = (D_{l+1}, St^0) = (Dl^{fl}_{l+1} \cup Dl^{fx}_{l+1}, St^0)$ which, applied to $\mathfrak{q}_l$, yields schedule $\mathfrak{q}_{l+1}$. Since this μ_{l+1} is a feasible minimal alternative indeed, as checked above for its flexible and fixed parts, the resulting schedule $\mathfrak{q}_{l+1}$ is also feasible.

5. Thus $\mathfrak{q}_{l+1}$ is a feasible schedule which arises from $\mathfrak{q}_l$ at $\underline{t}_l$ by application of suitable minimal alternative μ_{l+1}. Now check that $\mathfrak{q}_{l+1}$ satisfies the properties listed above indeed:

(i) $S^{fx,\mathfrak{q}_{l+1}} \supseteq (S^{fx,\mathfrak{q}_l} \backslash \{i \in A^{fx} | \underline{s}^{\mathfrak{q}}(i) > \underline{t}_l\}) \cup \{i \in A^{fx} | \underline{s}^{\mathfrak{q}}(i) = \underline{t}_l\} = \{i \in A^{fx} | \underline{s}^{\mathfrak{q}}(i) \leq \underline{t}_l\} = \{i \in A^{fx} | \underline{s}^{\mathfrak{q}}(i) < \underline{t}_{l+1}\}$
(only fixed activities with $\underline{s}^{\mathfrak{q}}(i) > \underline{t}_l$ are delayed)

(ii) $S^{fl,\mathfrak{q}_{l+1}} = S^{fl,\mathfrak{q}_l} \cup \{i \in A^{fl} | \underline{s}^{\mathfrak{q}}(i) = \underline{t}_l\} = \{i \in A^{fl} | \underline{s}^{\mathfrak{q}}(i) \leq \underline{t}_l\} = \{i \in A^{fl} | \underline{s}^{\mathfrak{q}}(i) < \underline{t}_{l+1}\}$
(the $\supseteq$-direction of the last equation since start times in $\mathfrak{q}$ coincide with time points $\underline{t}_{l'}$)

(iii) Let $i \in S^{fx,\mathfrak{q}_{l+1}}$ and show $\underline{s}^{\mathfrak{q}_{l+1}}(i) \leq \underline{s}^{\mathfrak{q}}(i)$. If i was already scheduled in $\mathfrak{q}_l$ then it has not been delayed and hence it is scheduled in $\mathfrak{q}_{l+1}$ as in $\mathfrak{q}_l$ whence $\underline{s}^{\mathfrak{q}_{l+1}}(i) = \underline{s}^{\mathfrak{q}_l}(i) \leq \underline{s}^{\mathfrak{q}}(i)$ according to (iii) for $\mathfrak{q}_l$. If i was not scheduled in $\mathfrak{q}_l$ then it has been just started at $\underline{t}_l$ and further $\underline{s}^{\mathfrak{q}}(i) \geq \underline{t}_l$ according to (i) for $\mathfrak{q}_l$ whence $\underline{s}^{\mathfrak{q}_{l+1}}(i) = \underline{t}_l \leq \underline{s}^{\mathfrak{q}}(i)$. In both cases we get the desired inequality.

(iv) Assume to the contrary that $\underline{s}^{\mathfrak{q}_{l+1}}(i^*) \neq \underline{s}^{\mathfrak{q}}(i^*)$ for some $i^* \in S^{fx,\mathfrak{q}_{l+1}}$ with $\underline{s}^{\mathfrak{q}}(i^*) < \underline{t}_{l+1}$ or $\underline{f}^{\mathfrak{q}_{l+1}}(i^*) \leq \underline{t}_{l+1}$. Then, according to (iii) for $\mathfrak{q}_{l+1}$, we have $\underline{s}^{\mathfrak{q}_{l+1}}(i^*) < \underline{s}^{\mathfrak{q}}(i^*)$ for each such i^*. By combining $\mathfrak{q}_{l+1}$ and $\mathfrak{q}$ appropriately we construct a new complete schedule $\mathfrak{q}'$ being a continuation of $\mathfrak{p}$ and dominating $\mathfrak{q}$ (where those activities i^* will guarantee domination since starting earlier in $\mathfrak{q}_{l+1}$ than in $\mathfrak{q}$). But as $\mathfrak{q}$ was chosen dominant, the resulting contradiction will show that activities i^* as assumed above cannot exist, hence implying (iv) for $\mathfrak{q}_{l+1}$.
To get $\mathfrak{q}'$, schedule each activity $i \in B^{\mathfrak{q}_{l+1}} := S^{\mathfrak{q}_{l+1}} \backslash \{i \in S^{\mathfrak{q}_{l+1},fx} | \underline{s}^{\mathfrak{q}}(i) \geq \underline{t}_{l+1} \wedge \underline{f}^{\mathfrak{q}_{l+1}}(i) > \underline{t}_{l+1}\}$ according to $\mathfrak{q}_{l+1}$ and each remaining $i \in B^{\mathfrak{q}} := A \backslash B^{\mathfrak{q}_{l+1}}$ according to $\mathfrak{q}$. Note that each i^* as assumed above is contained in $B^{\mathfrak{q}_{l+1}}$ and hence scheduled according to $\mathfrak{q}_{l+1}$, and hence strictly earlier than in $\mathfrak{q}$. Furthermore the schedule $\mathfrak{q}'$ has the following properties:
- Flexible activities are scheduled in $\mathfrak{q}'$ as in $\mathfrak{q}$: If a flexible activity is contained in $B^{\mathfrak{q}_{l+1}}$ and hence scheduled in $\mathfrak{q}'$ according to $\mathfrak{q}_{l+1}$ then it is also scheduled as in $\mathfrak{q}$ due to (v) and (vi) for $\mathfrak{q}_{l+1}$ (see below; since their proof does not depend on (iv) they can be assumed here as already proven). Otherwise it is contained in $B^{\mathfrak{q}}$ and scheduled according to $\mathfrak{q}$ anyway.
- Fixed (and hence all) activities do not start later in $\mathfrak{q}'$ than in $\mathfrak{q}$: For fixed $i \in B^{\mathfrak{q}_{l+1}}$ by using (iii), for $i \in B^{\mathfrak{q}}$ anyway.
- Each $i \in B^{\mathfrak{q}}$ starts at $\underline{t}_{l+1}$ or later: If additionally $i \in S^{\mathfrak{q}_{l+1}}$ then i is fixed with $\underline{s}^{\mathfrak{q}}(i) \geq \underline{t}_{l+1}$ (and $\underline{f}^{\mathfrak{q}_{l+1}}(i) > \underline{t}_{l+1}$). If $i \notin S^{\mathfrak{q}_{l+1}}$ then $\underline{s}^{\mathfrak{q}}(i) \geq \underline{t}_{l+1}$ with (i) and (ii) for $\mathfrak{q}_{l+1}$ as already shown above.
Thus, since all activities do not start later in $\mathfrak{q}'$ than in $\mathfrak{q}$ and some actually start earlier, $\mathfrak{q}'$ dominates $\mathfrak{q}$ - if $\mathfrak{q}'$ is also a continuation of $\mathfrak{q}$, which is shown next.

(continuation)

Given that $\mathfrak{q}'$ is a feasible schedule (which will be shown afterwards), it is also a continuation of $\mathfrak{p}$ at $\underline{t}$ (which then, as just seen, dominates $\mathfrak{q}$): For this, make sure for any given activity i that it satisfies the corresponding property in the continuation definition:
If $i \in B^{\mathfrak{q}_{l+1}}$ then it is scheduled according to $\mathfrak{q}_{l+1}$; since $\mathfrak{q}_{l+1}$ arises from $\mathfrak{p}$ at $\underline{t}$ by a sequence of minimal alternatives, it is a continuation whence i satisfies the corresponding property. Indeed, if $i \in S^{\mathfrak{p}}$ then either it is scheduled in $\mathfrak{q}_{l+1}$ (and hence in $\mathfrak{q}'$) as in $\mathfrak{p}$ or (in which case it is fixed with $\underline{f}^{\mathfrak{p}}(i) > \underline{t}$) it starts in $\mathfrak{q}_{l+1}$ (and hence in $\mathfrak{q}$) at $\underline{t}$ or later instead as in $\mathfrak{p}$; on the other hand, if $i \notin S^{\mathfrak{p}}$ then it starts in $\mathfrak{q}_{l+1}$ (and hence in $\mathfrak{q}$) at $\underline{t}$ or later. Thus $i \in B^{\mathfrak{q}_{l+1}}$ satisfies the corresponding condition also for $\mathfrak{q}'$.
On the other hand, if $i \in B^{\mathfrak{q}}$ then i is scheduled according to $\mathfrak{q}$ and hence satisfies the corresponding condition since $\mathfrak{q}$ is a continuation as well.
Thus, whether $i \in B^{\mathfrak{q}_{l+1}}$ or $i \in B^{\mathfrak{q}}$, each activity satisfies the conditions necessary for $\mathfrak{q}'$ being a continuation whence $\mathfrak{q}'$ is a continuation indeed.

(feasibility)
It remains to show that $\mathfrak{q}'$ is a feasible schedule indeed:

- *precedence*: Let be given i_1, i_2 with $i_1 \in P_{i_2}$. If either both $i_1, i_2 \in B^{\mathfrak{q}_{l+1}}$ or both $i_1, i_2 \in B^{\mathfrak{q}}$ then precedence (i.e. $\underline{f}^{\mathfrak{q}'}(i_1) \leq \underline{s}^{\mathfrak{q}'}(i_2)$) follows from precedence in $\mathfrak{q}_{l+1}$ and $\mathfrak{q}$, respectively. If $i_1 \in B^{\mathfrak{q}_{l+1}}$ and $i_2 \in B^{\mathfrak{q}}$ then precedence is satisfied with $\underline{f}^{\mathfrak{q}'}(i_1) = \underline{f}^{\mathfrak{q}_{l+1}}(i_1) \leq \underline{f}^{\mathfrak{q}}(i_1) \leq \underline{s}^{\mathfrak{q}}(i_2) = \underline{s}^{\mathfrak{q}'}(i_2)$, the first inequality with (iii) for $\mathfrak{q}_{l+1}$ if i_1 is fixed and since $\mathfrak{q}'$ and $\mathfrak{q}$ agree for flexible activities if i_1 is flexible. The case $i_1 \in B^{\mathfrak{q}}, i_2 \in B^{\mathfrak{q}_{l+1}}$ does not occur since $B^{\mathfrak{q}_{l+1}}$ is closed under precedence as $S^{\mathfrak{q}_{l+1}}$ is and the activities removed to get $B^{\mathfrak{q}_{l+1}}$ finish only after $\underline{t}_{l+1}$.

- *resources*: Up to $\underline{t}_{l+1}$, schedule $\mathfrak{q}'$ is resource-feasible since there it largely agrees with $\mathfrak{q}_{l+1}$: All activities $i \in B^{\mathfrak{q}_{l+1}} \subseteq S^{\mathfrak{q}_{l+1}}$ are scheduled according to $\mathfrak{q}_{l+1}$ whereas the remaining ones in $B^{\mathfrak{q}}$ do not start before $\underline{t}_{l+1}$. Thus, up to $\underline{t}_{l+1}$, schedule $\mathfrak{q}'$ agrees with $\mathfrak{q}_{l+1}$ except that some activities may start only at that point or later and hence not consuming any resources up to $\underline{t}_{l+1}$. Therefore $\mathfrak{q}'$ is resource-feasible on $[0, \underline{t}_{l+1}]$ since $\mathfrak{q}_{l+1}$ is.
On the other hand, from $\underline{t}_{l+1}$ onwards, schedule $\mathfrak{q}'$ is resource-feasible since there it agrees with $\mathfrak{q}$ except that some fixed activities $i \in B^{\mathfrak{q}_{l+1}}$ with $\underline{f}^{\mathfrak{q}_{l+1}}(i) > \underline{t}_{l+1}$ may be scheduled and hence finish earlier in $\mathfrak{q}'$ than in $\mathfrak{q}$. Thus, beyond $\underline{t}_{l+1}$, each such i is active in $[\underline{t}_{l+1}, \underline{f}^{\mathfrak{q}'_{l+1}}(i)] \subseteq [\underline{t}_{l+1}, \underline{f}^{\mathfrak{q}_{l+1}}(i)]$, such that for each time slot $\geq \underline{t}_{l+1}$ the resource consumption in $\mathfrak{q}'$ does not exceed that in $\mathfrak{q}$.
Putting both parts together, $\mathfrak{q}'$ is resource-feasible as a whole.

- *insertion*: Show for each combination (τ, σ) that $cc^{\mathfrak{q}'}_{\underline{t}'}(\tau, \sigma) \geq 0$ for each time $\underline{t}'$. For this show that the number of (τ, σ)-components having become available up to $\underline{t}'$ is at least as large as the number of (τ, σ)-components having become consumed up to $\underline{t}'$, i.e. $cc^{\mathfrak{q}'+}_{\underline{t}'}(\tau, \sigma) \geq cc^{\mathfrak{q}'-}_{\underline{t}'}(\tau, \sigma)$.
Since all activities finish in $\mathfrak{q}'$ not later than in $\mathfrak{q}$, all activities become eligible in $\mathfrak{q}'$ not later than in $\mathfrak{q}$. Thus also the number of τ-activities with initial state σ having become eligible up to $\underline{t}'$, i.e. the number of τ-components with state σ having become up to $\underline{t}'$ is in $\mathfrak{q}'$ at least as large as in $\mathfrak{q}$, i.e. $cc^{\mathfrak{q}'+}_{\underline{t}'}(\tau, \sigma) \geq cc^{\mathfrak{q}+}_{\underline{t}'}(\tau, \sigma)$.
On the other hand, since all flexible activities are scheduled in $\mathfrak{q}'$ as in $\mathfrak{q}$, (τ, σ)-components are consumed at the same times, whence also the number of (τ, σ)-components consumed up to $\underline{t}'$ is the same, i.e. $cc^{\mathfrak{q}'-}_{\underline{t}'}(\tau, \sigma) = cc^{\mathfrak{q}-}_{\underline{t}'}(\tau, \sigma)$.
Putting both parts together, we get $cc^{\mathfrak{q}'}_{\underline{t}'}(\tau, \sigma) = cc^{\mathfrak{q}'+}_{\underline{t}'}(\tau, \sigma) - cc^{\mathfrak{q}'-}_{\underline{t}'}(\tau, \sigma) \geq cc^{\mathfrak{q}+}_{\underline{t}'}(\tau, \sigma) - cc^{\mathfrak{q}-}_{\underline{t}'}(\tau, \sigma) \geq cc^{\mathfrak{q}}_{\underline{t}'}(\tau, \sigma)$ and hence insertion-feasibility of $\mathfrak{q}'$.

Thus having shown that $\mathfrak{q}'$ is also a feasible schedule, the contradiction described above applies indeed, from which we get (iv).

(v) and (vi) hold since activities already scheduled in $\mathfrak{q}_l$ have been kept scheduled as there (and hence agreeing with $\mathfrak{q}$) and new flexible activities have been scheduled as in $\mathfrak{q}$ as well.

6. ($\underline{t}_{l+1}$ is the next decision point indeed)
Finally we have to make sure that $\underline{t}_{l+1}$ is really the next end point in $\mathfrak{q}_{l+1}$ after $\underline{t}_l$ and hence $\delta_{l+1} = (\mathfrak{q}_{l+1}, \underline{t}_{l+1})$ the next decision point after choice of minimal alternative μ_{l+1} indeed. Assume this is not the case and let be $i^* \in S^{\mathfrak{q}_{l+1}}$ with that next end point $\underline{f}^{\mathfrak{q}_{l+1}}(i^*) \neq \underline{t}_{l+1}$.
First we show that $\underline{f}^{\mathfrak{q}_{l+1}}(i^*) < \underline{t}_{l+1}$: If otherwise $\underline{f}^{\mathfrak{q}_{l+1}}(i^*) > \underline{t}_{l+1}$ then let $j^* \in A$ one of the activities with $\underline{f}^{\mathfrak{q}}(j^*) = \underline{t}_{l+1}$. Activity j^* starts not later than at $\underline{t}_l$ in $\mathfrak{q}$ whence it is scheduled in $\mathfrak{q}_{l+1}$ due to (i) or (ii) but not as in $\mathfrak{q}$ and hence it is fixed with (v) and starts earlier than in $\mathfrak{q}$ with (iii), from which we get get $\underline{f}^{\mathfrak{q}_{l+1}}(j^*) < \underline{t}_{l+1}$. However, if we had $\underline{f}^{\mathfrak{q}_{l+1}}(j^*) \leq \underline{t}_l$ then also already $\underline{f}^{\mathfrak{q}_l}(j^*) \leq \underline{t}_l$ and hence also $\underline{f}^{\mathfrak{q}}(j^*) \leq \underline{t}_l$ with (iv) for $\mathfrak{q}_l$, contradiction to $\underline{f}^{\mathfrak{q}}(j^*) = \underline{t}_{l+1}$. Thus we would have $\underline{t}_l < \underline{f}^{\mathfrak{q}_{l+1}}(j^*) \leq \underline{t}_{l+1} < \underline{f}^{\mathfrak{q}_{l+1}}(i^*)$ whence the next endpoint after $\underline{t}_l$ in $\mathfrak{q}_{l+1}$ would not be $\underline{f}^{\mathfrak{q}_{l+1}}(i^*)$, contradiction to the choice of i^*. Thus we have $\underline{f}^{\mathfrak{q}_{l+1}}(i^*) < \underline{t}_{l+1}$. However, in this case we also have $\underline{f}^{\mathfrak{q}_{l+1}}(i) = \underline{f}^{\mathfrak{q}}(i)$ due to (iv) for $\mathfrak{q}_{l+1}$ and hence $\underline{t}_l < \underline{f}^{\mathfrak{q}}(i) < \underline{t}_{l+1}$, which contradicts $\underline{f}^{\mathfrak{q}}(i) \in \{\underline{t}_0, \ldots, \underline{t}_k\}$. Thus i^* as assumed does not exist and it follows that $\underline{t}_{l+1}$ is the next end point in $\mathfrak{q}_{l+1}$ after $\underline{t}_l$ indeed. Thus $\delta_{l+1} = (\mathfrak{q}_{l+1}, \underline{t}_{l+1})$ is the decision point which results by application of chosen minimal alternative $\mu_{l+1} = (Dl_{l+1}, St^0_{l+1})$.

This completes the induction step and hence the whole proof of the theorem.

□

8.3 Proof of Theorem 2 (Dominance rule with cut vectors)

Theorem 2. *Let $\delta = (\mathfrak{p}, t)$ and $\tilde{\delta} = (\tilde{\mathfrak{p}}, \tilde{t})$ be decision points with cut vectors cv and $\tilde{cv}$ such that cv dominates $\tilde{cv}$. Further let $\Delta = (\mathfrak{q}, MS(\mathfrak{q}))$ be a decision point corresponding to a complete schedule with minimal makespan among those which are (direct or indirect) successors of δ in the branching tree with minimal alternatives. Let $\tilde{\Delta} = (\tilde{\mathfrak{q}}, MS(\tilde{\mathfrak{q}}))$ be a corresponding decision point for $\tilde{\delta}$, respectively. Then $MS(\mathfrak{q}) \leq MS(\tilde{\mathfrak{q}})$.*

Proof. Let δ, cv, $\Delta = (\mathfrak{q}, t)$ and their $\sim$ed counterparts be as above. To prove the assertion, i.e. the inequality finally given there, a continuation of $\mathfrak{p}$ is constructed by combining $\mathfrak{p}$ and $\tilde{\mathfrak{q}}$ in an appropriate manner. Essentially we will extend $\mathfrak{p}$ by scheduling the remaining activities according to $\tilde{\mathfrak{q}}$ and making some corrections to guarantee the desired properties. Since in this way $\tilde{\mathfrak{q}}$ will form the "second half" of that constructed schedule, this continuation will have makespan $MS(\tilde{\mathfrak{q}})$, and then Theorem 1 ensures that there is a sucessor $\Delta = (\mathfrak{q}, MS(\mathfrak{q}))$ of δ in the branching tree with $MS(\mathfrak{q}) \leq MS(\tilde{\mathfrak{q}})$. Correspondingly, the proof mainly consists of defining the schedule and verifying that it is well-defined, feasible and a continuation indeed.

That schedule which will be shown to be a continuation of $\mathfrak{p}$ is constructed in two steps: At first schedule each $i \in S^{\mathfrak{p}}$, except fixed i with $f^{\mathfrak{p}}(i) > \tilde{t}$, according to $\mathfrak{p}$, and the remaining activities, i.e. either $i \in S^{\mathfrak{p},fx}$ with $f^{\mathfrak{p}}(i) > \tilde{t}$ or $i \in A \backslash S^{\mathfrak{p}}$, according to $\tilde{\mathfrak{q}}$, getting a schedule $\mathfrak{q}'$. Denote these sets of activities which are scheduled according to $\mathfrak{p}$ and $\tilde{\mathfrak{q}}$ by $B^{\mathfrak{p}}$ and $B^{\tilde{\mathfrak{q}}}$, respectively, i.e. $B^{\mathfrak{p}} := S^{\mathfrak{p}} \backslash \{i \in S^{\mathfrak{p},fx} | f^{\mathfrak{p}}(i) > \tilde{t}\}$ and $B^{\tilde{\mathfrak{q}}} := A \backslash B^{\mathfrak{p}}$.
In a second step take for each type τ the $\tilde{n}^0_\tau - n^0_\tau =: k^*_\tau$ earliest scheduled activities $i \notin S^{\mathfrak{p}}$ of type τ which are processed as repair tasks and process them as dummy tasks instead. More precisely, if $i^\tau_1, \ldots, i^\tau_{k_\tau}$ is an order of the set $\{i \in A_\tau \backslash S^{\mathfrak{p}} | \sigma^{\mathfrak{q}'}(i) = \sigma^{\tilde{\mathfrak{q}}}(i) = 1\}$ according to their increasing start times in $\mathfrak{q}'$ (and $\tilde{\mathfrak{q}}$), i.e. with $s^{\mathfrak{q}'}(i^\tau_1) \leq s^{\mathfrak{q}'}(i^\tau_2) \leq \ldots \leq s^{\mathfrak{q}'}(i^\tau_k)$, then change for $i^\tau_1, \ldots, i^\tau_{k^*_\tau}$ the state from 1 to 0, i.e. change for each such activity i its state to $\sigma(i) := 0$ and its end time to $f(i) := s^{\mathfrak{q}'}(i)$ (the order and correspondingly also the choice of the activities may be not unique due to equality in start times, but this does not matter, any choice will do).

This yields the final schedule $\mathfrak{q}''$. The different treatment of activities $i \in S^{\mathfrak{p}}$ with $f^{\mathfrak{p}}(i) > \tilde{t}$ in the first step will be due to compatibility reasons by combining $\mathfrak{p}$ and $\tilde{\mathfrak{q}}$, the second step is necessary since in general $\mathfrak{q}'$ still processes more activities as repair tasks than defective components available, such that one has make this correction to ensure insertion feasibility.

At first note the following properties of this schedule:
- activities $i \notin S^{\mathfrak{p}}$ start in $\mathfrak{q}''$ at $\tilde{t}$ or later: Indeed, each $i \notin S^{\mathfrak{p}}$ is contained in $B^{\tilde{\mathfrak{q}}}$ but not in $S^{\tilde{\mathfrak{p}}}(\subseteq S^{\mathfrak{p}})$ whence it starts in $\tilde{\mathfrak{q}}$ (and hence also in $\mathfrak{q}''$) at $\tilde{t}$ or later since $\tilde{\mathfrak{q}}$ is a continuation of $\tilde{\mathfrak{p}}$.
- As a consequence, up to $\tilde{t}$ schedule $\mathfrak{q}''$ coincides with $\mathfrak{p}$ except that $i \in S^{\mathfrak{p},fx}$ with $f^{\mathfrak{p}}(i) > \tilde{t}$ are scheduled according to $\tilde{\mathfrak{q}}$ and hence starting there possibly later at $s^{\tilde{q}}(i) \geq s^{\mathfrak{p}}(i)$ (and possibly only at or after $\tilde{t}$).
- From $\tilde{t}$ onwards, schedule $\mathfrak{q}''$ coincides with $\tilde{\mathfrak{q}}$ except that a) some flexible activities are processed in state 0 instead 1 (with the same start time, but reduced end time) and that b) (possibly) some flexible activities are scheduled according to $\mathfrak{p}$. For the latter case from $\tilde{t}$ onwards such an activity i is active on $[\tilde{t}, f^{\mathfrak{p}}(i)] \subseteq [\tilde{t}, f^{\tilde{\mathfrak{p}}}(i)]$, since due to dominance we have $i \in S^{\mathfrak{p},fl} = S^{\tilde{\mathfrak{p}},fl}$ (hence also starting up to $\tilde{t}$ in both cases) and $\tilde{t} \leq f^{\mathfrak{p}}(i) = f^{\geq t}(i) \leq \tilde{f}^{\geq \tilde{t}}(i) = f^{\tilde{p}}(i)$.

