R&D-Project Scheduling when Activities May Fail

Bert De Reyck[1] • Roel Leus[2]

[1]*London Business School, Regent's Park, London NW1 4SA, United Kingdom*

[2]*Department of Decision Sciences and Information Management, Faculty of Economics and Management, KULeuven, Leuven, Belgium*

bdereyck@london.edu • *Roel.Leus@econ.kuleuven.be*

An R&D project typically consists of several stages. Due to technological risks, the project may have to be terminated before completion, each stage having a specific likelihood of success. In the project-planning and -scheduling literature, this technological uncertainty has typically been ignored and project plans are developed only for scenarios in which the project succeeds. In this paper we examine how to schedule projects in order to maximize their expected net present value when the project activities have a probability of failure and when an activity's failure leads to overall project termination. We formulate the problem, show that it is NP-hard, develop a branch-and-bound algorithm that allows to obtain optimal solutions and provide extensive computational results. In the process, we establish a complexity result for an open problem in single-machine scheduling, namely for the discounted weighted-completion-time objective with general precedence constraints.

Keywords: project management; scheduling; risk; research and development; analysis of algorithms; computational complexity

Area of Review: Scheduling and Logistics

---

1. Introduction

An important feature of Research-and-Development (R&D) projects is that, apart from the commercial and market risks common to all projects, their constituent activities also carry the risk of technical failure. Therefore, besides projects overrunning their budgets or deadlines and the commercial returns not meeting their targets, R&D projects also carry the risk of failing altogether, resulting in time and resources spent without any tangible return. In this paper, we tackle the problem of scheduling the activities of an R&D project that is subject to technological uncertainty, i.e. in which the individual activities carry a risk of failure, and where an activity's failure results in the project's overall failure. The goal is to schedule the activities in such a way as to maximize the expected net present value of the project, taking into account the activity costs, the cash flows generated by a successful project, the activity durations and the probability of failure of each of the activities.

The algorithms developed in this paper are useful for any R&D setting where activities carry a risk of failure, and are of particular interest to drug-development projects in the pharmaceutical industry, in which stringent scientific procedures have to be followed to ensure patient safety in distinct stages before a medicine can be approved for production. The project may need to be terminated in any of these stages, either because the product is revealed not to have the desired properties or because of harmful side effects. The failure of one of the stages results in overall project termination. As stated by Gassmann et al. (2004), "If a drug candidate fails during the development phase it is withdrawn entirely from further testing. Unlike in the automobile industry, drugs are not modular products where a faulty stick shift can be replaced without throwing the entire car design away. In pharmaceutical R&D, drug design cannot be changed."

The contributions of this paper are the following. We introduce and formulate a generic model for optimally scheduling R&D-project activities with non-zero failure probability subject to precedence constraints, referred to as the *R&D-Project Scheduling Problem* (RDPSP). We show that the RDPSP is NP-hard and develop a branch-and-bound algorithm that is capable of solving the RDPSP to optimality. We present computational tests demonstrating the capabilities of the algorithm and we discuss how the model and algorithms can be extended to take into account the risk preferences of the decision maker. The complexity status of the single-machine scheduling problem with discounted weighted-completion-time objective is established as an intermediate result.

In our model we make a number of simplifying assumptions, including unlimited resources and no explicit consideration of the uncertainty in activity durations or project cash flows. These restrictions allow us to focus on the effect of possible technological failure on the development of optimal R&D-project schedules. We will show how to identify a project schedule that maximizes the project's expected net present value (expected NPV, eNPV), whereas a more simplified approach can result in a lower – and possibly negative – eNPV. In other words, we may find projects to be worthwhile to pursue while they would be rejected using more simplistic scheduling. These benefits of advanced scheduling procedures are significant especially for medium- to high-risk projects. Other insights include the fact that CPM-based schedules are good when the probability of failure is small and when the decision-maker is risk-seeking; longer schedules (with less parallel activities) tend to be better when the probabilities of failure are significant and when the decision-makers are risk-averse.

The remainder of this article is organized as follows. Related work is discussed in Section 2. Section 3 presents an introductory problem description by means of a real-life example from the pharmaceutical sector. A detailed problem formulation of the *R&D-*

Project Scheduling Problem (RDPSP) and an examination of its properties are given in Section 4. In Section 5, we provide an overview of a branch-and-bound algorithm for solving the RDPSP to optimality. We explain our upper-bounding procedure in Section 6 and provide details on branching and fathoming in Section 7. Section 8 investigates how risk preferences of the decision maker can be incorporated. In Section 9, we discuss computational tests that demonstrate the capabilities of our procedure. Section 10 presents a number of insights based on further numerical experiments. Finally, a summary and outlook on further research are given in Section 11.