In the rest of the proof it is shown that the constructed schedule $\mathfrak{q}''$ is a continuation of $\mathfrak{p}$ at t indeed. More precisely not only a continuation but also that it is well-defined and (this will make most work) feasible as well. We start with continuation :

(continuation)
Given that $\mathfrak{q}''$ is a well-defined and feasible schedule (which we will show below), it is also a continuation of $\mathfrak{p}$ at t: Each activity $i \in S^{\mathfrak{p}}$ which is flexible or fixed with $f^{\mathfrak{p}}(i) \leq t$ is scheduled as in $\mathfrak{p}$. The remaining activities in $S^{\mathfrak{p}}$, fixed activities with $f^{\mathfrak{p}}(i) > t$, are either scheduled as in $\mathfrak{p}$ as well (if $f^{\mathfrak{p}}(i) \leq \tilde{t}$) or start at t or later (even at $\tilde{t}$ or later, for $\tilde{t} \geq t$ since cv dominates $\tilde{cv}$; holding for $f^{\mathfrak{p}}(i) > \tilde{t}$). Activities $i \notin S^{\mathfrak{p}}$ start at t (even $\tilde{t}$) or later. Altogether $\mathfrak{q}''$ satisfies for each actvitiy i the condition to be a continuation of $\mathfrak{p}$ at t.
Note further that $MS(\mathfrak{q}'') = MS(\tilde{\mathfrak{q}})$.

(well-defined)
Next check that $\mathfrak{q}''$ is well-defined indeed: While the first step ($\mathfrak{q}'$) is unproblematic, in the second step one has to make sure that sufficiently many suitable activities with state 1 exist at all. Here note at first that $\tilde{n}^0_\tau - n^0_\tau \geq 0$ since cv dominates $\tilde{cv}$. Further, in $A \backslash S^{\tilde{\mathfrak{p}}}$ there are at least $\tilde{n}^0_\tau - n^0_\tau$ activities of type τ which are scheduled with state 1 in $\tilde{q}$ indeed: For this, denote by n^1_τ, correspondingly to n^0_τ, the number of τ-activities contained in $\mathfrak{p}$ and processed there in state 1 (and analogously for $\tilde{n}^1_\tau$). Then we have $n^0_\tau + n^1_\tau = |S^{\mathfrak{p}} \cap A_\tau| = |S^{\tilde{\mathfrak{p}}} \cap A_\tau| = \tilde{n}^0_\tau + \tilde{n}^1_\tau$ (middle equation because of $S^{\mathfrak{p},fl} = S^{\tilde{\mathfrak{p}},fl}$ due to cv dominating $\tilde{cv}$) and hence $\tilde{n}^0_\tau - n^0_\tau = n^1_\tau - \tilde{n}^1_\tau$. Further let $\tilde{m}^1_\tau$ be the number of τ-activities $i \in A \backslash S^{\tilde{\mathfrak{p}}}$ which are processed in state 1, i.e. as repair tasks in $\tilde{\mathfrak{q}}$ - those activities, from which $\tilde{n}^0_\tau - n^0_\tau$ are chosen to be processed in state 0 - and correspondingly m^1_τ the number of τ-activities $i \in A \backslash S^{\mathfrak{p}}$ which have to be processed in state 1 in any continuation of $\mathfrak{p}$. Then it holds $n^1_\tau + m^1_\tau = |A^1_\tau| = |\{i \in A_\tau | \sigma_1(i) = 1\}| = \tilde{n}^1_\tau + \tilde{m}^1_\tau$ and hence $n^1_\tau - \tilde{n}^1_\tau = \tilde{m}^1_\tau - m^1_\tau$, from which one gets $0 \leq \tilde{n}^0_\tau - n^0_\tau = \tilde{m}^1_\tau - m^1_\tau \leq \tilde{m}^1_\tau$ with the previous equation, which guarantees that the second step can be accomplished as described. Thus $\mathfrak{q}''$ is well-defined indeed.

(feasible)
Next we have to verify that $\mathfrak{q}''$ is a feasible schedule:

- *precedence*: Let $i_1, i_2 \in A$ with $i_1 \in P_{i_2}$. Regard the following cases:
(1) If both $i_1, i_2 \in B^{\mathfrak{p}}$, then precedence, i.e. $f^{\mathfrak{q}''}(i) \leq s^{\mathfrak{q}''}(i)$, is satisfied since so it is also in $\mathfrak{p}$.
(2) If $i_1, i_2 \in B^{\tilde{\mathfrak{q}}}$ then precedence holds with $f^{\mathfrak{q}''}(i_1) \leq f^{\tilde{\mathfrak{q}}}(i_1) \leq s^{\tilde{\mathfrak{q}}}(i_2) = s^{\mathfrak{q}''}(i_2)$ (where, by the way, the first inequality is strict if i_1 is one of the flexible activities the state of which has been changed in the second step of the construction).

(3) If $i_1 \in B^{\mathfrak{p}}$, $i_2 \in B^{\tilde{\mathfrak{q}}}$ then i_1 ends according to $\mathfrak{p}$ and i_2 starts according to $\tilde{\mathfrak{q}}$ (since the change in the second step only affects *end* times of activities in $B^{\tilde{\mathfrak{q}}}$, it needs not to be taken account for this case). Thus we have to show $f^{\mathfrak{p}}(i_1) \leq s^{\tilde{\mathfrak{q}}}(i_2)$. If $f^{\mathfrak{p}}(i_1) \leq \tilde{t} \leq s^{\tilde{\mathfrak{q}}}(i_2)$ then there is nothing to do. So let $f^{\mathfrak{p}}(i_1) > \tilde{t}$ or $s^{\tilde{\mathfrak{q}}}(i_2) < \tilde{t}$. In the first case i_1 is not fixed due to definition of $B^{\mathfrak{p}}$ and hence flexible. Then $i_1 \in S^{\mathfrak{p},fl} = S^{\tilde{\mathfrak{p}},fl}$ and $\tilde{t} \leq f^{\mathfrak{p}}(i_1) = f^{\geq t}(i_1) \leq \tilde{f}^{\geq \tilde{t}}(i_1) = f^{\tilde{\mathfrak{p}}}(i_1) = f^{\tilde{\mathfrak{q}}}(i_1) \leq s^{\tilde{\mathfrak{q}}}(i_2)$ yields precedence. In the second case, $s^{\tilde{\mathfrak{q}}}(i_2) < \tilde{t}$, assuming $i_2 \notin S^{\mathfrak{p}} \supseteq S^{\tilde{p}}$ would give $s^{\tilde{q}}(i_2) \geq \tilde{t}$ as $\tilde{q}$ is a continuation of $\tilde{p}$, contradiction. Thus one has $i_2 \in S^{\mathfrak{p}} \cap B^{\tilde{\mathfrak{q}}}$ and hence $i_2 \in S^{\mathfrak{p},fx}$ with $f^{\mathfrak{p}}(i_2) > \tilde{t}$, which yields $\tilde{t} < f^{\mathfrak{p}}(i_2) = f^{\geq t}(i_2) \leq \tilde{f}^{\geq \tilde{t}}(i_2) = f^{\tilde{\mathfrak{p}}}(i_2) = f^{\tilde{\mathfrak{q}}}(i_2)$, from which we get $s^{\mathfrak{p}}(i_2) \leq s^{\tilde{\mathfrak{q}}}(i_2)$. Then $f^{\mathfrak{p}}(i_1) \leq s^{\mathfrak{p}}(i_2)$ yields precedence also for the second case.
(4) The case $i_1 \in B^{\tilde{\mathfrak{q}}}$, $i_2 \in B^{\mathfrak{p}}$ does not occur since $B^{\mathfrak{p}}$ is closed under precedence as $S^{\mathfrak{p}}$ is and the removed fixed activities have no successors in $\mathfrak{p}$.

- *duration*: An activitity scheduled in $\mathfrak{q}''$ according to $\mathfrak{p}$ or $\tilde{\mathfrak{q}}$ gets the right duration from these schedules. If it is one of those flexible activities with state changed to 0 in the second step, it has duration 0 consistent with appropriately changed $f^{\mathfrak{q}''}(i) = s^{\mathfrak{q}''}(i)$ and $\sigma^{\mathfrak{q}''}(i) = 0$. Thus $\mathfrak{q}''$ is feasible w.r.t. duration.

- *resources*: Recall that until $\tilde{t}$, schedule $\mathfrak{q}''$ agrees with $\mathfrak{p}$ except that $i \in S^{\mathfrak{p},fx}$ with $f^{\mathfrak{p}}(i) > \tilde{t}$ are scheduled as in $\tilde{\mathfrak{q}}$ and hence start at $s^{\tilde{\mathfrak{q}}}(i) \geq s^{\mathfrak{p}}(i)$ (and possibly after $\tilde{t}$). Thus, up to $\tilde{t}$, such an i is active on $[s^{\tilde{\mathfrak{q}}}(i), \tilde{t}]$ instead on the same or larger interval $[s^{\mathfrak{p}}(i), \tilde{t}]$ (and not active at all if $s^{\tilde{\mathfrak{q}}}(i) > \tilde{t}$), whence the resource consumption for each time slot is not higher than in $\mathfrak{p}$. Thus $\mathfrak{q}''$ is resource-feasible up to $\tilde{t}$ since $\mathfrak{p}$ is. From $\tilde{t}$ onwards, schedule $\mathfrak{q}''$ coincides with $\tilde{\mathfrak{q}}$ except that some (if any at all) flexible activities are processed in state 0 instead state 1 (which only reduces resource consumption) and some activities may be scheduled according to $\mathfrak{p}$ and finishing after $\tilde{t}$. Regarding such an activity $i \in S^{\mathfrak{p}}$, it must be flexible since a fixed one with $f^{\mathfrak{p}}(i) > \tilde{t}$ is already scheduled according to $\tilde{\mathfrak{q}}$. Then $i \in S^{\mathfrak{p},fl} = S^{\tilde{\mathfrak{p}},fl}$ and we have $\tilde{t} < f^{\mathfrak{p}}(i) = f^{\geq t}(i) \leq \tilde{f}^{\geq \tilde{t}}(i) = f^{\tilde{\mathfrak{p}}}(i) = f^{\tilde{\mathfrak{q}}}(i)$ whence i is active on $[\tilde{t}, f^{\mathfrak{p}}(i)]$ instead on the same or larger interval $[\tilde{t}, f^{\tilde{\mathfrak{q}}}(i)]$. It follows that $\mathfrak{q}''$ is resource-feasible beyond $\tilde{t}$ since $\tilde{\mathfrak{q}}$ is, whence $\mathfrak{q}''$ is resource-feasible overall.

- *insertion*: To show that $\mathfrak{q}''$ is insertion-feasible, we have to give an extended schedule $\underline{\mathfrak{q}}''$ on marginal time steps which, projected to normal time units, agrees with $\mathfrak{q}''$ and which satisfies the insertion condition introduced earlier. As $\mathfrak{p}$ and $\tilde{\mathfrak{q}}$ are insertion-feasible, there are marginal extensions $\underline{\mathfrak{p}}$ and $\underline{\tilde{\mathfrak{q}}}$ which satisfy the insertion condition (and the other feasibility conditions as well). However, simply combining any two such extensions or, rather, simply taking their marginal parts to extend $\mathfrak{q}''$ would generally cause conflicts w.r.t. precedence, resources or insertion on the marginal level. Thus we have to adapt initially chosen extensions by appropriate shifts in marginal steps to get suitable extensions from which the marginal parts can be taken to extend $\mathfrak{q}''$ to a feasible $\underline{\mathfrak{q}}''$.
So choose at first $\underline{\mathfrak{p}}$ and $\underline{\tilde{\mathfrak{q}}}$ dominant in an analogous sense as at the beginning of the proof of Theorem 1, i.e. $\underline{\mathfrak{p}}$ has the property that there is no other extension $\underline{\mathfrak{p}}'$ of $\mathfrak{p}$ with $\underline{s}^{\underline{\mathfrak{p}}'}(i) \leq \underline{s}^{\underline{\mathfrak{p}}}(i)$ for all $i \in S^{\mathfrak{p}}$ and $<$ for at least one i (i.e. the vector of the marginal parts is chosen minimal).
The refinement $\underline{\mathfrak{q}}''$ is defined in several steps: At first, simply schedule each $i \in B^{\mathfrak{p}}$ according to $\underline{\mathfrak{p}}$ and each $i \in B^{\tilde{\mathfrak{q}}}$ according to $\underline{\tilde{\mathfrak{q}}}$, getting a schedule $\underline{\mathfrak{q}}^1$. Then, corresponding to the second step in the construction of $\mathfrak{q}''$, change the state of chosen flexible activities from 1 to 0 and set $\underline{f}(i) := \underline{s}^{\underline{\mathfrak{q}}^1}(i) + \underline{1}$, getting a schedule $\underline{\mathfrak{q}}^2$. While, as seen above, precedence (and resources) is satisfied in the projection to normal time units, it may be violated on the level of marginal units due to simply appending the schedules in this way.
To fix this, set $mp^* = max\{mp(\underline{f}^{\underline{\mathfrak{q}}^2}(i) | i \in B^{\mathfrak{p}})\} + 1$ and add a constant, sufficient large number to all current start and end times for activities $i \in B^{\tilde{\mathfrak{q}}}$ such that all marginal parts of start and end times of $B^{\tilde{\mathfrak{q}}}$ are larger than mp^*, whence we get
$mp(\underline{s}(i_1)) \leq mp(\underline{f}(i_1)) < mp^* < mp(\underline{s}(i_2)) \leq mp(\underline{f}(i_2))$ for all $i_1 \in B^{\mathfrak{p}}$ and $i_2 \in B^{\tilde{\mathfrak{q}}}$

getting a schedule $\underline{\mathfrak{q}}^3$. Set further $\underline{\tilde{t}} := (\tilde{t}, mp^*)$.
For $\underline{\mathfrak{q}}^3$ it may still be that the increasing sequence of start times $s^{\tilde{\mathfrak{q}}}(i_1^\tau) \leq \ldots \leq s^{\tilde{\mathfrak{q}}}(i_{k_\tau}^\tau)$ for the activity sequence from the second step in the construction does not translate to an increasing sequence of the $\underline{s}^{\mathfrak{q}^3}$-values as well, namely where we have equality of $s^{\tilde{q}}$-values. In this case recall that k_τ^* is the last index in the second step for which state 1 is changed to 0 for type τ, and then let $k_\tau^- \leq k_\tau^* \leq k_\tau^+$ be defined by $s^{\tilde{\mathfrak{q}}}(i_{k_\tau^- - 1}) < s^{\tilde{\mathfrak{q}}}(i_{k_\tau^-}) = \ldots = s^{\tilde{\mathfrak{q}}}(i_{k_\tau^*}) = \ldots = s^{\tilde{\mathfrak{q}}}(i_{k_\tau^+}) < s^{\tilde{\mathfrak{q}}}(i_{k_\tau^+ + 1})$ and add a further constant number of marginal time steps to start and end times of $i_{k_\tau^*+1}, \ldots, i_{k_\tau^+}$ and their successors such that there is some $\underline{t}_\tau^*$ with $\underline{s}^q(i') < \underline{t}_\tau^* < \underline{s}^q(i'')$ for each $i' \leq k_\tau^*$ and $i'' > k_\tau^*$.
Renumbering further $i_1^\tau, \ldots i_{k_\tau^*}^\tau$ and $i_{k^*+1}^\tau, \ldots, i_k^\tau$ appropriately - more precisely renumbering among i_j^τ's with same $s^{\mathfrak{q}''}$-value - gives a sequence which also satisfies $\underline{s}(i_1^\tau) \leq \ldots \leq \underline{s}(i_{k_\tau}^\tau)$. Note that this renumbering does not cause any conflict with earlier decisions since the actual order of activities with same start value in $\mathfrak{q}''$ has not been used before.
This yields the final schedule $\underline{\mathfrak{q}}^4 =: \underline{\mathfrak{q}}''$, which is the desired refinement $\underline{\mathfrak{q}}''$ of $\mathfrak{q}''$ and feasible w.r.t. precedence, duration and resources. It remains to show that it is also insertion-feasible.

For insertion-feasibility one has to show that for all combinations (τ, σ) and all $\underline{t}'$
$cc_{\underline{t}'}^{\mathfrak{q}''}(\tau, \sigma) = |\{i \in A_\tau | \sigma_0(i) = \sigma \wedge \underline{el}^{\mathfrak{q}''}(i) \leq \underline{t}'\}| - |\{i \in A_\tau | \sigma^{\mathfrak{q}''}(i) = \sigma \wedge \underline{s}^{\mathfrak{q}''}(i) \leq \underline{t}'\}| \geq 0$
Denoting as earlier the first term by $cc_{\underline{t}'}^{\mathfrak{q}''+}(\tau, \sigma)$ and the second one (without negative sign) by $cc_{\underline{t}'}^{\mathfrak{q}''-}(\tau, \sigma)$, it suffices to show that $cc_{\underline{t}'}^{\mathfrak{q}''+}(\tau, \sigma) \geq cc_{\underline{t}'}^{\mathfrak{q}''-}(\tau, \sigma)$.
1. $\underline{t}' \leq \underline{\tilde{t}}$. At first regard the case $\underline{t}' \leq \underline{\tilde{t}}$, for which we can establish even equality $cc_{\underline{t}'}^{\mathfrak{q}''}(\tau, \sigma) = cc_{\underline{t}'}^{\mathfrak{p}}(\tau, \sigma)$ where the r.h.s. is nonnegative since $\underline{\mathfrak{p}}$ is (insertion-)feasible. More precisely, we will show that they agree in both terms, i.e. $cc_{\underline{t}'}^{\mathfrak{q}''+}(\tau, \sigma) = cc_{\underline{t}'}^{\mathfrak{p}+}(\tau, \sigma)$ and $cc_{\underline{t}'}^{\mathfrak{q}''-}(\tau, \sigma) = cc_{\underline{t}'}^{\mathfrak{p}-}(\tau, \sigma)$. These equalities hold since up to $\underline{\tilde{t}}$ the schedules $\underline{\mathfrak{q}}''$ and $\underline{\mathfrak{p}}$ largely agree, without their differences having any effect on the corresponding terms. Indeed, recall that up to $\underline{\tilde{t}}$, schedules $\underline{\mathfrak{p}}$ and $\underline{\mathfrak{q}}''$ agree except for $i \in S^{\mathfrak{p}, fx}$ with $f^{\mathfrak{p}}(i) > \tilde{t}$, which are scheduled in $\underline{\mathfrak{q}}''$ according to $\underline{\tilde{\mathfrak{q}}}$, hence starting in $\underline{\mathfrak{q}}''$ later than in $\underline{\mathfrak{p}}$ (and possibly only after $\underline{\tilde{t}}$, in which case the corresponding activity is not scheduled in $\underline{\mathfrak{q}}''$ up to $\underline{\tilde{t}}$). Since these exceptional activities finish only after $\underline{\tilde{t}}$, all activities ending up to $\underline{\tilde{t}}$ are the same in $\underline{\mathfrak{p}}$ and $\underline{\mathfrak{q}}''$ and they end at the same times in both schedules. Thus also the activities which become eligible up to $\underline{\tilde{t}}$ are the same in both schedules and it holds $\underline{el}^{\mathfrak{p}}(i) = \underline{el}^{\mathfrak{q}''}(i)$ for each of them. This is especially true for flexible activities whence we get
$cc_{\underline{t}'}^{\mathfrak{q}''+}(\tau, \sigma) = |\{i \in A_\tau | \sigma_0(i) = \sigma \wedge \underline{el}^{\mathfrak{q}''}(i) \leq \underline{t}'\}| = |\{i \in A_\tau | \sigma_0(i) = \sigma \wedge \underline{el}^{\mathfrak{p}}(i) \leq \underline{t}'\}| = cc_{\underline{t}'}^{\mathfrak{p}+}(\tau, \sigma)$.
On the other hand, since the exceptional activities are fixed, in $\underline{\mathfrak{p}}$ and $\underline{\mathfrak{q}}''$ the same flexible activities start up to $\underline{\tilde{t}}$, and they do at the same times. Thus we have
$cc_{\underline{t}'}^{\mathfrak{q}''-}(\tau, \sigma) = |\{i \in A_\tau | \sigma^{\mathfrak{q}''}(i) = \sigma \wedge \underline{s}^{\mathfrak{q}''}(i) \leq \underline{t}'\}| = |\{i \in A_\tau | \sigma^{\mathfrak{q}''}(i) = \sigma \wedge \underline{s}^{\mathfrak{p}}(i) \leq \underline{t}'\}| = cc_{\underline{t}'}^{\mathfrak{p}-}(\tau, \sigma)$.
Altogether we have equality in the positive and negative part and hence $cc_{\underline{t}'}^{\mathfrak{q}''}(\tau, \sigma) = cc_{\underline{t}'}^{\mathfrak{p}}(\tau, \sigma) \geq 0$ for all $\underline{t}' \leq \underline{\tilde{t}}$.

2. Now let regard $\underline{t}' > \underline{\tilde{t}}$.
2.1 At first we show $cc_{\underline{t}'}^{\mathfrak{q}''+}(\tau, \sigma) \geq cc_{\underline{t}'}^{\tilde{\mathfrak{q}}+}(\tau, \sigma)$. For this it suffices to show that for each $i \in A_\tau$ with $\sigma_0(i) = \sigma$ and $\underline{el}^{\tilde{\mathfrak{q}}}(i) \leq \underline{t}'$ it also holds $\underline{el}^{\mathfrak{q}''}(i) \leq \underline{t}'$. So let be given $i \in A_\tau$ with $\sigma_0(i) = \sigma$ and $\underline{el}^{\tilde{\mathfrak{q}}}(i) \leq \underline{t}'$. Further assume, towards a contradiction, that $\underline{el}^{\mathfrak{q}''}(i) > \underline{t}'$. Then there is a predecessor $j \in P_i$ with $\underline{f}^{\mathfrak{q}''}(j) > \underline{t}'$ but $\underline{f}^{\tilde{\mathfrak{q}}}(j) \leq \underline{t}'$, and hence with $\underline{f}^{\mathfrak{q}''}(j) > \underline{f}^{\tilde{\mathfrak{q}}}(j)$. Then $j \notin B^{\tilde{\mathfrak{q}}}$ since neither it is scheduled in $\underline{\mathfrak{q}}''$ according to $\underline{\tilde{\mathfrak{q}}}$ nor it is one of those flexible activities which got state 0 in the second step, as then its end time would have been reduced and $\underline{f}^{\mathfrak{q}''}(j) < \underline{f}^{\tilde{\mathfrak{q}}}(j)$. Thus $j \in B^{\mathfrak{p}}$ and hence $\underline{f}^{\mathfrak{p}}(j) = \underline{f}^{\mathfrak{q}''}(j) > \underline{f}^{\tilde{\mathfrak{q}}}(j)$. Since the value on the l.h.s. is at least $\underline{t}' \geq \underline{\tilde{t}}$, one has $f^{\mathfrak{p}}(j) \geq \tilde{t}$

and hence $f^{\mathfrak{p}}(j) = f^{\geq t}(j) \leq \tilde{f}^{\geq \tilde{t}}(j) = f^{\tilde{\mathfrak{q}}}(j)$. But then $\underline{f}^{\mathfrak{p}}(j) > \underline{f}^{\tilde{\mathfrak{q}}}(j)$ yields $f^{\mathfrak{p}}(j) = f^{\tilde{\mathfrak{q}}}(j)$ and $mp(\underline{f}^{\mathfrak{q}''}(j)) = mp(\underline{f}^{\mathfrak{p}}(j)) < mp^* < mp(\underline{f}^{\tilde{\mathfrak{q}}}(j))$ for the marginal parts and hence $\underline{f}^{\mathfrak{p}}(j) < \underline{f}^{\tilde{\mathfrak{q}}}(j)$, contradiction. Thus predecessor j as assumed does not exist whence $\underline{el}^{\mathfrak{q}''}(i) \leq \underline{t}'$.

2.2 cc^-

2.2.1 Next we show $cc_{\underline{t}'}^{\mathfrak{q}''-}(\tau,\sigma) \leq cc_{\underline{t}'}^{\tilde{\mathfrak{q}}-}(\tau,\sigma)$ for $\sigma = 0$.
Compared with $\underline{\tilde{\mathfrak{q}}}$, in $\underline{\mathfrak{q}}''$ up to $\underline{\tilde{t}}$ a number of $\tilde{n}_\tau^0 - n_\tau^0$ *less* τ-activities have been started in state 0 (intact τ-components have been consumed), i.e. $cc_{\underline{\tilde{t}}}^{\mathfrak{q}''}(\tau,0) = cc_{\underline{\tilde{t}}}^{\tilde{\mathfrak{q}}}(\tau,0) - \tilde{n}_\tau^0 - n_\tau^0$. Since the next $\tilde{n}_\tau^0 - n_\tau^0 = k_\tau^*$ τ-activities which are processed in state 1 in $\underline{\tilde{\mathfrak{q}}}$ are processed in state 0 in $\underline{\mathfrak{q}}''$, the difference between both cc^--values decreases to 0 and hence equality at $\underline{t}' = \underline{t}_\tau^*$ whence we have $cc_{\underline{t}'}^{\mathfrak{q}''-}(\tau,0) \leq cc_{\underline{t}'}^{\tilde{\mathfrak{q}}-}(\tau,0)$ for all $\underline{t}' \leq \underline{t}_\tau^*$. Since after $\underline{t}_\tau^*$ a τ-activity is processed in state 0 in $\underline{\mathfrak{q}}''$ iff it is processed in state 0 in $\underline{\tilde{\mathfrak{q}}}$, the equality is maintained, i.e. it holds $cc_{\underline{t}'}^{\mathfrak{q}''-}(\tau,0) = cc_{\underline{t}'}^{\tilde{\mathfrak{q}}-}(\tau,0)$ for all $\underline{t}' > \underline{t}_\tau^*$.
Thus we have $cc_{\underline{t}'}^{\mathfrak{q}''-}(\tau,0) \leq cc_{\underline{t}'}^{\tilde{\mathfrak{q}}-}(\tau,0)$ for all $\underline{t}' \geq \underline{\tilde{t}}$.

2.2.2 To establish $cc_{\underline{t}'}^{q''}(\tau,\sigma) \geq 0$ for all cases, it would now suffice to finally show $cc_{\underline{t}'}^{q''-}(\tau,1) \leq cc_{\underline{t}'}^{\tilde{\mathfrak{q}}-}(\tau,1)$ for all $\underline{t}' > \underline{\tilde{t}}$; however, in general this inequality does not hold since up to $\underline{\tilde{t}}$ schedule $\underline{\mathfrak{q}}''$ has consumed more $(\tau,1)$-components than $\underline{\tilde{\mathfrak{q}}}$. Instead, the remaining case can be also covered by showing $cc_{\underline{t}'}^{q''}(\tau,1) \geq 0$ for $\underline{t}' > \underline{\tilde{t}}$ directly:
Compared with $\underline{\tilde{\mathfrak{q}}}$, in $\underline{\mathfrak{q}}''$ up to $\underline{\tilde{t}}$ a number of $n_\tau^1 - \tilde{n}_\tau^1 (= \tilde{n}_\tau^0 - n_\tau^0)$ *more* τ-activities have been started in state 1. However, since the first $n_\tau^1 - \tilde{n}_\tau^1$ τ-activities which are started in state 1 after $\underline{\tilde{t}}$ in $\underline{\tilde{\mathfrak{q}}}$ are processed in $\underline{\mathfrak{q}}''$ in state 0, this difference decreases and becomes equality at $\underline{t}' = \underline{s}^{\mathfrak{q}''}(i_{k_\tau^*})$. Thus we have $cc_{\underline{\tilde{t}}}^{\tilde{\mathfrak{q}}-}(\tau,1) + (n_\tau^1 - \tilde{n}_\tau^1) = cc_{\underline{\tilde{t}}}^{\mathfrak{q}''-}(\tau,1) = \ldots = cc_{\underline{t}_\tau^*}^{\mathfrak{q}''-}(\tau,1) = cc_{\underline{t}_\tau^*}^{\tilde{\mathfrak{q}}-}(\tau,1)$. Since further $cc_{\underline{\tilde{t}}}^{\mathfrak{q}''}(\tau,1) = cc_{\underline{\tilde{t}}}^{\mathfrak{q}''+}(\tau,1) - cc_{\underline{\tilde{t}}}^{\mathfrak{q}''}(\tau,1) \geq 0$ and $cc_{\underline{t}'}^{\mathfrak{q}''+}(\tau,1)$ is monotonically increasing for increasing $\underline{t}'$, we get $cc_{\underline{t}'}^{\mathfrak{q}''}(\tau,1) \geq 0$ for $\underline{t}' = \underline{\tilde{t}}, \ldots, \underline{t}_\tau^*$. Regarding $\underline{t}' > \underline{t}_\tau^*$, note that each activity of type τ which starts after $\underline{t}_\tau^*$ is processed in state 1 in $\underline{\mathfrak{q}}''$ iff it is processed in state 1 in $\underline{\tilde{\mathfrak{q}}}$. Thus equality of the cc^--parts is maintained and we have $cc_{\underline{t}'}^{\mathfrak{q}''-}(\tau,1) = cc_{\underline{t}'}^{\tilde{\mathfrak{q}}-}(\tau,1)$ for all $\underline{t}' > \underline{t}_\tau^*$, from which we get $cc_{\underline{t}'}^{\mathfrak{q}''}(\tau,1) = cc_{\underline{t}'}^{\mathfrak{q}''+}(\tau,1) - cc_{\underline{t}'}^{\mathfrak{q}''-}(\tau,1) \geq cc_{\underline{t}'}^{\tilde{\mathfrak{q}}+}(\tau,1) - cc_{\underline{t}'}^{\tilde{\mathfrak{q}}-}(\tau,1) = cc_{\underline{t}'}^{\tilde{\mathfrak{q}}}(\tau,1) \geq 0$

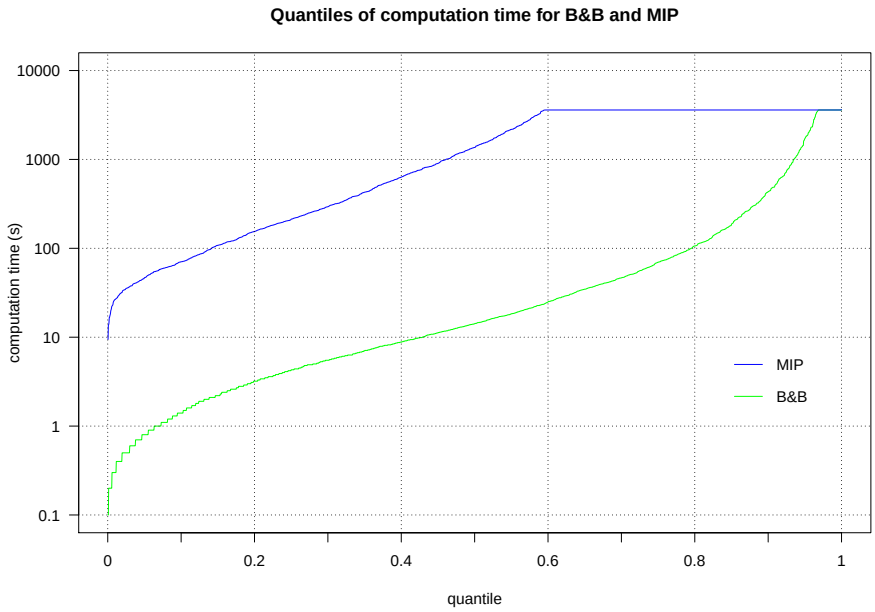
Putting all cases together, we get $cc_{\underline{t}'}^{\mathfrak{q}}(\tau,\sigma) \geq 0$ for all combinations (τ,σ) and times $\underline{t}'$, and hence insertion feasibility of $\underline{\mathfrak{q}}''$ and hence of $\mathfrak{q}''$ as well.

Altogether $\mathfrak{q}''$ is feasible and hence a continuation of $\mathfrak{p}$ indeed, which completes the proof.

□

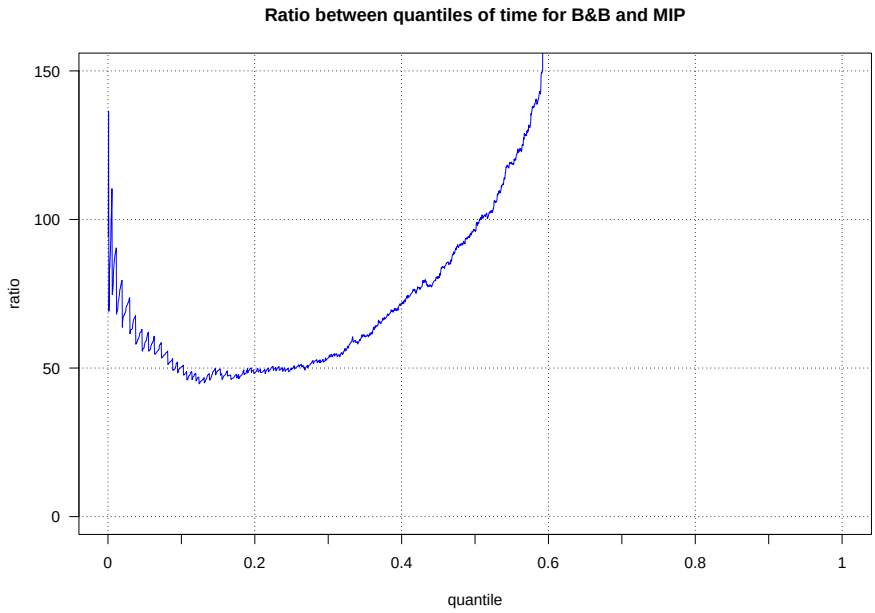
8.4 Supplements to Computational Study

8.4.1 Quantiles of computation time in logarithmic scale:

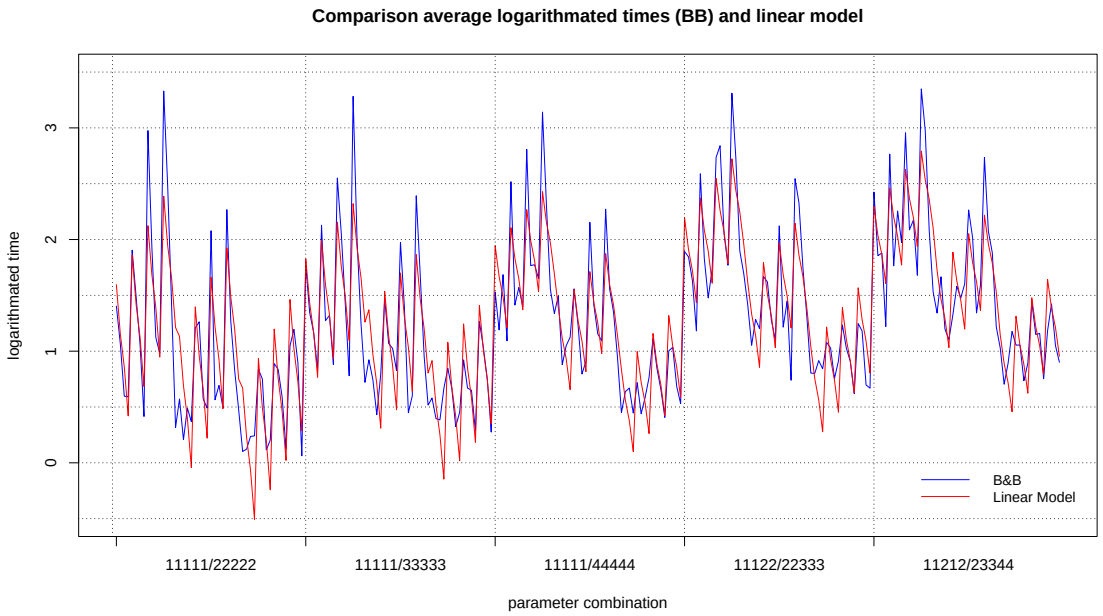


Both curves have similar shape (at least up to the time limit), but are distinguished by a roughly constant difference, which, since in logarithmic scale, corresponds to a quotient of at least one-and-a-half magnitudes of order. In fact this quotient is not that constant but reaches a minimum of about 50 at [0.1,0.25] and is significantly increasing afterwards. Due to timeout the further course cannot been tracked, but it may be assumed that a ratio of more than about 50-100 would be maintained well beyond this limit (and possibly throughout the whole range up to 1).

8.4.2 Ratio (i.e. the quotient) between corresponding quantiles for B&B and MIP



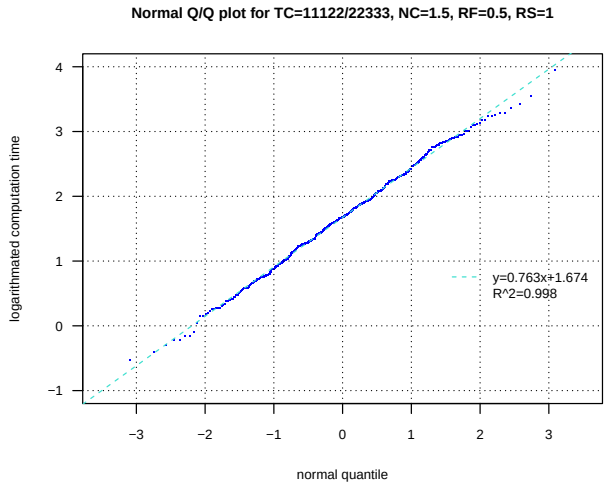
8.4.3 Comparison logarithmated times and linear model:

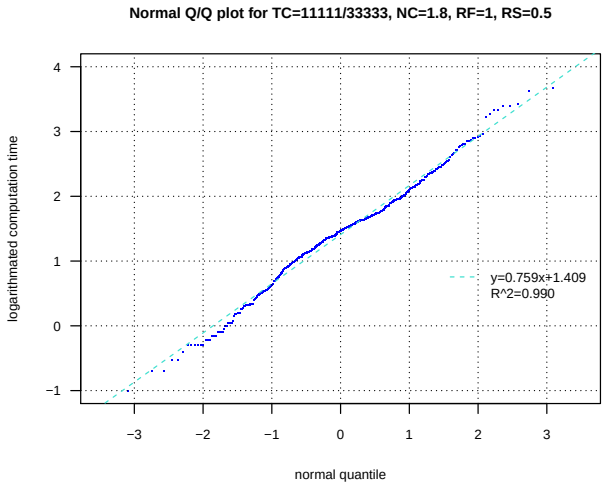


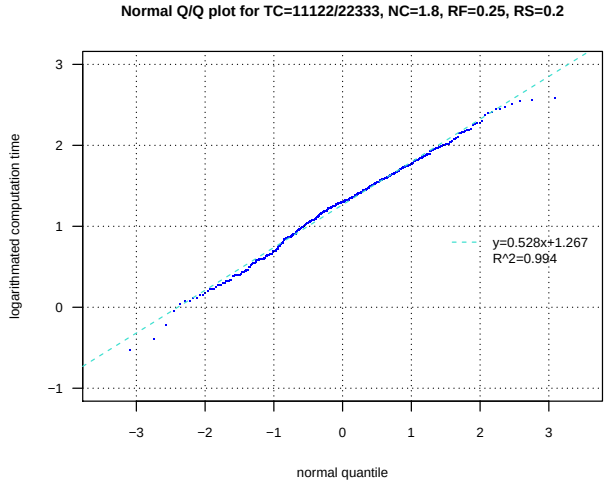
Corresponding coefficients:

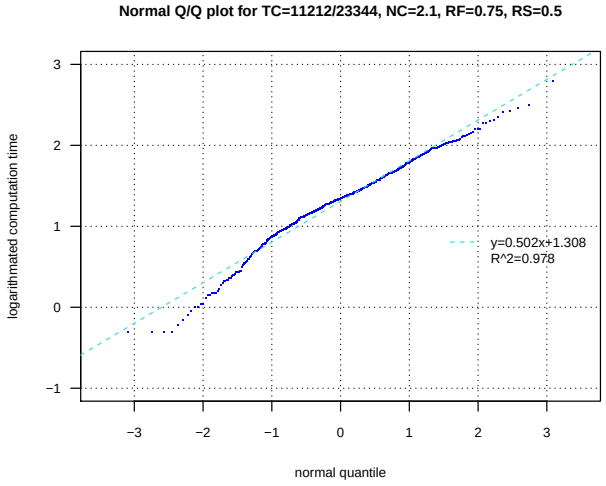
	11111/22222	11111/33333	11111/44444	11122/22333	11212/23344
intercept	3.94 (0.25)	4.21 (0.26)	4.74 (0.28)	5.10 (0.24)	5.17 (0.25)
NC	-1.54 (0.13)	-1.52 (0.13)	-1.85 (0.14)	-1.93 (0.12)	-1.91 (0.13)
RF	1.06 (0.11)	0.66 (0.12)	0.64 (0.12)	0.70 (0.11)	0.66 (0.11)
RS	-1.47 (0.11)	-1.33 (0.11)	-0.92 (0.12)	-0.95 (0.10)	-0.86 (0.11)
r.s.e.	0.38	0.36	0.35	0.36	0.38
R^2	0.73	0.76	0.75	0.74	0.70

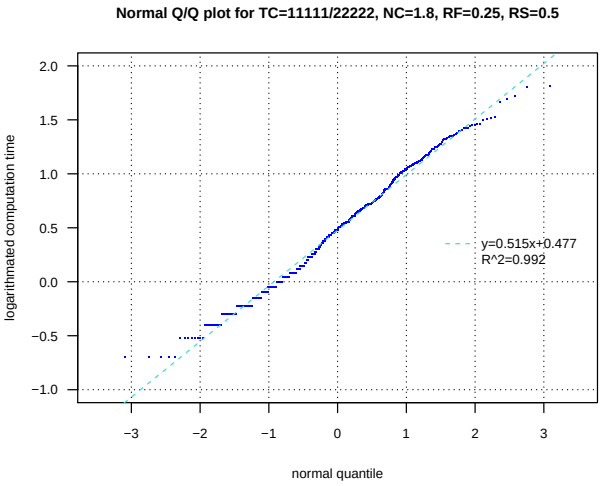
8.5 Distribution of computation times for some parameter combinations

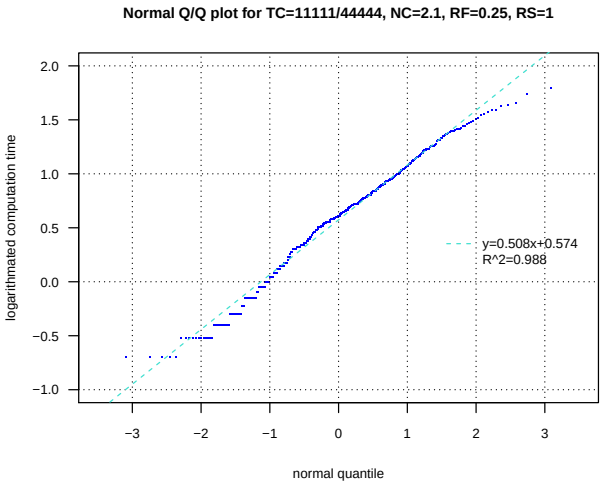


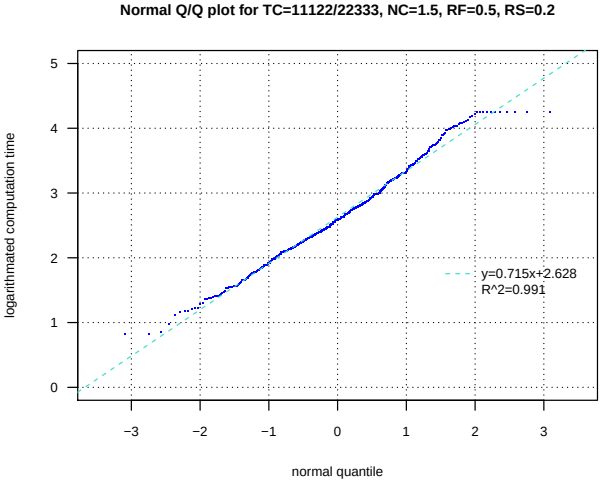







Normal Q/Q plot for TC=11111/22222, NC=1.8, RF=0.25, RS=0.5
logarithmated computation time
normal quantile
y=0.515x+0.477
R^2=0.992


Normal Q/Q plot for TC=11111/44444, NC=2.1, RF=0.25, RS=1
logarithmated computation time
normal quantile
y=0.508x+0.574
R^2=0.988


Normal Q/Q plot for TC=11122/22333, NC=1.5, RF=0.5, RS=0.2
logarithmated computation time
normal quantile
y=0.715x+2.628
R^2=0.991

Part III
Stochastic Activity Exchange Project Scheduling

Stochastic Activity Exchange Project Scheduling

Lukas Berthold

Abstract: Motivated by a practice in aircraft maintenance where an exchange of intact and defective type-identical components between different positions in a system can reduce the duration of a maintenance project, a model integrating such an exchange option in a standard project scheduling model (RCPSP) has been proposed and studied in a preceding paper [Berthold 2021]. The basic idea of component exchange is to move a defective component at a time-critical position to a less critical position with an intact component, which corresponds to move a repair task to a less critical position and replace it by a dummy task. One assumption of that model was that the operational state of components is known at the beginning, for example if a component is visible or if its state is monitored by a monitoring system. However, in many cases the operational state of a component becomes only known after a direct inspection whereas beforehand the probability that a component is defective may only be estimated from historical data. To meet this condition, a stochastic extension of the former deterministic model is introduced and studied. For this, a heuristic approach is proposed which regards the expected reduction of time-criticality by exchanges and performs those exchanges which reduce criticality most. Using simulation, the improvement of applying a component exchange policy compared with no exchange is quantified and the effect of different parameters is studied. For example, it turns out that especially tardiness can be reduced considerably and that the exchange option is more effective if there are few component types with many positions than more types with fewer positions.

1 Introduction and motivation

The problem studied in the following is motivated by a practice occuring in aircraft maintenance (and maintenance of complex systems in other industries), also called "cannibalization" [Salman et al. 2007, Banghart 2017]. Here, to reduce the duration of a maintenance project, the following component exchange procedure may be applied: Assume there are two aircraft in maintenance, the first one planned to finish its maintenance soon, the second one later. Now a component of the first aircraft has turned out to be defective, and a repair or waiting for a spare part would take too long to finish the first aircraft's maintenance in time, hence leading to a possibly costly delay. However, a component of the same type is also contained in the second aircraft (for example, as is often the case since the whole aircraft essentially belongs to the same aircraft family built by the same manufacturer) and is intact there. In such a situation, both components can be removed from their original positions and then moved to the other position, respectively, where they are installed (the second one after repair or replaced by a spare component); in other words, the components are exchanged between both aircraft. As a result, the first aircraft can complete its maintenance in time whereas the defective component, now at a non-critical position, will not cause any relevant delay. As a consequence, if both maintenance procedures are regarded as one overall maintenance project, the exchange procedure admits to move repair work or waiting time for a spare from a critical position to a non-critical one and hence to reduce duration of the project.

In the described situation we have assumed that the functional states (intact or defective) of the components are known in advance. This is or can be the case, if the condition of a component is monitored by a health monitoring system as they are extensively used in aviation [Chen et al. 2012, Rodrigues et al. 2015, Qing et al. 2019]. Under such circumstances one may have enough information at the start of maintenance to get a largely deterministic situation, and a corresponding deterministic model (including more components than only two and more types than only one) has been studied in the preceding paper [Berthold 2021], where also more background information on aircraft maintenance and its relevance with regard to safety and economical issues can be found.

However, for many components their functional state is initially not known, but is only revealed by direct inspection in the course of the maintenance procedure [Nickles et al. 1999, Aust and Pons 2019].

In this case at least the probability that a component is defective may be estimated from historical data as collected by monitoring systems and derived from the number of needed spare parts [Regattieri et al. 2005], and methodologies taking into account related concepts like failure rates and expected remaining useful time have been proposed for aircraft maintenance [Papakostas et al. 2010, Regattieri et al. 2015].
Dealing with such defect probabilities naturally suggests a stochastic version of the former deterministic model, and just such a model is presented and studied in the following. Besides this main difference w.r.t. (un)certainty on the components' states, there will be some further modifications: So the repair of a defective component is immediately started as soon as its state has been recognized (whereas in the earlier model it has been assumed that whole work is done at the position where a component is finally installed); and furthermore we will drop resources to study in detail the effects of pure exchange on time objectives.