2. Related work

The issue of parallel versus sequential scheduling of project activities, which lies at the core of the problem discussed in this paper, has been addressed, among others, by Dahan (1998), Eppinger et al. (1994) and Krishnan et al. (1997). This topic is also closely related to concurrent engineering, a systematic approach to the integrated, concurrent design of products and their related processes (Hill, 2002). Hoedemaker et al. (1999) provide some theoretical evidence as to why there are limits to the benefits of parallelization. Parallel (redundant) development of alternative technologies is studied in Abernathy and Rosenbloom (1969), Bard (1985) and Krishnan and Bhattacharya (2002), and a generic representation of multi-stage R&D problems is provided in Lockett and Gear (1973). Zemel et al. (2001) focus on the optimal timing of support activities for R&D tasks of variable length. Ding and Eliashberg (2002) examine the so-called 'pipeline problem': since New Product Development (NPD) projects may fail in each stage, multiple projects are started simultaneously in order to increase the likelihood of having at least one successful product. In our model, we lift the limiting assumption encountered in the aforementioned studies that R&D projects are limited to a single uncertain activity or sequential R&D stages only and allow the precedence relations between the individual activities to take the form of an arbitrary acyclic graph.

The literature on *deterministic* project scheduling is vast and contains numerous methods and algorithms for producing project schedules. For recent overviews of scheduling with NPV objective we refer to Herroelen et al. (1997) and Padman et al. (1997). The incorporation of *uncertainty* into project planning and scheduling has also resulted in numerous research efforts, particularly focusing on uncertainty in the activities' duration or cost; for a recent survey, see Herroelen and Leus (2005). None of these models, however, incorporates technological uncertainty in the form of stochastic-success activities.

Closely related to our model is that of Weitzmann (1979), who describes an optimal search procedure for obtaining maximum reward from a number of independent testing efforts; only sequential testing is considered. Granot and Zuckerman (1991) also examine sequencing for R&D projects with success or failure in individual activities but only consider sequential stages. Denardo et al. (2004) consider R&D projects that are successful if a successful path of edges from stem to leaf in a forest is found. Most similar to our setting are the works by Boros and Ünlüyurt (1999) and Ünlüyurt (2004) on sequential testing, and by Schmidt and Grossmann (1996) on scheduling NPD testing tasks, where also non-sequential testing is admitted; differences between these sources and this article are outlined in Section 4.1.

Schmidt and Grossmann (1996) point out that in many industries, including the chemical and pharmaceutical sectors, a number of the tasks involved in producing a new product are regulatory requirements such as environmental and safety tests. The failure of a single required test may prevent a potential product from reaching the marketplace. An informal overview of the importance of including the possibility of technical failure into planning is given in Blau et al. (2000), who focus especially on the pharmaceutical industry. DiMasi (2001) also refers to economic, efficacy, safety and 'other' reasons for cutting projects. In this paper, we will mainly refer to 'technical' success of products. More information on success probabilities in the pharma sector can be found in Zipfel (2003); a broader overview of key issues and strategies for optimization in pharmaceutical supply chains is provided by Shah (2004).

3. An example

In the US, the pharmaceutical drug-development process is monitored by the Food and Drug Administration (FDA), and typically follows four main stages: basic research, pre-clinical, clinical and FDA review, with the clinical stage subdivided in Phase I, II and III. Each clinical substage contains a number of tasks that are repeated several times, each time increasing in duration. Similar processes are followed in Europe and the rest of the world.