2 Literature review

Literature on related models in aircraft maintenance planning and deterministic project scheduling (and here especially extensions admitting more structural flexibility) have already been reviewed in [Berthold 2021]. Consequently, we will focus here on stochastic project scheduling, which is especially relevant for our work.
Projects are subject to diverse uncertainties [Pich et al. 2002], and one point where this becomes especially obvious are activity durations. In real projects, most activity durations can be given only approximately, reasons for which can be found, for example, in estimation errors or external influences such as weather or business partners [Herroelen and Leus 2005]; or, as in our case, it can be missing knowledge on functional states of components which are initially unknown and become revealed only later.
A natural possibility to deal with such uncertainties w.r.t. activity times is to treat them as stochastic variables. Accordingly, project networks with stochastic activity durations have been proposed as early as the first deterministic models. Indeed, the Program Evaluation and Review Technique (PERT), dealing with stochastic activity durations, has been introduced in the end-1950s [Malcolm et al. 1959] and hence at the same time as the Critical Path method (CPM) which works with deterministic durations [Kelley and Walker 1959]. In both approaches projects are represented by project networks and the goal is to determine minimal project duration (makespan); but since, compared with the deterministic CPM model, PERT assumes stochastic durations it aims to determine *expected* makespan. More precisely, the PERT methodology as originally stated works with beta distributions the parameters of which are obtained from a three-point estimation, but this assumption has been often relaxed later towards other distributions [Hajdu and Bokor 2014], such that "PERT networks" has essentially become synonymous to stochastic project networks with any distribution (and sometimes "PERT-networks" simply refer to any, also deterministic project networks, such that some authors have stressed this by using "stochastic PERT networks" which is actually a tautology w.r.t. the original use).
Much work has be done in context of PERT in the decades after its introduction, a comprehensive survey up to 1987 is given in [Adlakha and Kulkarni 1989]. As theoretical result concerning complexity - and in sharp contrast to efficient CPM methodology which admits to determine project duration in linear time - it has turned out that determining the expected makespan (as well as any quantiles of the makespan distribution) is $\#P$-complete already for a simple two-point distribution [Hagstrom 1988]. Here $\#P$ (pronounced "sharp P") is a complexity class containing counting versions of decision problems in NP, and $\#P$-complete are the hardest among them in a comparable manner as NP-complete problems are within NP (namely by polynomial reductions). See also [Arora and Barak 2008] for more about this class. Later work in PERT-networks dealt, for example, with the investigation of the quality of theoretical bounds via simulation [Ludwig et al. 2001]; but most of the focus turned to

the study of more comprehensive models, especially by taking into account resources. Indeed, PERT and CPM do not take resources into account whereas in practice virtually each project is subject to scarce resources. Thus the extension of the basic deterministic project scheduling model which also includes resources, the Resource Constrained Project Scheduling Problem (RCPSP), is from a practical perspective very relevant and, counting also its numerous variants and extensions, the most studied extension of the basic CPM project scheduling model; the surveys [Brucker et al. 1999, Hartmann and Briskorn 2010, Abdolshah 2014, Habibi et al. 2018] and books [Artigues et al. 2008, Schwindt and Zimmermann 2015] convey an impression of the breadth of research on the RCPSP. Consequently, a stochastic extension of the RCPSP, the stochastic RCPSP (SRCPSP), has been studied as well (however, as typical for most combinatorial optimization problems, later and less than its deterministic counterpart).

The study of the SRCPSP has been initiated in the 1980s [Igelmund and Radermacher 1983; Möhring and Radermacher 1984], but further investigations followed only around 2000, and since about 2010 the number of papers increased significantly (see below). In contrast to the deterministic case, the situation becomes more complicated especially insofar as in the stochastic case a solution is not given as a schedule (consisting of start and end times for each activity) but as a *scheduling policy*: Indeed, since exact durations are not known, definite start and end times cannot be given (and hence no fixed schedule); instead, for each decision point one has to specify which activities are to be started next, which yields a scheduling policy. Then a realization of the duration variables leads to a schedule such that in general the same policy will lead to many different schedules, depending on the realizations of the stochastic variables. Correspondingly, the objective is to find a scheduling policy which leads to the minimal expected makespan. This situation differs from PERT where already the trivial policy to start an activity as soon as it has become eligible (i.e. its predecessors have been finished) leads to the minimal expected makespan; thus the difficulty of PERT does not result from having to select from several policies, but only consists in determining the expected value. For the SRCPSP one has to additionally determine the scheduling policy which minimizes the expected makespan. In fact, to the best of our knowldege, the complexity state of the SRCPSP has not been precisely determined yet; especially it is not known if the SRCPSP is $\#P$-complete, which, however, seems improbable since it would imply that one could restrict to policies of polynomial size.

However, optimizing over all possible scheduling policies means to deal with potentially very large solutions and an immense solution space: Indeed, the number of possible decision points is (roughly) exponential in the size of the instance, and for each such point there are usually several decisions which could be made next. Thus already specifying a scheduling policy may need exponential size in general, and the number of possible scheduling policies to be considered may be, at least in principle, double-exponential (i.e. like $\mathcal{O}(2^{2^n})$, for example). Indeed, optimal policies may be rather complicated and may show unintuitive properties. For example, they may be non-elementary, meaning that it may be necessary for an optimal policiy to start activities at a time point different from end times of other activities (a case which cannot occur for RCPSP); an example for this phenomenon is given in [Rostami et al. 2017].

As a consequence, optimizing over all policies has been regarded as intractable [Stork 2001, p. 2], and therefore nearly all researchers have focused on more or less restricted policy classes like earliest-start policies, preselective policies (both already proposed by [Igelmund and Radermacher 1983]), job-based priority policies [Stork 2001], activity-based policies [Ballestin 2007], resource-based policies [Ashtiani et al. 2011] and other ones which can be also described in a more succinct manner and form a smaller search space. Furthermore most have concentrated on heuristic approaches, for example simulated annealing and tabu search [Tsai and Gemmill 1998], GRASP [Ballestin and Leus 2009], genetic algorithms [Ballestin 2007, Ashtiani et al. 2011], approximate dynamic programming [Li and Womer 2015], priority rules [Wang et al. 2017, Chen et al. 2018], and a combination of GRASP and genetic algorithm [Rostami et al. 2018].

Only very few authors have actually applied exact approaches trying to solve the SRCPSP optimally: In [Stork 2001] a branch-and-bound procedure is applied to determine the optimal solution for several policy classes like preselective and earliest start policies; and [Creemers 2015] applied a procedure to find optimal policies even for the unrestricted case. He could solve small to medium-sized instances, but he also reports very large state space sizes for a number of instances. For example, changing from instances with 30 activities to such with 60 leads to a memory increase by a factor 1000 and more.
However, so far this is currently the only work where SRCPSP instances have been actually optimally solved but it suggests that the unrestricted stochastic RCPSP may be not that intractable and inaccessible as suggested by earlier work; nevertheless it remains a very demanding optimization problem. In any case, for our study the particular difficulty of the SRCPSP will be also an motivation to restrict investigation to a heuristic approach at first.
Besides the SRCPSP, there is still another approach to deal with uncertainties in project scheduling which has received some major attention, called proactive/reactive scheduling. Here proactive scheduling means to determine a baseline schedule which is robust against disruptions, whereas reactive scheduling refers to procedures to repair a disrupted schedule; see for example [Deblaere et al. 2011, Lamas and Demeulemeester 2016] for proactive scheduling, [Van de Vonder et al. 2006] for reactive scheduling, and [Davari and Demeulemeester 2019] for an approach combining both aspects. In a more comprehensive setting, stochastic scheduling (and hence also the model introduced and studied in the following) and proactive/reactive scheduling can be seen as parts of the broader topic of project scheduling under uncertainty which also contains some further topics like fuzzy RCPSP or robust optimization approaches; see [Herroelen and Leus 2005] for an overview over different approaches. Furthermore we want to mention that uncertainty in projects can also concern resources [Lambrechts et al. 2011], even if such models have received far less attention than models assuming uncertainty w.r.t. activity durations. Correspondingly, as yet, project scheduling under uncertainty is de facto nearly synonymous to project scheduling under uncertainty w.r.t. activity durations.
Altogether one finds that, as in the deterministic case, no stochastic project scheduling model has thus far been extended by an exchange procedure comparable to the one described. Correspondingly, we will propose a stochastic model which on the one hand extends the problem already studied in [Berthold 2021], and which on the other hand makes use of concepts of stochastic project scheduling, especially similar as developed in context with the SRCPSP. Indeed one could say, while the SRCPSP extends basic PERT by resources, the problem in the following does so by exchange options.

3 Problem definition

3.1 General assumptions

The model presented in the following is a stochastic and modified variant of the project scheduling model with activity exchange (AEPSP) introduced in [Berthold 2021]. Just as there, the idea is to extend a standard project scheduling model by elements capturing component exchange operations as mentioned above; and also as there, this will be realized by project networks with two kinds of activities: On the one side activities associated with component positions possibly involved in exchange operations, on the other side the remaining activities. Whereas the latter ones are simply modelled as standard activities of predetermined duration (whence they will be called "fixed" here, as already in the deterministic model), the duration of the former ones is affected by stochastic outcomes and component exchanges (whence these activities will be called "flexible"). In the following we give a short review of the former deterministic model and, against this background, a first still general description of the key assumptions considered here.
Each "flexible" activity represents some component position which initially contains an individual component which can be either intact or defective. In the former model, this operational state was

assumed to be known at the beginning, which corresponds to the situation that the operational state of a component has been determined before by a diagnostic system, for example. Now, however, we assume the more common case, namely that the actual state of a component is not known in advance but becomes only known by direct inspection. For this we assume that the state is a random variable the value of which becomes known when the component and its position become accessible; in terms of scheduling, this is assumed to happen when the corresponding flexible activity becomes eligible (i.e. all its predecessors have been finished). This transition from a deterministic to a stochastic situation is the first and most important difference to the model in [Berthold 2021].
Depending on its operational state, defective or intact, a component either has to be repaired or not. Thus a component is associated with some amount of work to restore it (only nominal and of duration zero if it is already intact). In the preceding model it was assumed that this work, the *task* associated with the component, is done at the position of the component, thus defining the work content and duration of the flexible activity associated with that position. Usually, i.e. without exchange, that would mean that a defective component is repaired simply at the position where it has been found. In this case the flexible activity associated with that position is processed as repair task (and otherwise, if the component is intact, it is "processed" as dummy task); in this sense a flexible activity and a position are "associated", the flexible activity is related to the work which is actually done at that position. However, if exchanges are allowed, a component can be removed from its original position and moved to another one of the same type (whereas the component initially at that second position must be moved to another position, for example the first one whence both components are exchanged). Correspondingly, in this case also the associated task is moved to the other position to be processed there- and just this option to move work to be done at another position is the very idea behind that exchange option, since it admits to replace time-consuming repair work by no work at time-critical positions whereas moving repair tasks to less critical positions within the maintenance project.
If a position and its initially contained component are involved in an exchange, the task associated with the component (i.e. repair if it is defective) is performed only after the exchange has been realized, a start at the initial position, interruption by moving to another position and continuing repair there is not allowed. However, since it may be optimal to realize exchanges where one component is defective and already accessible, whereas the other one is intact and not accessible yet, it can happen that the start of a repair is delayed in order to realize that exchange. As a consequence, it can make sense to delay the start of a repair under the assumptions of the former model. So this was the situation in [Berthold 2021].
In the following, however, we assume that each repair is started immediately, i.e. as soon as a component has turned out to be defective (this is the second difference to the preceding model). For this one can imagine a central repair shop to which a defective component is sent and repaired, after which the restored component can be sent to and built in the original position - or possibly in another one for the same component type, given the exchange option. Sending to and from the repair shop and installing the component at its final position are assumed to be done in negligible time, i.e. time zero (as already in the preceding model). This exchange procedure can be exemplified by the following situation:

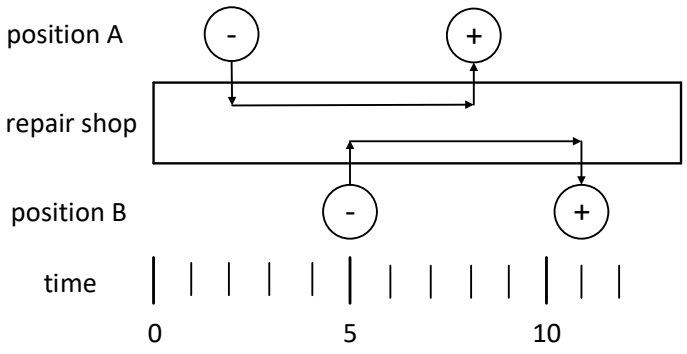


Figure 1: Repair shop without exchange

Here a maintenance project is running, currently we are at $t = 7$ and two positions A and B of the same type 1 are considered. Position A has become accessible at $t = 2$, position B at $t = 5$, and both components contained there, c_A at A and c_B at B, have turned out to be defective and sent to the repair shop. Repair of a type-1 component takes $\tilde{d}_1 = 6$ time units whence c_A and c_B will be restored at $t = 8$ and $t = 11$, respectively. Without exchange, each component will return to its original position after being repaired, which is also the point at which the activities associated with the positions will finish, i.e. activity A will end at $t = 8$ and activity B at $t = 11$ (remaining times 1 and 4; for convenience we will identify associated positions and activities and refer to them by the same letter).

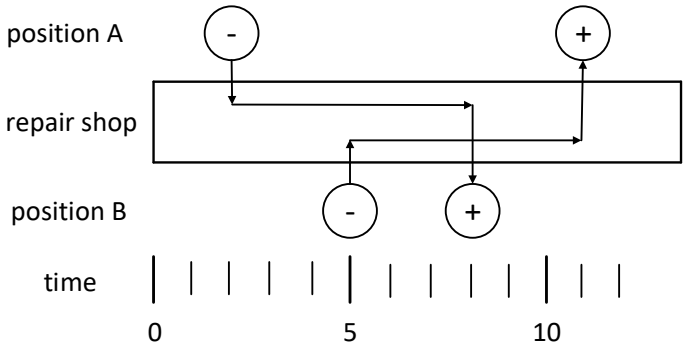


Figure 2: Repair shop with exchange

If exchange of components of the same type is allowed and also applied in this situation, component c_A will be moved to and installed at B after repair, and c_B at A. As a consequence, now activity A would end at $t = 11$ and activity B at $t = 8$, i.e. an exchange of components corresponds to an exchange of end times (or, equivalently, of the remaining times).

Hiding the repair shop, the component exchange can be visualized more clearly as exchange of end times:

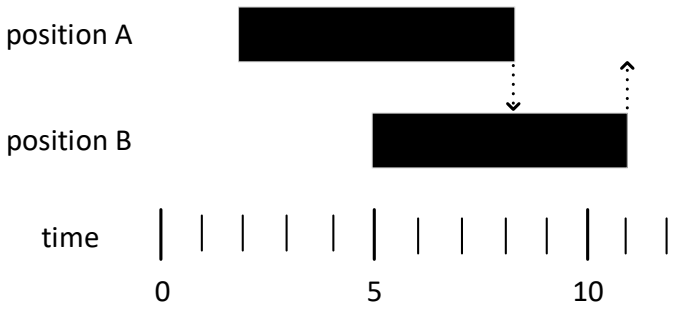


Figure 3: Exchanging finish times

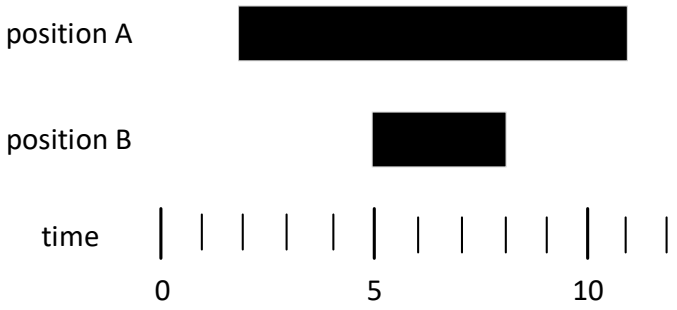


Figure 4: After exchange

The exchange option makes it necessary to distinguish, for each position, between the component initially and finally contained at some position. Of course, both are identical if the position is not involved in any exchange.
A component just repaired need not to be installed immediately at some open position of suitable type, let it be the original one or another open at that moment; instead it may be advantageous to delay the installation and to wait until a component at some time-critical position becomes accessible, since in this case this possibly defective component can be immediately replaced by the repaired component. Such forward-looking reasoning will be typical for this problem, and the situation becomes more complicated due to the uncertainty about the functional state of components becoming accessible only later. Also already intact components can take part in exchanges and may be removed from their original position and moved to another more critical one, which otherwise had to wait. Below we will see some examples for such intentional delays and removals of already intact components from their original positions and the advantages gained by such exchanges.
The second representation above suggests that alternatively one could also assume the possibility that the repair of a component is started at its original position and then possibly continued at another one. In contrast to this, in the former model it was assumed that whole repair must be performed at only one position, either the original one or at another one after an exchange.
The model introduced in the following will also allow another natural and equitable interpretation, namely replacement by spare components which can be delivered within a certain lead time. In this case, instead of being repaired, an intact spare component is ordered as soon as a component has been found to be defective and will arrive after some type-specific time. Thus repair times are translated into lead times; both describe the time after which an intact component is available again. As a consequence, one could also model mixed situations where defective components of one type are repaired whereas those of another one are replaced by ordered intact components. In the following, however, we will prefer to stay in the repair setting alone, for clarity and also to be in a line with the preceding model.

After this general overview on the key assumptions of the model studied in the following and its comparison with the preceding deterministic model, we start with some examples before stating it formally afterwards.

3.2 Examples

Regard the following maintenance project network:

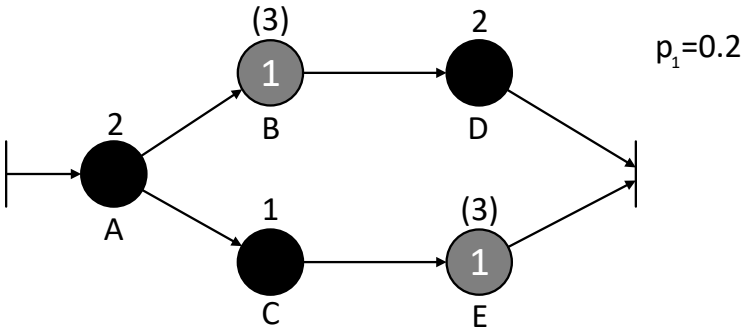


Figure 5: Project network without exchange

Here black nodes (A, C, D) correspond to fixed activities with predetermined durations and grey nodes (B, E) to flexible activities. Each flexible activity corresponds to a component position designated for components of some type (here both positions are for components of type 1, as indicated by the number within the node); for convenience we will often identify flexible activities and corresponding component positions and refer to them by the same letter, i.e. simply speaking of activity B which starts at t or position B which initially contains an intact component etc. This will not cause any difficulties since it will be clear either from the context if an activity or a position is talked about, or simply since it is explicitly attributed as "activity A" or "position A".
Each component position initially contains an individual component of corresponding type (here type 1 for both positions) and of some still unknown operational state (either intact or defective). Known, however, is the probability that a component at a given position is defective. This probability is assumed to be the same for components of the same type (here $p_1 = 20\%$ for the only type 1). Thus the initial state of a position is a random variable with possible values 0 (intact) and 1 (defective) and we assume that all of these random variables are independent (as already in the preceding paper, the choice of the values is reversed compared with the more common use, but it will be more convenient in our setting especially since in this case value 0 indicates work of duration 0 whereas value 1 indicates work of non-zero duration). The number in brackets above a flexible node denotes the duration of a repair task for a defective component of corresponding type; so it takes $\tilde{d}_1 = 3$ time units to repair a defective component of type 1 whereas an intact component needs no time. Thus, associated with a component is the time to restore that component, which, as its state, is a random variable as well.
At first we regard the case where each component remains at its original position (visiting the repair shop in between if it turns out to be defective). At $t = 2$, when A has been finished, activities B and C become eligible (i.e. all its preceding activities have been finished). For a flexible activity (as here B) and the corresponding position, this is the point at which we assume that the component which is initially contained at that position becomes accessible (as also already assumed for the deterministic model) and that its actual operational state becomes visible (which is new here). Here it turns out that the component at B is defective. Without the possibility to exchange it returns to (position) B after repair at $t = 5$ which finishes activity B (since position B has got a restored component now which is decided to remain there). Besides B, activity C has been started at $t = 2$ as well; at $t = 3$ activity E becomes eligible and the component contained there (or, more precisely, at *position* E)

turns out to be intact whence there is nothing to do and (activity) E can be finished immediately. Finally, after the component at B has been repaired, D can start at $t = 5$ whence the whole project finishes at $t = 7$.
Now we assume that exchanges are allowed, i.e. components can be moved to positions different from (but of the same type as) those where they have been initially found. Thus defective components need not necessarily return to their original positions after repair but can be installed in other positions of the same type; and also already intact components can be removed from their initial positions and installed in other ones. However, since we assume all component positions to be essential for the operability of the system (i.e. aircraft), each position must finally contain an operational component whence a component removed from one position must be replaced by another component of the same type. Operations of removing, moving and installing are assumed to be done in negligible time, i.e. time zero; only repair times contribute to duration. Due to the exchange option, we distinguish, for each position, the component initially found there from that one finally installed there: both are identical, if a component remains at its original position, they are different (but of the same type), if the component was involved in an exchange.
So let the project start again, and after B and C have become eligible at $t = 2$, the component at B turns out to be defective whence its repair is started, and C is started as well. At $t = 3$ activity E becomes eligible and the component at the corresponding position turns out to be intact again. Now the question arises if an exchange between B and E, i.e. of the components initially found there makes sense. It does: The component from B is still under repair until $t = 5$, so the remaining repair time is 2. Thus, if the components between B and E are exchanged, position B gets the intact component from E whence activity B can be immediately finished, and D can be started still at $t = 3$ and ends at $t = 5$. On the other hand position E gets the restored component from B at $t = 5$ which also finishes activity E. Thus the maintenance project ends at $t = 5$ and hence earlier than without exchange at $t = 7$. Therefore the exchange option can reduce the project duration indeed.
In this example it could be seen that an exchange of components corresponds to an exchange of end times (and remaining times): Without exchange B and E would end at 5 and 3, respectively, a component exchange leads to an exchange of these end times.
In the case just seen the question, whether an exchange is to be realized or not, arose not before both components had become accessible and thus their states were already known. However, in general it may be advantageous to take possible exchanges into account already when only one component is accessible and intact whereas the other component is not accessible yet. In this case also the exchange cannot be performed yet, nevertheless it has to be decided if the first, already intact component is to remain at its current position and the corresponding activity can be declared finished such that its successors can start, or if it is better to wait until the second component becomes accessible in order to possibly realize an exchange (depending on the state found for this second component). The following example illustrates this situation:

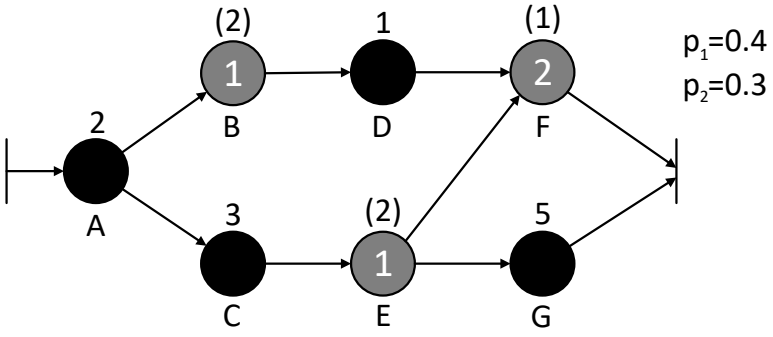


Figure 6: Project network without exchange

Here A, C, D, G are fixed activities, B, E, F are flexible, B and E of type 1 and F of type 2 (and all of still unknown state). The defect probabilities are $p_1 = 0.4$ for type 1 and $p_2 = 0.3$ for type 2. Repair times for defective components are $\tilde{d}_1 = 2$ for type 1 and $\tilde{d}_2 = 1$ for type 2.
The project starts with A, which is finished at $t = 2$. At that point B and C become eligible and B turns out to be defective. Repair of the component found at B is immediately started and C is started as standard activity as well. At $t = 4$ the component from B is restored, and now the question arises if a later exchange with E is advantageous. Without exchange, the restored component is simply built in position B again which finishes activity B at $t = 4$ whence D can be started. Following C, E becomes accessible at $t = 5$ and depending on the initial state of E, activity E ends at $t = 5$ (intact with probability 0.6) or at $t = 7$ (defective with probability 0.4). After finish of D and E, position F becomes accessible, and depending on the initial state of F, we get finish times (with state outcomes and their probabilities) 5 (E and F intact: 0.42), 6 (E intact, F defective: 0.18), 7 (E defective, F intact: 0.28) and 9 (E and F defective: 0.12) whereas G ends at 10 (E intact: 0.6) or 12 (E defective: 0.4) and hence determines the project duration. Thus expected makespan under given conditions is $\mathbb{E}(MS|B\ initially\ defective\ \wedge\ no\ exchange\ between\ B\ and\ E) = 10.8$.
On the other hand, for an exchange the restored component must wait until $t = 5$ when activity E becomes eligible. If the component at E turns out to be defective, an exchange between B and E corresponds to an exchange of remaining times 0 and 2, i.e. B gets the component from E which will be repaired at $t = 7$ and E gets the repaired and now-intact component from B and can be immediately finished at $t = 5$. Then D ends at 8 and G at 10. Since F ends not later than 9, the project duration is 10 - if the component at E has been found defective. If not, i.e. if it is already intact, then an exchange is superfluous since both remaining times are 0. Thus both B and E end at 5, and, as a consequence, D at 6 and G again at 10. Since F finishes not later than at 8, the project duration is 10 as well. Altogether we always get a makespan of 10 if the end of B is delayed for a (possible) exchange with E whence this alternative is to be preferred if expected makespan is the relevant objective.
This case was still not too complicated to be treated within text, but in general one has to solve a stochastic dynamic programm to take all possible decisions and outcomes of state variables into account. For this regard the instance above with $\tilde{d}_2 = 6$ instead of 1: Again A finishes at 2, C is started and will be finished at 5, repair of B ends at 4, and again (at $t = 4$) the question arises if the end of B is to be delayed for possible exchange with E. Seen from the situation $t = 4$, the full stochastic dynamic program to find the optimal choice in order to achieve a minimal expected makespan (see Appendix 8.1) yields as best policy to delay the end of B and to perform an exchange if the component at E turns out to be defective; if it is intact then an exchange does not have any effect and hence is not necessary. This policy leads to an expected makespan of 10.84 (slightly better than 11.1 without delay and exchange), the actual makespan will take a value from $\{10, 12, 13, 14\}$, depending on the outcomes of the random state variables of E and F. This example makes clear that an optimal solution for the decision problem considered here does not consist of a simple schedule with fixed start and end times for each activity but rather of a *scheduling policy* which makes for each possible decision point a decision depending on all information known up to that point (the dynamic program from above determines only a partial policy, namely for the case where B has already turned out to be defective; a complete policy, applicable already from the start at $t = 0$, would have to take into account that B was intact and thus would need to decide whether to delay the end of B in this case or not). As mentioned in the literature review, such scheduling policies are well-known as solutions of stochastic scheduling problems [Stork 2001].
Where finding an optimal scheduling policy is regarded as infeasible as for the stochastic RCPSP, the optimization is typically restricted to special classes of policies or applies heuristics. We will also not solve the problem presented here optimally but will apply a heuristic approach the basic idea of which we want to demonstrate already at this point for the example above. For this we will use expected

durations for random variables the value of which is still unknown: Thus E gets a deterministic duration $d_E = 0.4 \cdot 2 = 0.8$ and F gets $d_F = 0.3 \cdot 6 = 1.8$. We start at the current point where the question of delay or exchange arises, here at $t = 4$. Without delay, we simply perform a forward pass using the determinstic durations and get a makespan of 10.8. With delay (and subsequent exchange), B is not finished at 4, but is delayed until E becomes accessible at 5. Corresponding to an exchange of components, the remaining durations are exchanged: 0 for B (since the component is already restored) and 0.8 for E (expected duration associated with the component found there). Thus B starts at 5 with duration 0.8 and E with duration 0. Forward pass leads to a makespan of 10; since this is smaller than that without delay, a delay is performed according to this criterion. Of course, this is only a heuristic criterion, and it may (and will sometimes) also suggest not the optimal decision i.e. a different one from that determined by a dynamic program as above. Nevertheless it is reasonable to assume that this relatively simple criterion shows the right tendency and hence is worth to be considered, especially given the high effort to determine the optimal decision as indicated above. Indeed, our computational studies will show that this criterion and its extensions for different cases lead to an improvement if the exchange option is allowed.

Before defining the problem formally below, we still give a summary of the assumptions of the model:

- An instance is given by a project network containing two kinds of activities, fixed and flexible (and some further data like types, type-dependent defect probabilities and repair durations).
- Fixed activities are standard activities of predetermined duration.
- Each flexible activity represents a component position of some type, which is also the type of that activity.
- In general there are several component positions (and hence flexible activities) of the same type. (otherwise no exchange would be possible for this type, such that the activitiy is simply performed according to the state found there).
- Each component position initially contains a component of corresponding type.
- This component is initially either intact or defective which is its (initial) operational state.
- This operational state is not known before but a random variable which can take two possible values 0 (intact) and 1 (defective). The probability to be defective is the same for all positions of same type, i.e. depends only on the type of the component.
- All these random variables are independent.
- The actual state of the component initially contained at a position becomes visible if this position becomes accessible, which in turn is the case if the corresponding flexible activity becomes eligible, i.e. all its predecessors have been finished.
- A component found to be defective is repaired, the repair is immediately started.
- The repair time is the same for components of the same type, i.e. it depends only on the type of a component.
- For an intact component there is nothing to do, it is associated with a dummy task of duration zero.
- A component becomes unavailable for exchange if it is finally installed at some position.
- After repair a defective component becomes intact (such that an intact component has been either initially intact or restored, i.e. primary or secondary intact).
- A flexible activity is finished if an intact is finally installed at the corresponding position.
- Each component (also an initially intact one) can be removed from its initial position and moved to another position of the same type and installed there (initially defective components after having been repaired); in this case it must be replaced by another component (from another position).
- In contrast to repair, operations of removing, moving and installing are assumed to take no time.
- Thus, for each position, one can distinguish between the component initially contained there and that finally assigned to that position; if the initial component is not moved to another position, then

both are the same.
- A component exchange between two positions corresponds to an exchange of end times (or, alternatively, remaining times) of activities.

Now we are prepared to define instances formally.

3.3 Formal instance definition

An instance $\mathcal{E}$ of the Stochastic Activity Exchange Project Scheduling Problem (SAEPSP) is given by the following data:

- a set of *activities* $A = \{0, \ldots, n+1\}$ where 0 will act as dummy start activity and $n+1$ as dummy end activity
  (in our examples we prefer to use letters for activities to avoid the use of too many numerical values, which could be a source of confusion)
- a binary *precedence* relation $P \subseteq A \times A$ such that (A, P) is an acyclic digraph and 0 (resp. $n+1$) is direct or indirect predecessor (resp. successor) of all other nodes. For an activity $i \in A$, let P_i (resp. S_i) denote the set of direct predecessors (resp. successors) of i.
- a set $\mathcal{T} = \{1, \ldots, \bar{\tau}\}$ of *types*
- a partition $A = \bigcup_{\tau=0}^{\bar{\tau}} A_\tau$ (i.e. with $A_\tau \cap A_{\tau'} = \emptyset$ for $\tau \neq \tau'$) where
  $A_0 =: A^{fx}$ is the set of *fixed* activities (with $0, n+1 \in A_0$),
  A_τ is the non-empty set of *activities of type* τ for $\tau = 1, \ldots, \bar{\tau}$, the union of which is
  $A^{fl} := \bigcup_{\tau=1}^{\bar{\tau}} A_\tau = A \backslash A_0$ the set of *flexible* activities
  $\tau(i)$ denotes the type of flexible activity i, i.e. it holds $i \in A_{\tau(i)}$
- durations of fixed activities $d_i \in \mathbb{N}_0$ for $i \in A_0$ with $d_0 = d_{n+1} = 0$
- durations of flexible types $\tilde{d}_\tau \in \mathbb{N}$ for $\tau = 1, \ldots, \bar{\tau}$
  (these durations for flexible types refer to the time needed for repair taks, i.e. to restore defective components of given type; the dummy tasks for intact components need no time)
- defect probabilities $p_\tau \in (0, 1)$ for $\tau = 1, \ldots, \bar{\tau}$
  (each flexible activity i is associated with an initial state random variable σ_i^0 which can take either the value 0 (intact) or 1 (defective), the latter with probability $p_{\tau(i)}$; all these random variables are assumed to be independent)
- set $M \subseteq A$ of milestone activities and due dates dd_i for each milestone $i \in M$
  (if tardiness is the objective under consideration; a due date dd_i is a point of time up to which a milestone activity i should be finished, otherwise the exceedance is penalized by the objective function)
- objective is makespan $MS = f_{n+1}$ or tardiness $Ta = \sum_{i \in M} max\{f_i - dd_i, 0\}$,
  where f_i denotes the finish time of activity i. Whereas makespan is simply the project duration, for tardiness one considers by which amount of time each milestone activity exceeds its due date and summarizes over all these exceedances.

Applying the notation to the second example above, we have $\mathcal{T} = \{1, 2\}$, $A_1 = \{B, E\}$, $A_2 = \{F\}$, $\tilde{d}_1 = 2$, $\tilde{d}_2 = 1$, $p_1 = 0.4$, $p_2 = 0.3$. Since makespan has been regarded, no milestone activities were specified.

In contrast to the deterministic case the initial state value function $\sigma_0 : A_{fl} \to \{0, 1\}$ is replaced

by initial state variables σ_i^0 and defect probabilities p_τ for $\tau \in \mathcal{T}$, a natural transition from the deterministic to the stochastic case. Furthermore we will not regard any resources here; the main reason for this is to study better the interplay between stochasticity and exchange without complicating the situation by introducing a new dimension (which, of course, could be interesting in subsequent studies).
Thus having specified instances formally, we turn to the actual scheduling problem now. In order to state the scheduling problem properly, we have to define (partial) schedules, decision points and feasible exchange operations at first.

3.4 Schedules, decisions points, decisions

3.4.1 Partial schedules

A *(partial) schedule* $\mathfrak{p} = (S, s, f)$ is given by the following data:

- a set of *scheduled activities* $S \subseteq A$ which is closed under precedence, i.e. if $j \in S, i \in P_j$ then $i \in S$. If $S = A$ then the schedule is *complete*.
- start $s(i)$ and finish times $f(i)$ with $s(i) \leq f(i)$ for each $i \in S$ (with values in $\mathbb{N}_0$)

Where necessary for further distinction, we will specify the objects associated with a schedule by a superscript like $S^\mathfrak{p}, f^\mathfrak{p}$, etc. Besides trivial $s(i) \leq f(i)$, we further assume that a schedule is feasible w.r.t. precedence (i.e. $f(i) \leq s(j)$ if $i \in P_j$) and duration for fixed activities (i.e. $f(i) - s(i) = d_i$ for $i \in A^{fx}$). For flexible activities we specify no condition at this moment since feasibility of a partial schedule will result from applying feasible decisions (from which also the other feasibility conditions just assumed will follow); see below. Particularly, due to the options to exchange and also to delay the end times of flexible activities to admit later exchanges as described below, it will be possible that the "duration" $f_i - s_i$ is neither 0 nor d_τ for a flexible activity $i \in A_\tau$ of type τ.
Whereas the start and end times of fixed activities need no further explanation, for flexible activities, due to their special character, it is important to clarify their special interpretation. As already seen in the examples, each flexible activity corresponds to a component position of some type (such that we have found it convenient to refer to them by the same i). Now the "start" of a flexible activity corresponds to the time when the associated position is reached, which is also the time when the individual component contained there becomes accessible and its state known. If found to be defective, its repair is immediately started, such that in this case the start time of the activity is additionally also the start time of repair of the component. On the other hand, the "end" of a flexible activity corresponds to the time when an intact component (either initially intact or repaired) is finally assigned to the associated position and installed there, in which case we also say that the position is "closed". From this point the component there is not available for further exchanges any more.
Furthermore we will assume that at each time between start and end of a flexible activity a component is assigned to the position associated with that activity. At start this is the component found at the position, at end this is the component finally installed there (both can be the same). In between other components may be assigned to the position, such that the assignment is only preliminary between start and end time, and can be be switched by exchange operations.
Regarding start times in general (also for fixed activities), note that without resource restrictions or any further complicating restriction an activity can (and will) be started as soon as it has become eligible, i.e. if its predecessors have been finished. Thus the start time of an activity is given by the maximal end time of its predecessors and hence determined by them. Due to this redundancy, it would also suffice to restrict description of a schedule to the pair (S, f).

3.4.2 Decision points

Schedules will mostly occur here in extended form as decision points. A *decision point* is a triple $\delta = (\mathfrak{p}, t, F)$ consisting of

- a partial schedule $\mathfrak{p}$
- a time t with $s_i \leq t$ for each $i \in S$
- a set of finished activities F with $\{i \in S | s_i < t\} \subseteq F \subseteq \{i \in S | s_i \leq t\}$

The explicit set of finished activities is introduced to keep track whether flexible activities are delayed even after their end time has arrived in order to swap them with another component later down the line. Fixed activities enter the set immediately when their end times have arrive. Examples for this have been presented above.
An activity $i \in A$ is *eligible* at a decision point if it has not been scheduled yet and its predecessors have been finished, i.e. if $i \notin S$ and $P_i \subseteq F$. Thus, in principle, an eligible activity can be started next.

3.4.3 Decisions

As already suggested by the name alone, a decision point does not only describe the schedule up to the current point in time but also represents a situation in which a decision is to be made. For many scheduling problems, as also for the RCPSP, the typical decision in such a situation consists of deciding which eligible activities have to be started next. However, as already mentioned above, in our case of the SAEPSP this is not a real decision since all activities can and will be started as soon as they have become eligible. Instead, decisions are related to the exchange option characteristical for our problem. More precisely, a decision has two aspects, exchange and delay:
The first decision concerns the end times of not finished flexible activities, which can be exchanged in an arbitrary manner among activities of the same type (of course, for some activities the end times can remain the same and in fact one can also decide to do no exchange at all). That means, that the finish times for flexible $i \in S \backslash F$ can be permutated in any way which respects types (a flexible activity can get the end time only from another one of the same type). This corresponds to a permutation of components assigned to positions (and hence to flexible activities): Namely, if $i = \pi(j)$ and hence the end time of i is updated as $f'(i) := f(j)$, then this can be understood that the component assigned before to position j and being intact and hence ready for installation at $f(j)$ is now assigned to i where it can be installed at the same time. Thus, with this new assignment, position i could be closed now at $f'(i)$, which translates to the new end time $f'(i)$ for flexible activity i.
On the other hand, for each flexible activity the end time of which has been just reached (i.e. $f(i) = t$ for current decision time t), it is decided if this activity is actually to be finished (corresponding to an installation of the intact component currently assigned to the corresponding position and hence declaring the position for closed), or if the end of this activity (and hence the closure of the associated position) is to be delayed (for a possible later exchange). This latter possibility corresponds to the option that the intact component currently assigned to the position associated with that activity is not finally installed here, but is kept available for a possible later exchange.
Formally a decision can be represented as a pair (π, D) where π is a permutation of $\{i \in S^{fl} \backslash F | f_i \geq t\}$ respecting types (i.e. $\pi(i) = j \Rightarrow \tau(i) = \tau(j)$) and $D \subseteq \{i \in S^{fl} \backslash F | f_i = t\}$. Note, of course, that π may be the identity (no exchange) or that it may leave at least some activities fixed (hence do not taking part in any exchange).
For a decision point δ, we denote by $\mathcal{D}^\delta$ the set of possible decisions for that decision point. Given a decision (π, D) for the current decision point, the next decision point is constructed as follows:

- apply π to the corresponding finish times in $\mathfrak{p}$, i.e. set $f'(i) := f(\pi(i))$

- declare all activities in $\tilde{F}^{fl} := \{i \in S^{fl} \backslash D | f_i = t\}$ as finished

Moreover, besides these changes immediately given by the decision made, there are the following further changes:

- proceed to the next finish time $t' := min\{f_i | i \in S \backslash F\}$ (which may be the same due to activities of duration 0)
- set fixed activities ending there as finished $\tilde{F}^{fx} := \{i \in S^{fx} | f_i = t'\}$
- then the whole new set of finished activities is $F' := F \cup \tilde{F}^{fl} \cup \tilde{F}^{fx}$
- determine the set $El' = \{i \in A \backslash S | P_i \subseteq F'\}$ of new eligible activities
- and start them at t' with end times according to fixed duration for fixed activities and according to initial state for flexible ones, i.e. set

$$f_i := \begin{cases} t' + d_i & , i \in El^{fx} \\ t' + \sigma_i \tilde{d}_{\tau(i)} & , i \in El^{fl} \end{cases}$$

  where σ_i denotes the realization of the state variable for flexible activity i
- for flexible activities the end of which has been decided to be delayed set t' as new end time, i.e. $f_i := t'$ for each $i \in D$

Altogether this yields the new decision point $(\mathfrak{p}', t', F')$ resulting from the application of (π, D) on δ.

Note that this next decision point depends not only on the last decision(s) but also on the last stochastic realizations (as reflected by the end times of flexible activities which have become eligible and got their duration according to the component found at the corresponding position). Specifying a decision for each possible decision point, one gets a *scheduling policy*. Correspondingly, a scheduling policy (for given instance) can be represented as a function which maps decision points to decisions, i.e. $\Pi : \mathcal{DP} \to \mathcal{D}$ where $\mathcal{DP}$ is the set of decision points and $\mathcal{D}$ that of decisions; here the decision $\Pi(\delta) = (\pi, D)$ has to be feasible for the decision point.
Scheduling policies can come in the form of compact rules, which specify which activity is to be scheduled according to some set of general criteria, but also in the form of extensive tables which specify a particular scheduling decision for each theoretically possible decision point. Since the number of schedules and hence also of possible decision points is exponential in the instance size and since further there are usually several possible decisions for each decision point, the number of possible scheduling policies is roughly double-exponential (like $\mathcal{O}(2^{2^n})$) and hence forming a huge solution space.
Given an instance, the application of a scheduling policy and the realizations of the random variables will determine a unique complete schedule with some makespan (or tardiness). Then the objective of a stochastic scheduling problem is to find a policy which leads to minimal expected makespan (or tardiness). More precisely, applying a scheduling policy Π to an instance $\mathcal{E}$ yields, by successively revealing the realizations of state values $\bar{\sigma}$, a well-defined and unique complete schedule $\mathfrak{s}^{\Pi}(\bar{\sigma})$ with likewise well-defined objective value $Obj(\mathfrak{s}^{\Pi}(\bar{\sigma}))$ (with $Obj \in \{MS, Ta\}$ the objective under consideration). Weighting these values with the probability of the state vector realization $\bar{\sigma}$, one gets the expected objective value for applying that policy Π on the instance $\mathcal{E}$, i.e. $\mathbb{E}[Obj(\mathfrak{s}^{\Pi}(\bar{\sigma}))]$.
Now, given an SAEPSP instance, the principal goal is to find a policy which minimizes the expected makespan (or tardiness) and to determine its expected value, formally:

Objective SAEPSP: Find Π^* with $\mathbb{E}(Obj(\mathfrak{s}^{\Pi^*}(\bar{\sigma}))) = min_{\Pi}\, \mathbb{E}(Obj(\mathfrak{s}^{\Pi}(\bar{\sigma})))$

where $Obj \in \{MS, Ta\}$ is the objective under consideration.

This completes the formal definition of our optimization problem. However, since it contains PERT and hence it is a $\#P$-hard problem, we will not try to solve it exactly such that the objective will remain theoretical in this study. Instead we will apply a heuristic which promises reasonably good solutions. This heuristic solution procedure is presented next.

4 Solution procedure using exchanges

4.1 General idea

The heuristic solution procedure presented in the following steps from one decision point to the next one, and at each such point it checks for each relevant pair of flexible activities whether an exchange between them, i.e. an exchange of their remaining durations and end times, is promising according to a certain criterion. Pairs found to be promising are ordered according to their valuation computed for that criterion, which yields a priority list, and then realized in this order unless there is a conflict with an exchange of higher priority realized before. The next ending activity (which need not to be one of those candidates for exchange, but can be a fixed one as well) determines the next decision point, at which the procedure is repeated. This procedure is continued until the end of the project has been reached and the makespan (or tardiness) value achieved is determined.
That criterion applied to judge if an exchange is "promising" or not is based on the delaying effect which the ending of some activity i at some t has on the ending of the activities related to the corresponding objective, i.e. the end activity (for makespan) or the milestone activities (for tardiness). More precisely, we regard the end times of these objective-related activities which result from performing a forward pass starting at an end time t for i: Starting with earliest finish time for i as $EF_i := t$ yields earliest finish times EF_j for each direct and indirect successor j of i, including the end activity $n+1$ and some (but in general not all) milestone activities (if tardiness considered). Thus the end of i at t enforces a minimal finish time for the end activity and minimal due date exceedances of some milestone activities (if such are given), and hence implies a lower bound for the objective regarded. However, for a forward pass we would need complete information on durations and hence also known durations for flexible activities which have not become eligible yet, whence we only know the distribution of their (initial) duration.
To circumvent this difficulty, we will use expected durations for flexible activities for which the initial state has not been revealed yet, i.e. duration $p_\tau d_\tau$ for a flexible activity of type τ. In the field of stochastic project scheduling it is a well-known fact that a forward pass using expected durations as substitutes leads to an underestimation of expected start and finish times and hence also of project duration (commonly known as Jensen's inequality, see [Adlakha and Kulkarni 1989, Möhring 2001]), whereas, as mentioned earlier, determining the true expected start and end times as well as path lengths is a $\#P$-complete problem. Nevertheless the correlation with the true expected values may be supposed sufficiently strong to yield in most cases reliable information if an exchange may be "promising" indeed. Furthermore the effect of systematic underestimation will be partially compensated since the exchange criteria presented below are based more on relative changes instead on absolute values. Finally the value of still better estimations (some of which have been also suggested but being also computationally more expensive; see [Ludwig et al. 2001]) may be limited since in our setting durations do not only depend on outcomes of stochastic variables but also on exchanges. Altogether, for a first heuristic approach to the problem studied here and to establish some first benchmark results, it seems worthwile to work with those approximate values from forward pass with expected durations, and our results will actually show that exchange based on them will yield a significant improvement compared with no exchange indeed.
So far we have only described the values associated with single flexible activities (ending at some point t). These values are used for an exchange criterion as follows: Given a potential exchange pair

of activities i_1 and i_2 ending at t_1 and t_2 and having values $v(i_1)$ and $v(i_2)$, respectively, an exchange means exchanging end times, resulting in new values $v'(i_1)$ and $v'(i_2)$. As mentioned above, due to underestimation these values are lower bounds for the optimal value z of the objective considered, whence we have $v(i_1) \leq z$ and $v(i_2) \leq z$ and hence $max\{v(i_1), v(i_2)\} \leq z$, and $max\{v'(i_1), v'(i_2)\} \leq z$. Thus the maximal value before and after exchange, respectively, is also a lower bound and it is advantageous if it decreases after exchange. Correspondingly, we regard an exchange as promising, if $max\{v'(i_1), v'(i_2)\} < max\{v(i_1), v(i_2)\}$. However keep in mind, that these values are not hard lower bounds due to uncertainty of stochastic durations, such that the criterion only reflects a tendency towards a lower objective value which may be achieved by application of given exchange.
In order to avoid performing repeated forward pass computations at each decision point, we will compute as preprocessing the distances between each pair of activities i, j where j is direct or indirect successor of i. Here the distance between such pair i, j is the minimal duration between end of i and start of j (using expected durations for flexible activities), i.e. $dist(i, j) := ES_j - EF_i$. A table of these values can be efficiently computed in time $\mathcal{O}(n^2)$.
Now we are prepared to state the exchange criteria for both objectives regarded here, makespan and tardiness. For both the main idea is similar, only adapted to each objective.