We present an example of a pharmaceutical project initiated by a biotech company based in Cambridge, England. The project was started in 2001 with an expected US market launch in 2008, assuming that the product makes it successfully through all the

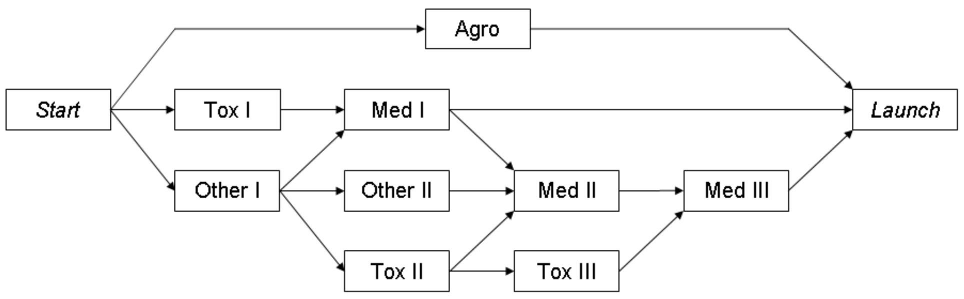


Figure 1: Precedence network for the example project.

stages. At the time of this writing, all activities prior to the clinical stage have been successfully performed, and the company is developing a project plan for the clinical development and launch of the product. The total remaining duration of the project is approximately five years, for a total cost of approximately £15 million (all data are disguised). For the purpose of this paper, we have simplified the project plan, which contains more than 300 activities, by identifying natural task groupings, yielding the aggregate project network structure in Figure 1. More details can be found in Crama et al. (2006). Phase III in this project is subdivided in three runs of toxicological studies on animals, referred to as 'Tox', and medical studies on humans, referred to as 'Med'. The remaining activities in Phase III have been grouped into two tasks named 'Other', which include manufacturing of the product, chemical product analysis and pharmacological studies. The project also includes the ancillary agronomical task ('Agro'). Each medical study has to be preceded by its corresponding toxicology study. The toxicology studies, however, do not require the results from the previous medical study. Some toxicology and medical studies are dependent on the 'Other' activities in the network. The agronomical activity can be scheduled freely.

Table 1 gives for each activity group the total development cost, the duration and the probability of technical success (PTS). The project has an estimated overall PTS of 16.2%. If successful, the NPV of net sales equals £300 million. For this example, we use a discount rate of 1% per month.

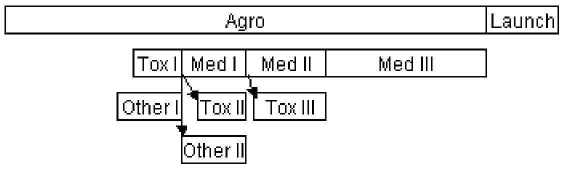
While developing a schedule for this project, several considerations are in order. If all activities are carried out as soon as possible, the revenues of the project, if successful, are received as soon as possible, resulting in a high present value. On the other hand, development costs are also incurred early on. A better option is to execute the project according to the late-start schedule as determined by the Critical Path Method

task	cash flow (£)	duration (months)	PTS
Agro	−12,000,000	60	100%
Tox I	−300,000	6	75%
Other I	−1,000,000	8	100%
Med I	−200,000	8	80%
Other II	−300,000	8	100%
Tox II	−100,000	6	75%
Med II	−200,000	10	80%
Tox III	−700,000	9	75%
Med III	−400,000	20	60%
Launch	300,000,000	-	-

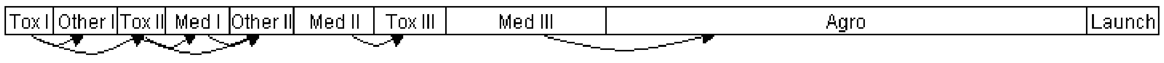
Table 1: Project data (disguised).

(CPM). This corresponds with the first schedule of Figure 2 and results in an eNPV of approximately £13 million.

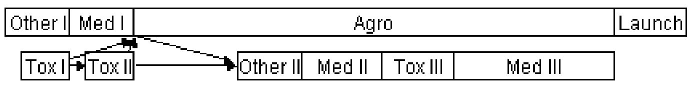
Alternatively, we can schedule the activities carrying technical risk in series, thereby avoiding unnecessary expenditures when one of the activities fails. One such schedule is depicted in Figure 2(b), with an eNPV of approximately £10 million. Note that the arrows in Figure 2 do not represent the technological precedence relations but extra 'information flows': knowledge of the outcome of an uncertain activity constitutes useful information since a failure allows to abandon the project without investing in the remaining tasks. Information flows implied by the precedence relations are not shown.



(a) CPM late-start schedule



(b) Serial schedule



(c) Optimal schedule

Figure 2: Three possible schedules for the R&D project.