4.2 Criterion for Makespan

Being at some decision point, let be given two flexible activities i_1 and i_2 of the same type. There are two cases which we consider: On the one hand, where both activities are eligible or active (i.e. both have already become eligible yet); on the other hand, where only one (say i_1) is, whereas the other one has not become eligible yet.
In the first case the end times for both activities are known (if an activity has become eligible, its initial state has become revealed). In this case the values are set to $v(i_1) := t_1 + dist(i_1, n+1)$ and $v(i_2) := t_2 + dist(i_2, n+1)$, each giving a (soft) lower bound for the completion time of the project, i.e. makespan. An exchange would result in values $v(i_1) = t_2 + dist(i_1, n+1)$ and $v(i_2) = t_1 + dist(i_2, n+1)$. Correspondingly, an exchange is promising if $max\{v'(i_1), v'(i_2)\} < max\{v(i_1), v(i_2)\}$; in this case it is regarded as exchange candidate.
Still more intuitively, an equivalent condition can be also formulated using slacks: For this, let $t'_{n+1} = ES_{n+1} = EF_{n+1}$ be the end time of the final activity, i.e. the project duration, computed by forward pass starting with the end times of activities which are eligible or active at the current decision point. Then the slack of i_1 is given by $sl(i_1) = t'_{n+1} - dist(i_1, n+1) \geq 0$ and correspondingly for i_2. Due to exchange one gets new slacks $sl'(i_1) = sl(i_1) + t_1 - t_2$ and $sl'(i_2) = sl(i_2) + t_2 - t_1$ (or nearly so: If at least one of these values is negative, then the corresponding activity becomes "over-critical", i.e. EF_{n+1} is increased; but in this case one gets a deterioration anyway). Then an exchange is promising if the minimal slack is increased, i.e. $min\{sl'(i_1), sl'(i_2)\} > min\{sl(i_1), sl(i_2)\} \geq 0$, i.e. the more critical activity has been made less critical than before. This slack-based criterion can be easily seen to be equivalent to that above using values $v(\cdot)$ and $v'(\cdot)$. This exchange criterion is illustrated by the following example:

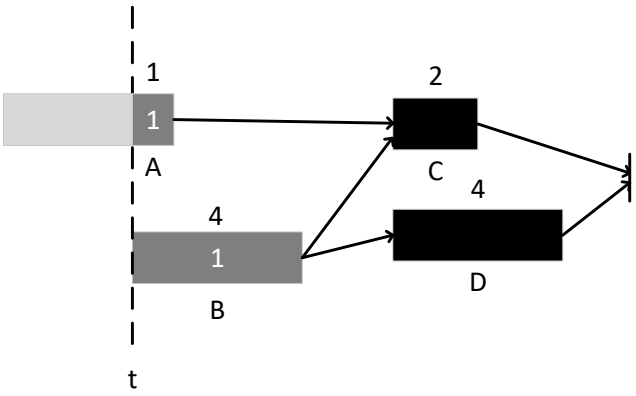


Figure 7: Exchange criterion both activities eligible or active

Here we are at point t in the running project, activity A of type 1 is carried out as repair task which will finish in 1 time unit. Activity B, also of type 1, has just become eligible. Without exchange, the project would finish at $t+8$, whereas with exchange between A and B it would finish at $t+6$ since in this case A would end at $t+4$ and B at $t+1$. Recasted in the criterion above, we have end times $EF(A) = t+1$ and $EF(B) = t+4$ without exchange and further $dist(A, n+1) = 2$ and $dist(B, n+1) = 4$, from which we get values $v(A) = t+1+2 = t+3$ and $v(B) = t+4+4 = t+8$, the maximum of which is $t+8$. With exchange we get values $v'(A) = t+4+2 = t+6$ and $v'(B) = t+1+4 = t+5$, the maximum of which is $t+5$. Since $max\{v'(A), v'(B)\} = t+6 < t+8 = max\{v(A), v(B)\}$, the exchange is advantageous.

A possible exchange can be already taken into account if one activity is not eligible yet. For example, if an activity i_2 is not eligible yet but at a critical position according to forward pass computation with deterministic durations, whereas another activity is not critical and which will not become critical even if it is delayed somewhat. In this case it could be advantageous to delay the end of i_1 until i_2 becomes eligible and exchange the end times then; then activity i_2 can be immediatley finished (since receiving a restored component), thus increasing the slack of i_2, whereas i_1 gets the whole time of i_2 which may not be that long to make i_2 critical. This gives our second case.

Formally, given a pair i_1 and i_2 of the same type τ where i_1 is eligible or active at the current decision point and ending at t_1, whereas i_2 is not eligible yet. For activity i_1 the value is defined as before as $v(i_1) = t_1 + dist(i_1, n+1)$, i.e. the critical path lower bound for $n+1$ given that i_1 finishes at t_1. For activity i_2 compute $EF_{i_2} = ES_{i_2} + p_\tau d_\tau$ and set $v(i_2) := ES_{i_2} + p_\tau d_\tau + dist(i_2, n+1)$. Using the model with expected durations, the exchange will be performed if i_2 becomes eligible at ES_{i_2}. Then the remaining durations are exchanged which are $max\{t_1 - ES_{i_2}, 0\}$ for i_1 (0 if $t_1 \leq ES_{i_2}$, i.e. if the component initially associated with i_1 has been restored) and (expected) $p_\tau d_\tau$ for i_2. This results in new end times $t'_1 = ES_{i_2} + p_\tau d_\tau$ and $t'_2 = ES_{i_2} + max\{t_1 - ES_{i_2}, 0\} = max\{ES_{i_2}, t_1\}$ from which we get new values $v'(i_1) = t'_1 + dist(i_1, n+1)$ and $v'(i_2) = t'_2 + dist(i_2, n+1)$. Again an exchange between i_1 and i_2 is considered as promising if $max\{v'(i_1), v'(i_2)\} \leq max\{v(i_1), v(i_2)\}$.

This case is especially relevant, if activity i has been actually finished and the question arises if its real end is to be delayed until i_2 becomes eligible to perform the exchange. Such a situation is exemplified by the following example:

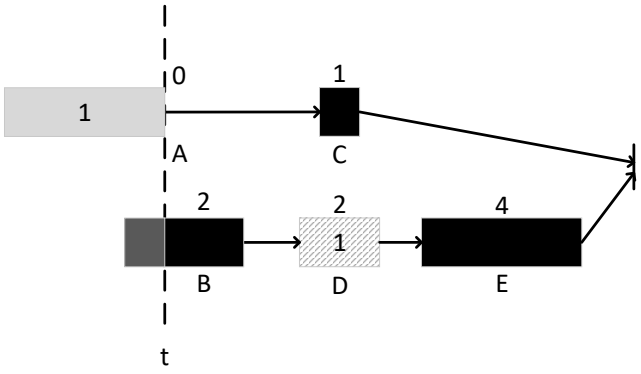


Figure 8: Exchange criterion one activity not eligible yet

Here the the actual repair of the component associated with A has just finished such that this position could be closed and the corresponding activity finished. In this case the project would end (in expectation) at $t + 8$ since fixed activity B would end at $t + 2$, then type 1 activity D would follow (with repair time $\tilde{d}_1 = 4$ and defect probability $p_1 = 0.5$ we get expected duration 2) and finally the project ends with E. On the other hand, for an exchange between A and D one had to wait until $t+2$ when D becomes eligible. Then A gets an expected remaining time of 2 whereas D gets the repaired component from A with remaining time 0. Since C ends (in expectation) at $t + 5$ and E at $t + 6$, the project would finish at $t + 6$ and hence earlier such that an exchange is to be preferred.

4.3 Criterion for Tardiness

For tardiness, the values are defined similarly, adapted to several milestone activities and by regarding the sum of due date exceedances (total tardiness) instead of finish time of the end activity only; given the correspondingly modified values, the exchange criterion will be the same. Again the criterion is formulated for the cases as above: Two activities eligible or active, or one so, the other one not eligible yet.

So for the first case let be given two flexible activities i_1 and i_2 both of type τ and eligible or active, ending at t_1 and t_2, respectively. The value of i_1 is defined by the total due date exceedance implied by the ending of i_1 at t_1 as given by forward pass (and analogously for i_2 with t_2). More precisely, for a milestone activity m_j which is direct or indirect successor of i_1, let $ES_{m_j}^{f(i_1)=t_1}$ be the earliest start time of m_j as given by forward pass starting at t_1 as end time of i_1; further denote by $\bar{d}_{m_j}$ the expected duration of m_j, i.e. d_{m_j} if it is a fixed activitiy or $p_\tau d_\tau$ if it is flexible of type τ. Then m_j will finish at $EF_{m_j}^{f(i_1)=t_1} = ES_{m_j}^{f(i_1)=t_1} + \bar{d}_{m_j}$, resulting in an exceedance $max\{ES_{m_j}^{f(i_1)=t_1} + \bar{d}_{m_j} - dd_{m_j}, 0\} =: v(i_1, m_j)$. Summing over all milestone activities succeeding i_1 yields the value of i_1, i.e. $v(i_1) := \sum_{m_j \in \mathcal{M} \cap \bar{S}_{i_1}} v(i_1, m_j)$. The value of i_2 is defined analogously. An exchange between i_1 and i_2 corresponds to an exchange of their end times, i.e. giving $f'(i_1) = t_2$ and $f'(i_2) = t_1$. Using these new end times, the new values of both activities are computed as before, giving values $v'(i_1)$ and $v'(i_2)$. Now, as already for makespan, the criterion for a promising exchange can be defined as satisfying $max\{v'(i_1), v'(i_2)\} < max\{v(i_1), v(i_2)\}$.

Further we also regard again the case where one activity is eligible or active, whereas the other one has not become eligible yet. So let be i_1 and i_2 activities both of the same type τ, i_1 having become eligible but not finished with current end time t_1, whereas i_2 is not eligible yet. For i_1 the value $v(i_1)$ is computed as in the first case. Regarding i_2, the time at which i_2 will become eligible is estimated, as for makespan, by computing earliest start time ES_{i_2} starting with the given end times at the current decision point. Then, using $t_2 := EF_{i_2} = ES_{i_2} + \bar{d}_{i_2}$, the value $v(i_2)$ of i_2 can be computed as before by determining the exceedance of each milestone activity m_j which is (in)direct

successor of i_2 as implied by i_2 ending at t_2, and taking the sum over all these exceedance values, i.e. as $v(i_2) = \sum_{m_j \in \mathcal{M} \cap \bar{S}_{i_2}} max\{ES_{i_2} + \bar{d}_{i_2} + dist(i_2, m_j) + \bar{d}_{m_j} - dd_{m_j}, 0\}$. Then, as in the other cases, the criterion for a promising exchange is to fulfill $max\{v'(i_1), v'(i_2)\} < max\{v(i_1), v(i_2)\}$.

4.4 Further details

Application of the criterion for the objective under consideration at given decision point yields a set of exchange candidates. However, this set may contain conflicting pairs (i_1, i_2) and (i_1, i_3) which cannot be realized simultaneously. To resolve such conflicts, the potential exchange pairs are prioritized according to the following criteria, where ties are broken by the subsequent ordering criterion (for quantitative criteria, higher values get higher priority):

- pairs where both activities are eligible or active have higher priority
  than those where only one activity is (whereas the other one has not become eligible yet)
- $max\{v(i_1), v(i_2)\}$ (criticality)
- $max\{v(i_1), v(i_2)\} - max\{v'(i_1), v'(i_2)\}$ (improvement)
- $-i_1$ (i.e. lower i_1 has higher priority)
- $-i_2$ (ditto)

This yields a priority list of exchange candidates $(i_1^1, i_2^1), (i_2^1, i_2^2), \ldots, (i_1^k, i_2^k)$ which are used as follows: An exchange (i_1^j, i_2^j) is carried out (i.e. the end times of i_1^j and i_2^j are exchanged) if both activities are eligible or active and there is no conflicting pair of higher priority, i.e. no (i_1^h, i_2^h) with $h < j$ and $\{i_1^h, i_2^h\} \cap \{i_1^j, i_2^j\} \neq \emptyset$. This gives a subsequence $(i_1^{j_1} i_2^{j_1}), \ldots, (i_1^{j_l} i_2^{j_l})$ of exchanges which are actually realized and which altogether form a permutation $\pi = (i_1^{j_1} i_2^{j_1}) \cdots (i_1^{j_l} i_2^{j_l})$ of eligible or active activities. Furthermore, each activity which has just reached its finish time (and hence could be finished) and which only occurs in a pair with some activity which is not eligible yet (and hence not in the subsequence just defined) is delayed and hence added to the delay set D.
Both these elements yield a fully specified decision pair (π, D) which is realized at the current decision point with the steps described earlier.

Altogether the main steps of our solution procedure can be summarized as follows:

Summary of procedure:

Start at $t = 0$ with the empty schedule as initial decision point
Being at decision point $\delta = (\mathfrak{p}, t)$

- determine the set of eligible activities E_δ
- start all eligible fixed activities at t
- determine the decision pair $Dc_\delta = (\pi, D)$ for δ
- carry out the exchanges given by π
- delay the activities which could be finished but are contained in the delay set D
- determine the next decision time and hence the next decision point δ'
- repeat with $\delta := \delta'$

Stop if the final activity is reached and hence a complete schedule is constructed.

5 Computational Study

5.1 Experimental setting

5.1.1 Instance generation

The scheduling policy based on the proposed exchange criteria has been tested by simulation on a large number of instances. The aim was to quantify the improvement resulting from the application of the exchange policy compared with no-exchange in terms of given objective (makespan or tardiness). For this, the instances have been generated by a full-factorial design taking into account the following parameters:

Network Complexity: Network complexity is "the average number of non-redundant arcs per node including the dummy activities" [Kolisch and Sprecher 1996], or, equivalently, the average number of direct successors (in a network without redundant arcs) and hence a measure for the density of the precedence relation. We will consider the values $NC \in \{1.5, 1.8, 2.1\}$, as given by the RCPSP test instances from which the underlying network structure of our test instances has been taken as described below.

Type Combination: The type combination of an SAEPSP instance is the sequence $\bar{a} = a_1 a_2 \dots a_{\bar{\tau}}$ specifying the number of positions for each type, i.e. $a_\tau = |A_\tau|$ for each type τ. We will consider the following nine type combinations with up to 5 types:

(5 5 5 5 5) ; (10 10 10 10 10) ; (15 12 10 8 5) ; (20 15 10 5) ; (20 10) ; (10) ; (20) ; (30) ; (40)

(the first three combinations for five types, then one for four types, one for two types, four for one type; the first two combinations each with a uniform number for all types; the second one together with the next two with increasingly steeper decreasing numbers; the last four combinations with increasing numbers for the only type)
Especially in the diagrams we will often refer to type combinations by their position in the order given above, thus, for example, "type combination 3" meaning (15 12 10 8 5) and so on.

Defect probabilities: For our study we assume that in each SAEPSP instance the defect probability is the same for all component types, i.e. $p_1 = \dots = p_{\bar{\tau}} =: p$, where p can attain values $0.1, 0.2, \dots, 0.8$ (i.e. 10 - 80% in steps of 10%).

Altogether this yields $3 \cdot 9 \cdot 8 = 216$ parameter combinations. Since the effect of network complexity will turn out to be only small or moderate, we will often take averages over NC-values and focus on the resulting 72 type/probability combinations (short: t/p-combinations). In the figures below we will order these t/p-combination primary by their type, secondary by their probability p.
The number of non-dummy activities is kept at $n = 60$ (also as given by used RCPSP test instances; note that $60 \geq \sum_{\tau=1}^{\bar{\tau}} a_\tau$ for all type combinations listed above).
Note that for the deterministic version in the preceding paper [Berthold 2021], the term "type combination" referred to the double sequence $b_1 \dots b_{\bar{\tau}} / a_1 \dots a_{\bar{\tau}}$ where a_τ was (as here) the number of postions of type τ and b_τ the numbers of initially defect ones among them (which were known in the deterministic model). Here, in the stochastic setting, instead of these latter values only their expected values ($p_\tau a_\tau$ instead of b_τ) are known such that a type combination in the former sense would correspond best to $p_1 a_1 \dots p_{\bar{\tau}} a_{\bar{\tau}} / a_1 \dots a_{\bar{\tau}}$, or simply $p / a_1 \dots a_{\bar{\tau}}$ by removing redundancies and taking our choice for one uniform probability. However, since p and $\bar{a}$ are treated here as independent parameters, we will narrow the notion "type combination" to that part of information which refers to numbers of flexible positions.
For each parameter combination $(NC, \bar{a}, p)$ from those 216 ones, a number of 4000 instances have been generated as follows:

At first, to get the underlying network structure, instances from the PSPLIB, a benchmark instance

library for the RCPSP and some of its variants [Kolisch and Sprecher 1996, online available under www.om-db.wi.tum.de/psplib/], have been taken. Here the set J60 contains 480 RCPSP instances on 60 nodes generated according to a full-factorial design involving network complexity and two resource-related parameters (resource factor and resource strength). For each of the values $NC \in \{1.5, 1.8, 2.1\}$ this set contains 160 instances, from which 20 instances have been chosen. Omitting from these the resource-related information (resource consumption and capacities) yields 20 standard CPM project networks for each NC value. For each such CPM network, disjoint subsets $(A_\tau)_{\tau\in\mathcal{T}}$ with $|A_\tau| = a_\tau$ are chosen from its 60 non-dummy activities. The activitiy i_τ chosen first for each τ determines the type-specific duration d_τ for that type via $d_\tau := d'_{i_\tau}$, where d'_{i_τ} is the duration of i_τ in the project network (and in the RCPSP instance from which it is derived). Adding finally the uniform defect probability p completes the SAEPSP instance (still without having specified the objective). In this way for each CPM network (and hence for each original RCPSP instance) 200 such set families $(A_\tau)_{\tau\in\mathcal{T}}$ are randomly chosen according to given $a_1 \dots a_\tau$. Altogether this yields $20 \cdot 200 = 4000$ instances for each parameter combination $(NC, \bar{a}, p)$ (20 CPM project networks for given NC value and 200 instances randomly derived from it (each by randomly choosing $(A_\tau)_{\tau\in\mathcal{T}}$) according to $a_1 \dots a_{\bar{\tau}}$, finally complemented by the uniform defect probability p)
As objectives makespan and tardiness have been regarded. Declaring makespan as objective to any of the instances generated as above immediately gives a full problem instance of the SAEPSP (with makespan objective), whereas for tardiness additionally a set of milestones and corresponding due dates must be still generated to get a fully specified problem instance (for the SAEPSP with tardiness objective). For this, a set $M' \subseteq \{1, \dots, 60\}$ with $|M'| = 8$ is randomly chosen; adding further the dummy end activity as natural milestone gives the full set of milestone activities $M := M' \cup \{n+1\}$. Their due dates are determined according to an additional parameter called *due date factor* which controls how tight due dates are set; the lower the due date factor, the tighter the due dates. More precisely, given a value $df \in \{0.8, 1.0, 1.2\}$, due dates are determined as follows: For each milestone $i \in M$ let EF_i be the earliest finish time computed with expected durations $p_\tau \cdot d_\tau = p \cdot d_\tau$ for flexible activities of type τ (as already seen in the description of the solution approach, but starting the forward pass simply at $t = 0$ with activity 0). Then i's due date is set to $dd_i := df \cdot EF_i$ whence a lower due date factor leads to tighter due dates indeed. Thus, if tardiness is considered as objective, the due date factor is a further parameter to get a fully specified instance for the SAEPSP (with tardiness objective).

These are the instances on which the exchange policy has been subsequently applied; depending on the objective under consideration, makespan or tardiness, the corresponding exchange criterion has been used. Since applying the policy on an instance takes only negligible time, compution time will not be relevant for our study. Instead, the efficiency and performance of our approach has been measured by the improvement of the objective resulting from application of the exchange policy compared with no-exchange, i.e. where each flexible activity is simply performed in its initial state. For this, the exchange policy is carried out first on an instance; the initial states which are revealed during this run, namely each time a flexible activity becomes eligible, are stored, which admits afterwards, in a second run by forward pass, to determine the duration if the exchange policy had not been applied, i.e. if each flexible activity had been simply performed in its initial state and each activity had been started immediately after becoming eligible. Then the exchange policy has turned out successful for given instance and realization of random state variables, if it has led to a better objective (lower makespan or tardiness) than without having made any exchange. I.e., if z is the objective value for exchange and z' for non-exchange for given instance (and realization of the random states), the difference $z' - z$ yields the improvement due to the application of the exchange policy (which also may be negative, corresponding to a decline). Based on these improvement values for single instances we will regard average relative improvements, proportions of improvements ($z' - z > 0$) and declines ($z' - z < 0$) etc. for full parameter combinations as well as t/p-combinations (i.e. additionaly averaged over NC

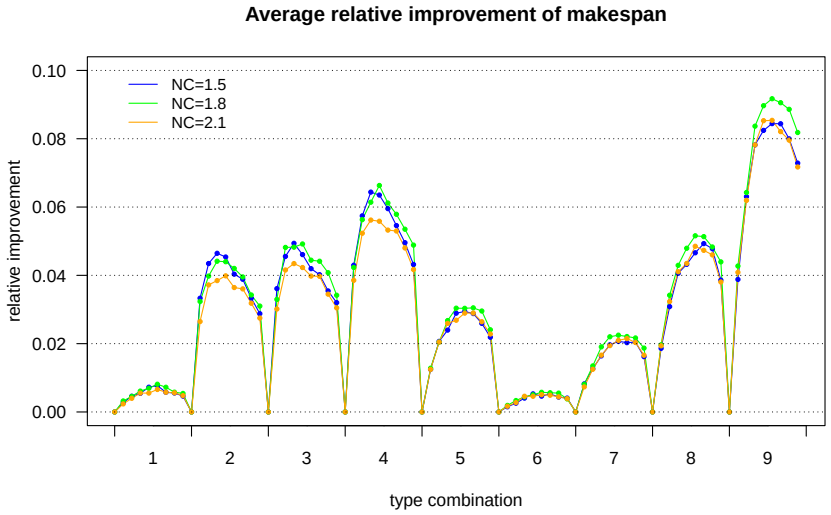
values).

Having thus described the general framework for the computational study so far, we can now turn to the results.

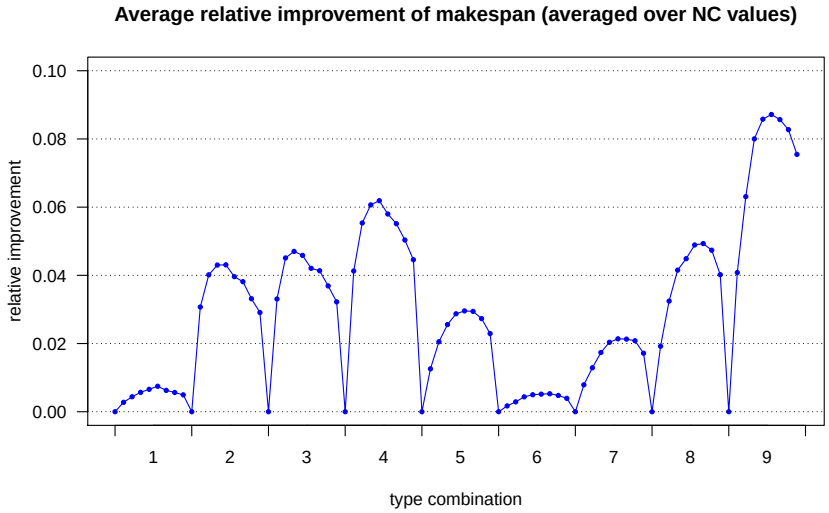
5.2 Results

5.2.1 Makespan

We start with the results for the makespan objective, for which essential dependencies become already visible by considering the average relative improvements:



First it can be noted that on average the exchange policy indeed leads to improvements for all parameter combinations. These improvements vary between minimal (<1%) and 9%, 3-6% is a typical value for many combinations; the precise value strongly depends on the type combination, less on the defect probability and only marginally on the NC value (at least for the regarded ones). Namely, as far as the last parameter is concerned, the differences between the NC values are rather small or virtually non-existent (standard errors are about 0.1-0.2% for the larger improvement values); for example, for $NC = 2.1$, $p = 0.3 - 0.5$ and type combinations 2-4 (with 4-5 types) the improvement is slightly smaller than for the lower NC-values (for which there is almost no difference), whereas for type combinations 6-9 (one type) the improvements for medium value $NC = 1.8$ are slightly better than for both other ones (for which they are also almost the same). But since also these maximal differences in relative improvement do not exceed 1% and there are no qualitative differences, we can confidently take the averages over the NC values and focus on the dependencies on t/p combinations, the pattern of which becomes still better visible this way:



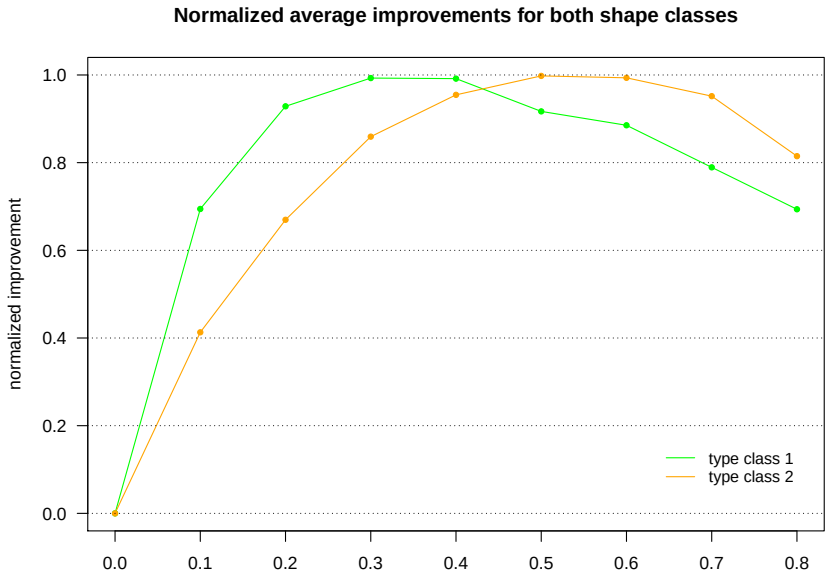
Going now somewhat more into details, we see that highest improvement is reached with 8.7% for type combination (40); followed by 6.2% for (20 15 10 5); 4-5% for (10 10 10 10 10), (15 12 10 8 5) and (30); 3% for (20 10); 2% for (20); and finally 0.7% for (5 5 5 5 5) and 0.5% for (10) (since probabilities form a more secondary structure, we will regard them below and focus here only on the maximum values for each type combination, also since they are nearly attained for several p values). Thus, not surprisingly, more positions generally lead to higher improvements since there are more opportunities for exchanges. However, less clear, the number of types and especially how the positions are distributed among them are not less important: For example, (10 10 10 10 10) and (15 12 10 8 5) each have 50 flexible positions, but a similar improvement as (30) with only 30 positions and considerably less than (40) with 40; (20 10) is only between (20) and (30); and (5 5 5 5 5) has only a fraction of what (25) would have. So, given a fixed number of positions, the improvement is higher the more uneven the positions are distributed among the types; having m flexible positions, type combination (m) would yield a higher improvement than $(m/2\ m/2)$ and this a higher than $(m/k \dots m/k)$ for $k > 2$. This can be easily explained as exchanges are only possible between positions of the same type such that distributing them onto several types reduces exchange possibilities. In this respect it is interesting to consider the tradeoff between number of types and number of positions: How much the number of positions must be increased if the positions are distributed among more types to keep the improvement the same? For this the already compared type combinations (10 10 10 10 10) and (30) with essentially the same improvement give an example: Here distributing the position on 5 types is nearly compensated by increasing the number of positions by two-third; and for (5 5 5 5 5) and interpolated (25) the discrepancy is still larger. For short, exchange becomes more effective for few types with more positions than for more types with only a moderate number of positions

This higher effectivity of exchange towards concentration and asymmetry of the type combination cannot be only observed by opposing those extreme distributions, but also for intermediate ones: So for (20 10) and (30), and also the sequence (10 10 10 10 10), (15 12 10 8 5), (20 15 10 5). For the latter it is interesting to see that the increase from the first to the second one is only marginal, but from the second to the third significant, even though the second is intermediate w.r.t. to the "steepness" of the type combination, i.e. the difference between subsequent numbers: 0 for the first, 2.5 for the second (on average), 5 for the third type combination. This may indicate a kind of superlinear dependency on the asymmetry, if this one is measured, for example, by just that steepness: Weak asymmetries still contribute proportionally less to the improvement, but it increases faster the more asymmetry has been has been already reached (at least to a certain point).

Also remarkable and perhaps also a sort of such superlinear dependency is the accelerating increase for the sequence (10)...(40) (with maximal values 0.5, 2.1, 4.9, 8.7%) and the jump from 0.7 to 4.3%

for (5 5 5 5 5) and (10 10 10 10 10) (for which one may initially expect more a simple doubling of improvement). The former values even seem to indicate a (nearly perfect) quadratic relationsship (with $improvement(a_1)$=$0.54 \cdot (a_1/10)^2$ %), the latter ones may be also roughly consistent with such an dependency. In any case, generally it seems that the opportunities for improvement increase faster than the sole number of positions, which is surely at least partially explainable by the fact that the number of potential exchange pairs increases quadratically indeed. Moreover, if there are only few positions, the probability is high that they become eligible at too different times such that waiting of earlier positions for later ones for a possible exchange may not be worthwile anymore. As a consequence, exchange may become effective only if a certain limit of positions is reached, in which case namely the density of positions becomes high enough to admit advantageous exchanges.

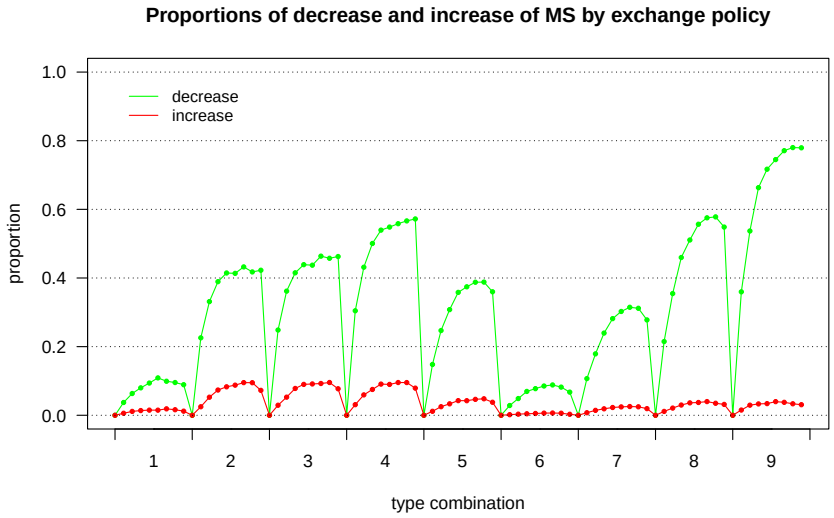
Having considered the effect of type combinations, now we turn to the probabilities and their influence on the average improvements. Generally this dependency is relatively uniform for all type combinations: Increasing the probability, at first the improvement value steeply increases, reaches its maximum between 30-50% and then decreases somewhat for higher probabilities. A more or less near-maximal value is already attained 10-20% under and 20-30% above the actual optimal probability, such that the maximal value is not very sensitive w.r.t. to optimal probability, whence a near-optimal improvement can be reached for a broader range of probabilities.
A closer look further reveals two different classes of type combinations, apparently depending on the number of types: On the one hand type combinations 2-4 (with 4-5 types), where the maximal value is nearly reached already at 20%, the maximal value at 30-40%, after which the value decreases moderately and rather steadily for higher p; on the other hand type combinations 5 and 6-9 (2 and 1 type(s), respectively), for which a nearly maximal value is reached only at 30-40%, the actual maximum at 50-60% and accelerating decrease afterwards (but value for 80% still rather high). Visually this results in two different curve shapes, in the first case asymmetric with steep increase and moderate, steady decrease (somewhat "cliff"-like), in the second case a more symmetric parabola-like shape. To make this different behaviour for these two type combinations classes better visible and comparable, one can normalize the values for each type combination (maximal value corresponding to 1) and take the average for each class. This yields the following picture:



Clearly these curves show the behaviour and the differences described above, on the one hand the shift towards higher probabilities for the second class, on the other one the shapes "cliff"- vs. parabola-like (which, by the way, is indeed nearly a parabola with correlation coefficient $r = 0.99$). Resulting from

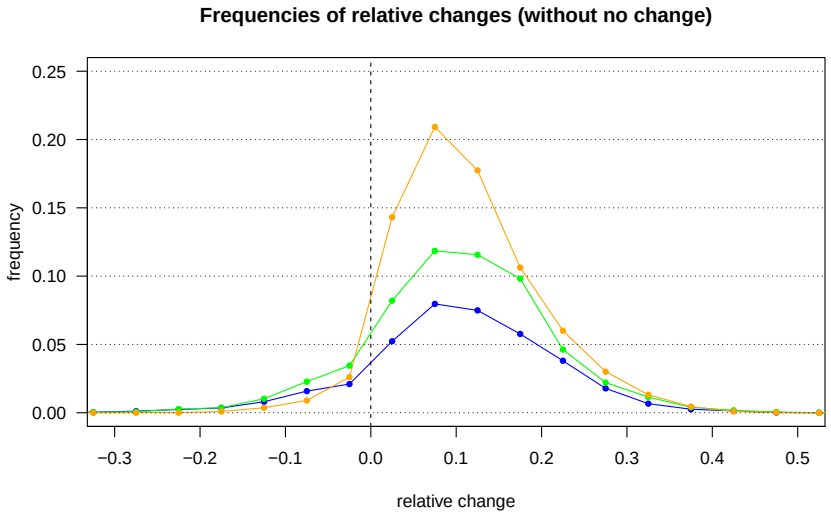
this shift and the different shapes, larger discrepancies occur for small probabilities $p = 0.1$ (relative difference 0.28) and $p = 0.2$ (relative difference 0.26), for other ones the differences are about 0.15 or lower. Due to the broad peaks a value of 90% of maximal improvement is reached for 20-60% for the first class, and 35-75% for the second one. Thus, as mentioned, near-maximal improvement values can be achieved for a broader range of probabilities.

Due to the stochastic nature of the problem, the applied policy does not always lead to an improvement, but sometimes also to a deterioration or no change at all. Corresponding proportions can be read out from the following graph (no-change proportions as complement of increase and decrease; note that a decrease of makespan is an improvement):



Apparently, the proportion of improvements largely shows the same pattern as already seen for the average values above, with 40-60% being a typical value for many combinations. The proportion of deteriorations are lower by a factor 4-6 for the first type combination class, and by 10-20 for the second one. It can be also seen, that the policy often does not lead to a change at all, the proportion of which frequently reaches 40% or more (100%-(increase+decrease)). A significant amount of such non-changes seems inevitable for this stochastic problem, and especially the makespan objective may be prone to it, since to reduce makespan, one has to remove a defective component from (what would be, without exchange) the critical path and move it to a position without generating a new critical path; however, in general there is no critical path, only more or less probable candidates depending on the outcomes of stochastic variables; and further there may be no flexible positions on such a candidate. In other words, one has to have some luck to have a chance to improve at all, and then further luck to make the right decisions. Taking this into account, no-change rates of several ten percent seem conceivable. This also explains the average improvements of "just" several percents, much more may be not possible even for a near-optimal, but possibly complicated policy, at least for the makespan objective. We will see, that the tardiness objective admits considerably higher improvements since there several points and hence different paths are taken into account such that the probability to actually realize an improvement is considerably higher.
Looking more precisely, also somewhat different shapes are recognizable, compared with those for the averages above and also among each other, and again corresponding to both type combination classes: So for type combinations 2-4 (i.e. type combination class 1 from above) the proportion of improvements increase up to $p = 0.3$ but then remains nearly constant or slightly increases up to $p = 0.8$ (in contrast to moderate decrease for $p > 0.4$ for the averages above), whereas for type combinations 5 and 6-9 (class 2) an increase up to $p = 0.6 - 0.7$ and a slightly lower value for $p = 0.8$ can be observed.

Thus for both classes the maximal proportion of improvement is attained for higher probabilities than for the average improvements. As a consequence, the relative contribution of improvements decreases with higher probabilities.
However, the proportions of decreasings and increasings only convey a rather coarse idea of the underlying distribution and admit different possibilities how the changes are really distributed. For a still more detailed picture the distribution of relative changes is exemplified for three t/p-combinations ($t/p = 2/0.2, 4/0.3, 9/0.5$). Points in the following figure correspond to classes of breadth 0.05, the proportion of precisely 0%, which would be far higher than the contribution of any 5%-interval, is skipped here:

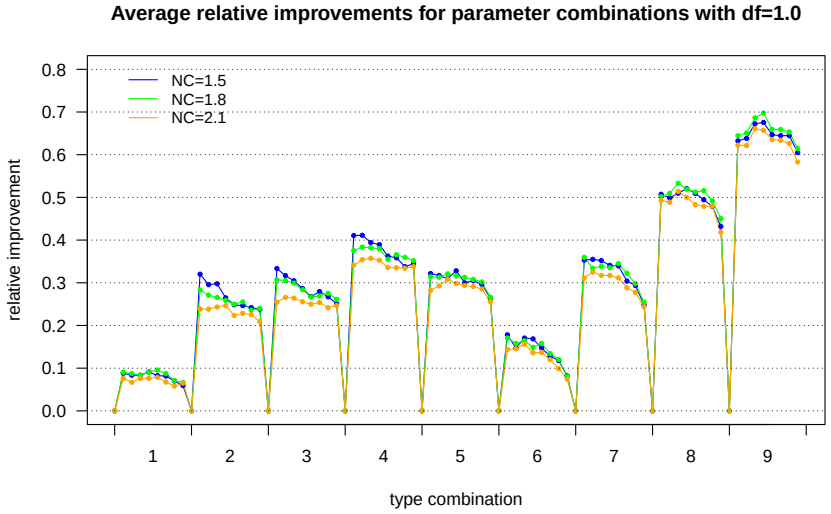


Corresponding to the results above, the main part of the distributions clearly concentrates in positive range. All three distributions have approximately the shape of a bell-curve. The maximal values are roughly proportional to the average relative improvements for these t/p-combinations (factors 2/2/2.4), and all are attained at about 0.1 (taking also the second largest value into account which is only slightly lower especially for 2/0.2 and 3/0.4). This position is also consistent with the results given so far: Given, for example, an average improvement of 6% and a proportion of 40% of no-changes (as for 4/0.3), one would expect average improvements of 10% for the cases where a change actually does occur. The breadth (for example measured at the 50%-level of the maximum value) is also similar for all three combinations, only somewhat smaller for 9/0.5, which may be also reflected by steeper decreases and changed proportions towards the edges of the distributions.
A somewhat closer look reveals also some deviations from a symmetric bell shape and the proportionalities in the center: So there is some asymmetry in form of steeper decrease towards negative values, especially visible at the relatively low frequencies, figuratively speaking the "dents" at frequency class [-0.05,0), which are probably an artefact of removing the proportion of no-changes here. This decrease is especially distinct for 9/0.5, for which in the negative range the frequencies become even smaller than for the other combinations (as already indicated above by the low proportion of deteriorations for type combination class 2). On the other side, for the higher positive changes the order between the combinations is kept, but the proportions there are also changed compared with the center (possibly partially also an effect of the allegedly smaller breadth of the distribution of 9/0.5).
However, most of these differences are more secondary as concerning only the edges of the distributions, i.e. values with low frequencies. Thus the main and most obvious difference can be simply attached to the peak values (at least for the three regarded combinations). Given the other correspondences w.r.t. shape, the position of the maximum and the breadth, the average improvement is

largely determined by that peak values, which also explains the relative proportionality with the average relative improvements mentioned above. Altogether it should be also noted that a shift towards higher values of changes for combinations with higher average improvements - which would be also a conceivable possibility - cannot be observed for the chosen t/p-combinations, whence the increase is largely driven by the increase of maximal frequencies alone.
Thus having got an overview for makespan, now we turn to the results for the tardiness objective.

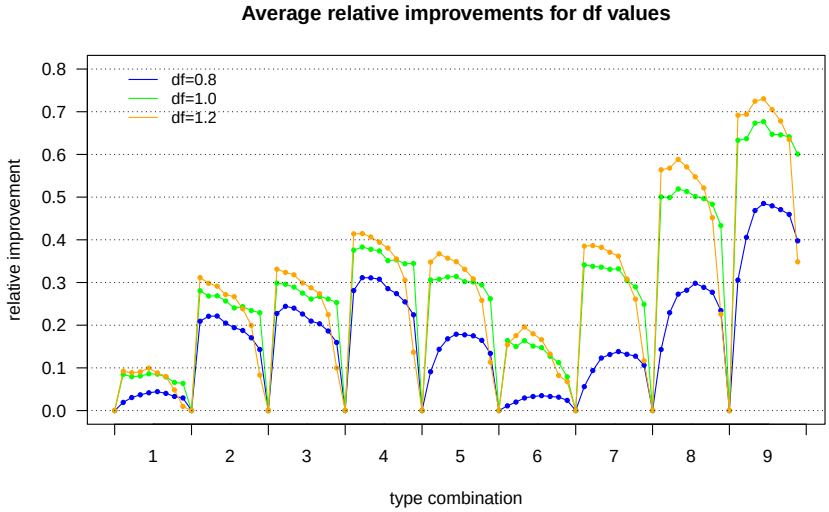
5.2.2 Tardiness

As we will see, qualitatively the results for tardiness are in many respects similar to those for makespan; however, the improvements are considerably higher and further, specific for tardiness, there are differences depending on how tight the due dates are set. Concerning relative improvement due to exchange, note at first that it may be undefined since tardiness may have value 0 (which can especially occur if due dates are set more loosely). More precisely, if i is an instance and T_i^e and T_i^{ne} are the tardiness values which occur by applying exchange and no exchange respectively (for some realization of random variables), then the relative improvement $(T_i^{ne} - T_i^e)/T_i^{ne}$ is undefined if T_i^{ne} is 0 (and, even if not, becomes artificially high if it is only slightly greater than 0). Thus, as substitute for instance i (and corresponding realization) we chose the value $(T_i^{ne} - T_i^e)/(\sum_{j=1}^n T_j^{ne}/n)$ (where the sum runs over the instances of the corresponding df/tp combinations, the number of which is $n = 4000$); i.e., the tardiness improvement for an instance (and realization) is set in relation to the average tardiness without exchange for that df/tp-combination. Then the resulting average relative improvement corresponds to $\frac{1}{n}\sum_{i=1}^n (T_i^{ne} - T_i^e)/(\sum_{j=1}^n T_j^{ne}/n) = (\sum_{i=1}^n T_i^{ne} - T_i^e)/\sum_{i=1}^n T_i^{ne} = 1 - \sum_{i=1}^n T_i^e / \sum_{i=1}^n T_i^{ne}$. Using these modified relative improvements, again we consider at first the dependency on NC values, here for $df = 1.0$:



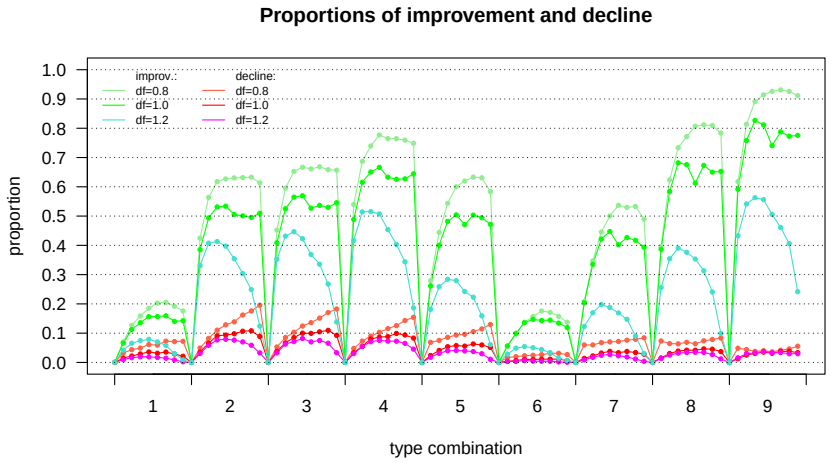
The largest, but still moderate differences occur for type combinations 2-4 and $p = 0.1 - 0.3$ for which the improvements are somewhat higher for lower NC values (for type combination 2 about 0.31 for $NC = 1.5$ compared with about 0.24 for $NC = 2.1$); as a consequence, while the values are somewhat decreasing for $NC = 1.5$, they are nearly constant for $NC = 2.1$, such that the improvement is essentially independent of p over the large range [0.1,0.8]. For other t/p combinations the differences between NC values are rather small. For $df = 0.8$ (i.e. tight due dates) the differences between NC values are somewhat larger, for $df = 1.2$ (loose due dates) they are comparable with 1.0. However, as for makespan, altogether they are similar enough such that in the following we will restrict to the averages over NC values. The corresponding average improvements, now for all df values, then look

as follows:



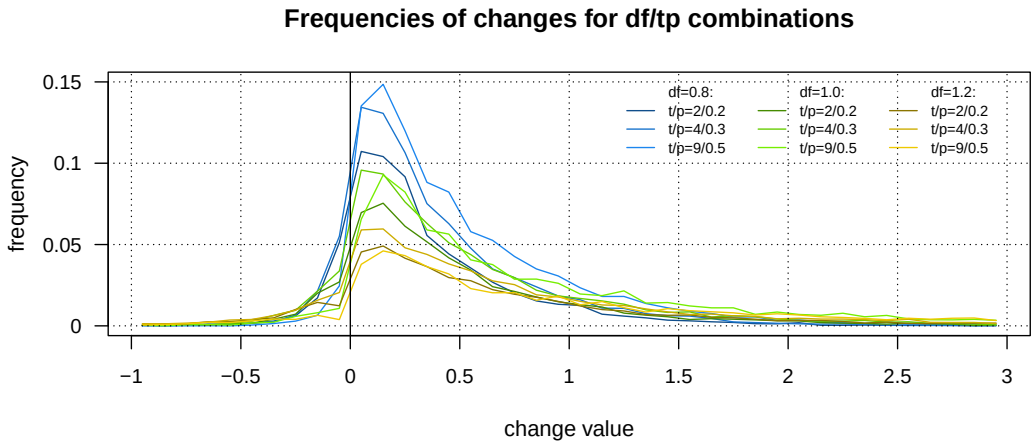
At first and most important, one can note that the average relative improvements are considerably larger than for makespan and typically vary between 0.2 and 0.4 for medium type combinations; the proportions between different type combinations, however, are rather similar to those already seen for the makespan. One reason for the higher improvements is the same as for makespan, but read conversely: Since now there are several milestones which contribute to the objective, there are more exchange opportunities which may actually have an effect on the objective; it becomes less probable that the possible and performed exchanges do not affect a single critical path as for makespan.
On the other hand, whereas makespan is measured from 0 to the end of the project, for tardiness the reference values, i.e. the due dates, are already set near the expected end times of the milestones, such that the relative improvement is magnified even if the absolute improvement per milestone may be comparable with the improvement of the makespan. For example, a makespan of 27 with exchange and 30 without corresponds to a relative improvement of 10%, whereas we would get 50% if tardiness is regarded with the last activity as only milestone and due date 24.
New for tardiness and also significant are the dependencies on the due date factors, i.e. on the tightness of the due dates: In general, the relative improvements increase with higher df values, i.e. more relaxed due dates. This can be explained by taking into account that tighter due dates generally lead to higher tardiness values such that comparable or even somewhat higher absolute improvements translate to lower relative improvements. Even if in many cases the differences are small (especially between $df = 1.0$ and 1.2), higher discrepancies can be observed between 0.8 and the other df values for type combinations 5+(7-9), i.e. type combination class 2, for only one or two types: Here the maximal improvement values for $df = 0.8$ are only about 50-70% compared with those for 1.0 and 1.2, and furthermore, qualitatively, the shape (i.e. the dependency on p) is different, more parabola-like with near-maximal values for $p = 0.3 - 0.7$, whereas otherwise the improvements are near-maximal for $p = 0.1 - 0.6$ or even up to 0.8 (typically slightly decreasing towards higher p). For several types, i.e. type combination class 1, the values for $df = 0.8$ are somewhat smaller than for 1.0, but with a similar shape. Minor differences can be also observed between otherwise similar $df = 1.0$ and 1.2, where $p = 0.8$ for $df = 1.2$ has a value less than one half or even one third of the maximum, whereas for $df = 1.0$ this value still remains near-maximal as well. This only weak dependency on p especially for $df = 1.0$ and here particularly for several types (i.e. type combination class 1) results in a plateau-like, only slightly decreasing shape. This is also different to the "cliff"- or even parabola-like shape observed for makespan; in some sense it may be an extreme of the "cliff"-shape, resulting from

having several instead only one reference point. Generally, compared with makespan, there is a clear tendency for tardiness towards independency from defect probabilities, most pronounced for $df = 1.0$, $df = 1.2$ with $p \leq 0.6$, and $df = 0.8$ with several types.
But as for makespan, behind those average values a considerable variation can be found such that we also take a look at the proportions of improvements and declines. Here the differences between NC values are even smaller than for the average values, such that we will regard only the averages over NC values:

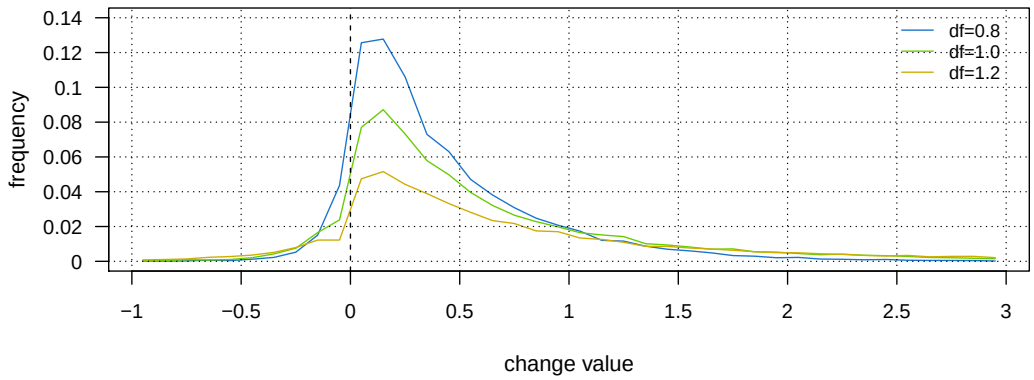


For each df value, the proportions between type combinations are roughly similar to the average improvements above and also to the corresponding proportions for makespan, relatively somewhat lower only for type combinations 8 and 9. Generally the proportion of improvements (and declines) decrease with higher df value. The shape for $df = 0.8$ and 1.0 is similar with steep increase and near-maximal values for $p \geq 0.2 - 0.3$; for $df = 1.0$, this maximal level is lower by 10-15%, whereas for $p \leq 0.2$ the difference is smaller. Values for $df = 1.2$ are considerably lower especially for one or two types, and, still more remarkable, the shape is very different with maximal values only at $p = 0.2 - 0.4$, but accelerating decrease for higher p (whereas near-maximal levels are kept for the other df values). Thus, surprisingly, the dependency on tightness is just converse to that for the averages, where highest improvements were reached for $df = 1.2$. This counterintuitive observation can be explained by taking into account that not only the relative improvement becomes higher for loose due dates (as already argued above), but also the proportion of instances with tardiness 0 already without exchange (and for which, hence, exchange cannot lead to a further improvement). As a consequence, for loose due dates the exchange policy cannot lead to an improvement for a higher percentage of instances, which also reduces the proportion of improvements. So, altogether, we have the somewhat paradoxical situation that for loose due dates a significant proportion of instances cannot be improved at all by exchange (or otherwise), but for those which can, the relative improvement is higher on average and even high enough to compensate the loss due to those unimprovable instances (and conversely for tight due dates: less instances which cannot be improved, but also lower relative improvements). This situation becomes clearer if frequency distributions are regarded next.