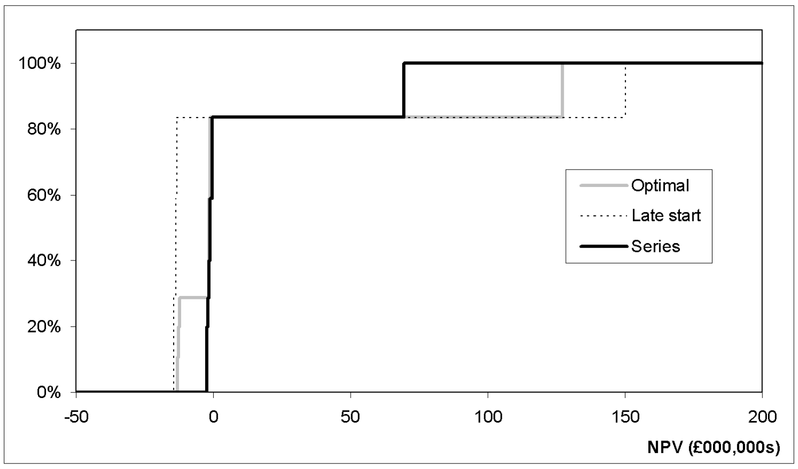


Figure 3: Cumulative distribution function of NPV for the three schedules.

Finally, a schedule allowing for a partial overlap of R&D activities is shown in Figure 2(c), yielding an eNPV of approximately £16 million, which can be shown to be the highest value achievable. This schedule exhibits the optimal trade-off between overlapping activities 'at risk' and the cost of delaying project completion and market launch. Finding such a schedule is the objective of the algorithms that will be presented in this paper.

The probability distributions of the project's NPV for each of the three schedules are depicted in Figure 3. Clearly, the different schedules exhibit very different risk profiles. The series schedule is conservative and minimizes the downside risk, but the total project execution time is maximal. On the other extreme, a CPM schedule results in a large downside risk, compensated for by an earlier launch date, yielding a higher upside potential. In between these two extremes we find the optimal schedule, which strikes a balance between timeliness of project launch and limiting at-risk investments and the associated costs.

4. Problem formulation and properties

4.1 Problem formulation and notation

The objective of the RDPSP is to maximize the eNPV of the project by constructing a project schedule specifying when to execute each activity. The final project payoff is only achieved when all activities are successful, and the project is terminated as soon as an activity fails. We focus on the case where all activity cash flows during the development phase are negative, which is typical for R&D projects. Activity success or failure is

N	$= \{0, 1, \ldots, n\}$, the set of project activities; $N_i = N \backslash \{i\}$ $(i \in N)$ and $N_{0n} = N \backslash \{0, n\}$
c_i	cash flow of activity $i \in N_n$, non-positive integer; incurred at the start of the activity
C	integer end-of-project payoff, ≥ 0; received at the start of activity n
d_i	duration of activity $i \in N_n$, non-negative integer (positive for $i \in N_{0n}$)
p_i	probability of technical success (PTS) of activity $i \in N_n$
r	continuous discount rate
A	(strict) partial order on N, i.e. an irreflexive and transitive relation, representing technological precedence constraints
s_i	starting time of activity $i \in N$, ≥ 0; starting-time vector $\mathbf{s}$ is a schedule
δ	project deadline

Table 2: Definitions.

revealed at the end of each activity. Consequently, each activity will only be started if all the activities scheduled to finish earlier have a positive outcome. Therefore, in the objective function, the activity cash flows are weighted by the probability of joint success of all its scheduled predecessors. We do not consider resource constraints and duration uncertainty, and consider the PTS of the different tasks as independent. The parameters that are used throughout the paper are defined in Table 2.

Without loss of generality, we assume activity 0 to be a dummy representing project initiation, with $c_0 = d_0 = 0$ and $p_0 = 1$, and $(0, i) \in A$ for all $i \in N_0$. Activity n represents project completion and is a successor of all other activities. Activities N_{0n} are referred to as *intermediate* activities; we assume that $d_i > 0$ for $i \in N_{0n}$. A deadline δ is imposed on project completion: we require that $s_n \leq \delta$. This deadline is needed because optimization will try to push activity start times to infinity if the optimal eNPV of a particular problem instance is negative. A second reason for using a deadline is that it allows to examine the impact of schedule length on the quality of the schedule.