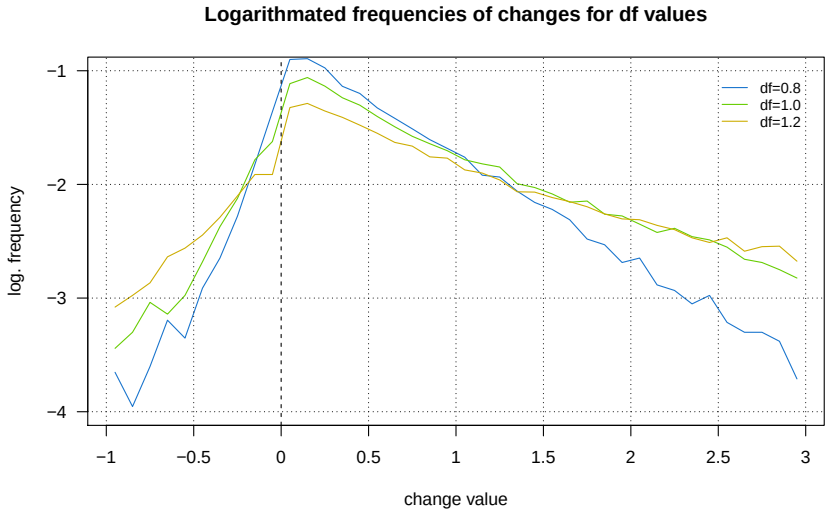
As for the makespan objective we finally consider the frequency distribution of relative changes. For this we choose the same t/p-combinations as before (2/0.2, 4/0.3, 9/0.5), for each of which three df-values are possible, hence resulting in nine df/tp-combinations. As noted above, we use as substitute the value $(T_i^{ne} - T_i^e)/(\frac{1}{n}\sum_{j=}^{n}(T_j^{ne}))$ for improvements. This yields the following picture (again without no-changes; bin size 0.1, i.e. value at 0.05 is the frequency of (0,0.1], at 0.15 that of (0.1,0.2] and so on):



It can be read out that the frequencies are higher the lower the df values (i.e. the tighter the due dates are set): Whereas for $df = 0.8$ the maximum reaches values 0.11-0.15, for $df = 1.0$ we have 0.08-0.10, and for $df = 1.2$ only 0.05-0.06. Qualitatively, this also corresponds to the observed proportions of improvements and deteriorations which are higher for lower df-values (tighter due dates).
The averages for the due date factors (i.e. averaged over considered t/p-combinations) show this still more clearly:



However, beyond these different absolute values the general behaviour is rather uniform for all df/tp-combinations (which also justifies to regard the averages for the due date factors): Towards negative values (deteriorations) there is a steep decrease, whereas towards higher positive values it is slower. This results in an asymmetric shape (hence qualitatively different from the rather symmetric one observed for makespan) where the positive side overweighs the negative one by far. The decreases at both sides seem to be at least roughly exponential, which is also confirmed by a look at the logarithmated frequency values:



Indeed each curve essentially consists of only two nearly-linear segments, a steeper increasing and a slower decreasing one, the transition between which (corresponding to the area around the maximum) is very short to nearly non-existent. Regarding here especially the right part towards positive values (frequencies of improvements), it can be seen that the deviation from a perfectly linear behaviour is indeed rather small and still smaller for the individual df/tp-combinations (see Appendix 8.2), such that the frequency values actually decrease exponentially.
However, there are significant differences w.r.t. slope coefficients: So the slope is steeper for lower df-values (tighter due dates) whereas it is considerably shallower for higher ones (loose due dates). As a consequence, even if the frequencies around the maximal value are highest for $df = 0.8$, due to the steeper decrease the frequencies are roughly the same at about 1-1.5 (but there already at a rather low level) and are reversed for still higher change values.
The considerably steeper decrease especially for $df = 0.8$ also explains in more detail the higher relative improvements for higher df-values 1.0/1.2, which, as observed, stands in some contrast to their lower proportions of improvement: Indeed, even if the frequencies are generally low for higher change values, they are already almost zero for $df = 0.8$ where for $df = 1.0/1.2$ they are still significant enough to contribute, weighted by the higher change values, to the average improvement such that their lower contribution of higher frequencies at low change values (for just these df-values) is compensated and even reversed. Also note, that the maximal frequencies are already attained for change values 0.1-0.2 such that due to these low weights their contribution is only rather moderate or low and can be more than compensated by the contribution of high change values more easily. Altogether, both these effects, still significant contribution of high change values and attaining the lower maximal frequencies already at low change values result in higher relative improvements for $df = 1.0/1.2$, even if their proportion of improvement is lower than for $df = 0.8$. This explains the counterintuitive reversed behaviour of average relative improvements and portions of improvement w.r.t. their dependency on due date factors.

6 Conclusion

Having introduced a deterministic model integrating activity exchange within a project scheduling model before, here a stochastic version of that deterministic model is presented. The transition from a deterministic model to a stochastic leads to considerable changes: So solutions are not given as simple schedules but as scheduling policies which further increases the complexity of the problem. Since optimizing over all possible scheduling policies is usually considered intractable, we apply here

a heuristic approach for which the improvement by application of the heuristic is studied for different objectives (makespan, tardiness) and parameter combinations. It turns out that the exchange policy actually leads to significant improvements especially for higher numbers of component positions and if these are distributed among fewer types.
Further work could also take into account resources, which were not considered here to study the effects of exchange better. Accordingly, one had to develop heuristics which integrate corresponding features, for which the work on the SRCPSP may be helpful. Beside heuristics, one could try to tackle the problem studied here or even a resource extension exactly, which however seems rather challenging given the experience made especially with the SRCPSP. Furthermore one could also consider additional inventories such that a limited number of defective components may be also replaced by intact components; theoretically such a case can be simulated already in the model here, but practically such instances would look differently than those generated in our study. In that sense it may be interesting to study the interplay between exchanges, resources and inventories by suitable simulation models, for which, however, also suitable policies would be needed which integrate all these factors in an appopriate way. The insights from that one studied here for exchange may be helpful on this way.

7 References

Abdolshah, M. (2014): A Review of Resource-Constrained Project Scheduling Problems (RCPSP) Approaches and Solutions. International Transaction Journal of Engineering, Management & Applied Sciences & Technologies 5, 253-286

Adlakha, V.G., and V.G. Kulkarni (1989): A classified Bibliography of Research on Stochastic PERT Networks. INFOR: Information Systems and Operational Research 27, 272-296

Arora, S., and B. Barak (2009): Computational complexity: A modern approach. Cambridge University Press, Cambridge

Artigues, C., S. Demassey, and E. Néron (2008): Resource-Constrained Project Scheduling: Models, Algorithms, Extensions and Applications. Wiley, Hoboken

Ashtiani, B., R. Leus, and M.-B. Aryanezhad (2011): New competitive results for the stochastic resource-constrained project scheduling problem: exploring the benefits of pre-processing. Journal of Scheduling 14, 157-171

Aust, J., and D. Pons (2019): Taxonomy of Gas Turbine Blade Defects. Aerospace 6(5), Art. 58

Ballestín, F. (2007): When it is worthwile to work with the stochastic RCPSP? Journal of Scheduling 10, 153-166

Ballestín, F., and R. Leus (2009): Resource-Constrained Project Scheduling for Timely Project Completion with Stochastic Activity Durations. Production and Operations Management 18, 459-474

Banghart, M. (2017) Identification of Reverse Engineering Candidates utilizing Machine Learning and Aircraft Cannibalization Data. International Journal of Aviation, Aeronautics, and Aerospace 4(4), Art. 5

Berthold, L. (2021): Project Scheduling with Activity Exchange motivated by Maintenance Operations. Unpublished paper, Institute of Operations Management, University of Hamburg (part of this thesis)

Brucker, P., A. Drexl, R. Möhring, K. Neumann, and E. Pesch (1999): Resource-constrained project scheduling: Notation, classification, models, and methods. European Journal of Operational Research 112, 3-41

Chen, D., X. Wang, J. Zhao (2012) Aircraft Maintenance Decision System Based on Real-time Condition Monitoring. Procedia Engineering 29, 765-769

Chen, Z., E. Demeulemeester, S. Bai, and Y. Guo (2018): Efficient priority rules for the stochastic resource-constrained project scheduling problem. European Journal of Operational Research 270, 957-967

Creemers, S. (2015): Minimizing the expected makespan of a project with stochastic activity durations under resource constraints. Journal of Scheduling 18, 263-273

Davari, M., and E. Demeulemeester (2019): The proactive and reactive resource-constrained project scheduling problem. Journal of Scheduling 22, 211-237

Deblaere, F., E. Demeulemeester, and W. Herroelen (2011): Proactive policies for the stochastic resource-constrained project scheduling problem. European Journal of Operational Research 214, 308-316

Habibi, F., F. Barzinpour, and S.J. Sadjadi (2018): Resource-constrained project scheduling problem: review of past and recent developments. Journal of Project Management 3, 55-88

Hagstrom, J. N. (1988): Computational complexity of PERT problems. Networks 18, 139–147

Hajdu, M., and O. Bokor (2014): The Effects of Different Activity Distributions on Project Duration in PERT Networks. Procedia Social and Behavioral Sciences 119, 766-775

Hartmann, S., and D. Briskorn (2010): A survey of variants and extensions of the resource-constrained project scheduling problem. European Journal of Operational Research 207, 1-14

Herroelen, W., and R. Leus (2005): Project scheduling under uncertainty: Survey and research potentials. European Journal of Operational Research 165, 289-306

Igelmund, G., and F.J. Radermacher (1983): Preselective Strategies for the Optimization of Stochastic Project Networks Under Resource Constraints. Networks 13, 1-28

Kelley, J.E., and M.R. Walker (1959): Critical-Path Planning and Scheduling. Proceedings of the Eastern Joint Computer Conference 1959

Lamas, P., and E. Demeulemeester (2016): A purely proactive scheduling procedure for the resource-constrained project scheduling problem with stochastic activity durations. Journal of Scheduling 19, 409-428

Lambrechts, O., E. Demeulemeester, and W. Herroelen (2011): Time slack-based techniques for robust project scheduling subject to resource uncertainty. Annals of Operations Research 186, 443-464

Li, H., and N.K. Womer (2015): Solving stochastic resource-constrained project scheduling problems by closed-loop approximate dynamic programming. European Journal of Operational Research 246, 20-33

Ludwig, A., R.H. Möhring, and F. Stork (2001): A Computational Study on Bounding the Makespan Distribution in Stochastic Project Networks. Annals of Operations Research 102, 49-64

Malcolm, D.G., J.H. Roseboom, C.E. Clark, and W. Fazar (1959): Application of a Technique for Research and Development Program Evaluation. Operations Research 7, 646-669

Möhring, R.H. (2001): Scheduling under uncertainty: Bounding the Makespan Distribution. In: H. Alt (ed.): Computational Discrete Mathematics. Springer, Berlin

Möhring, R.H., and F.J. Radermacher (1984): Introduction to Stochastic Scheduling Problems. Proceedings of the Conference on Operations Research, Oberwolfach 26.2.-3.3.1984

Nickles, G., H. Him, S. Koenig, A. Gramopadhye, and B. Melloy (1999): A Descriptive Model of Aircraft Inspection Activities. Federal Aviation Administration

Pich, M.T., C.H. Loch, and A. De Meyer (2002): On Uncertainty, Ambiguity, and Complexity in Project Management. Management Science 48, 1008-1023

Qing, X., W. Li, Y. Wang, and H. Sun (2019) Piezoelectric Transducer-Based Structural Health Monitoring for Aircraft Applications. Sensors 19(3), Art. 545

Regattieri, A., M. Gamberi, R. Gamberini, and R. Manzini (2005): Managing lumpy demand for aicraft spare parts. Journal of Air Transport Management 11, 426-431

Regattieri, A., A. Giazzi, M. Gamberi, and R. Gamberini (2015): An innovative method to optimize the maintenance policies in an aircraft: General framework and case study. Journal of Air Transport Management 44, 8-20

Rodrigues, L., J. Gomes, F. Ferri, I. Medeiros, R. Galvao, and C. Nascimento (2015): Use of PHM Information and System Architecture for Optimized Aircraft Maintenance Planning. IEEE Systems Journal 9(4), 1197-1207

Rostami, S., S. Creemers, and R. Leus (2018): New strategies for stochastic resource-constrained project scheduling. Journal of Scheduling 21, 349-365

Salman, S., C.R. Cassady, E.A. Pohl, and S.W. Ormon (2007): Evaluating the Impact of Cannibalization on Fleet Performance. Quality and Reliability Engineering International 23, 445-457

Schwindt, C., and J. Zimmermann (eds.) (2015): Handbook on project management and scheduling (2 vol.). Springer, Berlin

Stork, F. (2001): Stochastic Resource-Constrained Project Scheduling. Ph.D. Thesis, Technische Universität Berlin, Germany

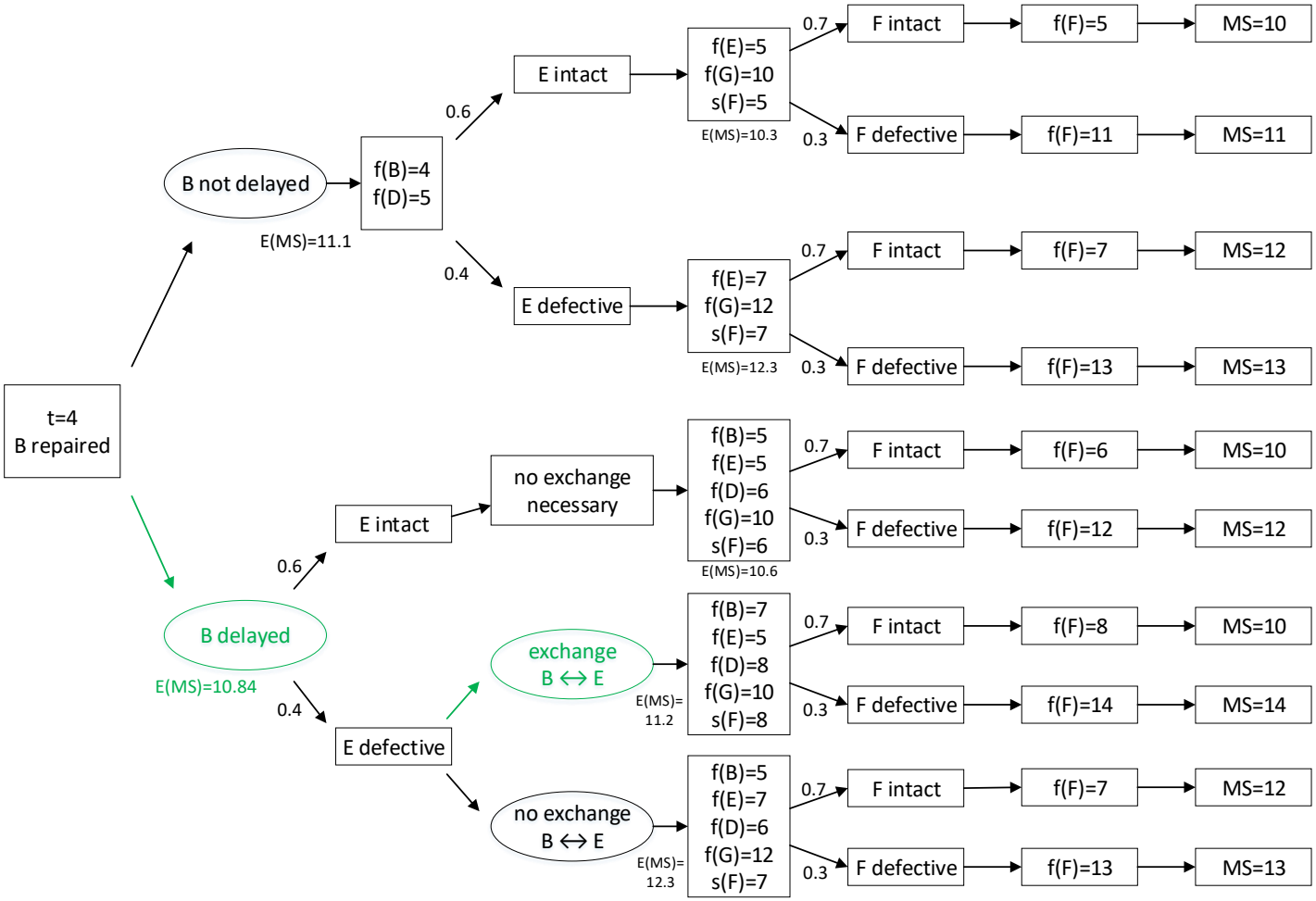
Tsai, Y.-W., and D.D. Gemmill (1998): Using tabu search to schedule activities of stochastic resource-constrained projects. Eurpean Journal of Operational Research 111, 129-141

Van de Vonder, S., F. Ballestin, E. Demeulemeester, and W. Herroelen (2007): Heuristic procedures for reactive project scheduling. Computers & Industrial Engineering 52, 11-28

Wang, Y., Z. He, L.-P. Kerkhove, and M. Vanhoucke (2017): On the performance of priority rules for the stochastic resource constrained multi-project scheduling problem. Computers & Industrial Engineering 114, 223–234

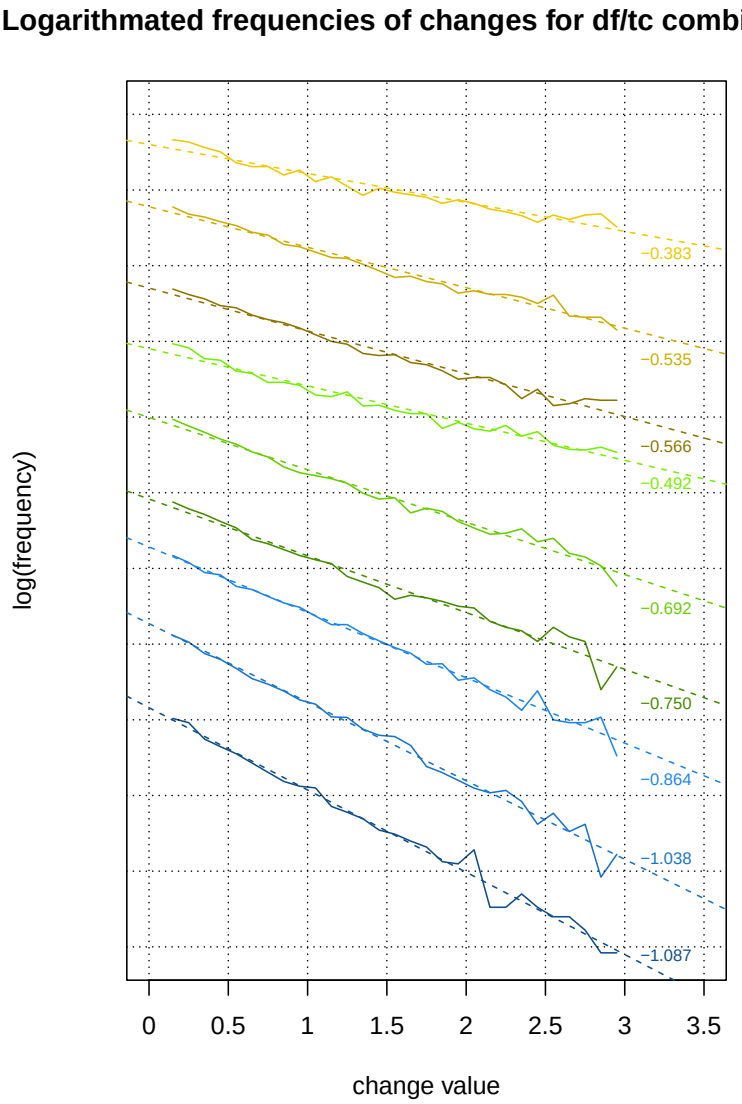
8 Appendix

8.1 Stochastic Dynamic Program



Optimal decisions are highlighted in green.

8.2 Frequencies of changes for df/tp combinations



Colours as earlier for non-logarithmated frequencies, i.e. blue for $df = 0.8$, green for 1.0, yellow-brown for 1.2; darker for $t/p = 2/0.2$, medium for 4/0.3, lighter for 9/0.5. Indicated values are the slopes of corresponding linear regression lines. For better readability the graphs are successively shifted by 1 in the y-axis, which also corresponds to one unit of that axis.
There are some differences between *tp*-combinations for given *df*-value, but which are in most cases smaller than those between *df*-values. Especially 2/0.2 and 4/0.3 have nearly the same slope coefficients for each *df*-value whereas the slope values for 9/0.5 are a bit higher (slower decrease). The higher similarity between the first two t/p-combinations is not that surprising given that type combinations 2 and 3 are and behave more similar to each other than to type combination 9 (and additionally probabilities 0.2 and 0.3 are nearer to each other than 0.5).

Auflistung der eigenen Arbeitsanteile

Berthold, L., and M. Fliedner: Optimal spare part utilization in maintenance projects and its complexity

Durchführung: 75%

Schriftliche Abfassung: 100%

Berthold, L.: Project Scheduling with Activity Exchange motivated by Maintenance Operations

Durchführung und schriftliche Abfassung: 100%

Berthold, L.: Stochastic Activity Exchange Project Scheduling

Durchführung und schriftliche Abfassung: 100%

Eidesstattliche Versicherung

Hiermit erkläre ich, Lukas David Berthold, an Eides statt, dass ich die Dissertation mit dem Titel "Project Scheduling with Activity Exchange" selbstständig und bei einer Zusammenarbeit mit anderen Wissenschaftlern gemäß der beigefügten Darlegung nach § 6 Abs. 4 der Promotionsordnung der Fakultät für Betriebswirtschaft vom 9. Juli 2014 verfasst und keine anderen als die von mir angegebenen Hilfmittel benutzt habe. Die den herangezogenen Werken wörtlich oder sinngemäß entnommenen Stellen sind als solche gekennzeichnet.
Ich versichere, dass ich keine kommerzielle Promotionsberatung in Anspruch genommen habe und die Arbeit nicht schon in einem früheren Promotionsverfahren im In- oder Ausland angenommen oder als ungenügend beurteilt worden ist.

Ort, Datum Unterschrift