Relation A imposes the following constraints on $\mathbf{s}$:

$$s_i + d_i \leq s_j \qquad \forall (i, j) \in A$$

For an arbitrary relation E on N, define $\mathcal{S}(E) = \{\mathbf{s} \in \mathbb{R}_{\geq}^{n+1} : s_i + d_i \leq s_j, \forall (i, j) \in E\}$, which is a convex polyhedron ($\mathbb{R}_{\geq}$ denotes the set of positive real numbers). $\mathcal{S}(E)$ is non-empty if and only if the corresponding precedence graph $G(N, E)$ is acyclic. The set of feasible schedules for RDPSP is $\{\mathbf{s} \in \mathcal{S}(A) : s_n \leq \delta\}$. Clearly, if $A \subseteq E$ then $\mathcal{S}(E) \subseteq \mathcal{S}(A)$. If $A \subseteq E$ and $G(N, E)$ is acyclic, we say that E is a *feasible extension* of A. For a given schedule $\mathbf{s}$, we define the schedule-induced strict order $R(\mathbf{s}) = \{(i, j) \in N \times N | i \neq j \wedge s_i + d_i \leq s_j\}$, which corresponds to the precedence constraints implied by $\mathbf{s}$ (see e.g. Bartusch et al., 1988; Neumann et al., 2003).

This paper investigates the determination of an optimal schedule for RDPSP. For $i \in N_0$, we define

$$q_i(E) = \prod_{(k,i)\in E} p_k, \text{ with } E \text{ any order relation on } N$$

As explained in Section 3, for schedule $\mathbf{s} \in \mathcal{S}(A)$, the activity pairs in $R(\mathbf{s})$ can be considered as representing 'information flows': the probability that activity i is initiated and hence induces expenditures is equal to the probability that all activities scheduled to finish no later than s_i succeed, which equals $q_i(R(\mathbf{s}))$. Remark that $q_n(R(\mathbf{s}))$ is a constant, independent of the schedule; we write $q_n \equiv q_n(R(\mathbf{s}))$. RDPSP can now be formulated as follows:

$$\max \qquad g(\mathbf{s}, E) = q_n C e^{-rs_n} + \sum_{i=1}^{n-1} q_i(E) c_i e^{-rs_i}$$

$$\text{subject to} \quad \begin{cases} \mathbf{s} \in \mathcal{S}(A) \\ E = R(\mathbf{s}) \\ s_n \leq \delta \end{cases}$$

In the objective function $g()$, each activity cash flow c_i is weighted with two factors, namely with $q_i(E)$, the probability of joint success of all predecessors in time, and with a discount factor e^{-rs_i}, dependent on the starting time s_i of activity i.

Schmidt and Grossmann (1996) propose a generalized version of the foregoing model, in which multiple scenarios are allowed for the activity data (c_i, d_i, p_i); project payoff is a piecewise-linear decreasing function of time but is not discounted. They do not, however, obtain exact results: they approximate the non-linear objective function with a piecewise-linear function and, for larger instances, impose further simplifications such as a project deadline equal to the longest path length in $G(N, A)$, $r = 0$, and a linear approximation of the objective.

A significant body of literature exists on the problem of diagnosing a complex system by means of a sequence of tests of its components, we refer to Boros and Ünlüyurt (1999) and Ünlüyurt (2004) for reviews. Their setting is rather similar to ours, apart from the fact that (1) $R(\mathbf{s})$ needs to be a complete order on N (a full sequence), and (2) no discounting is considered ($r = 0$). It will become evident from Section 4.3 that these two properties go hand in hand.

4.2 Sketch of the solution approach

In the next paragraphs, we draw a sketch of the solution approach. A detailed description of our solution algorithm is provided in Section 5.

RDPSP is solved in two phases. In the first phase, we produce a feasible extension E of A, which yields values $\mathbf{q}(E)$. We then optimize $g(\mathbf{s}, E)$ in $\mathbf{s}$ subject to $\mathbf{s} \in \mathcal{S}(E)$ and the deadline constraint, which constitutes the second phase. If we implicitly or explicitly enumerate all feasible extensions of A, we are guaranteed to find an optimal schedule for RDPSP, since for each feasible schedule $\mathbf{s} \in \mathcal{S}(A)$ it holds that $\mathbf{s} \in \mathcal{S}(R(\mathbf{s}))$, and $R(\mathbf{s})$ extends A; a corresponding relation E is called an *optimal feasible extension.*

The second phase (optimization for *given* coefficients $\mathbf{q}$) amounts to project scheduling with NPV objective without resource constraints (see Herroelen et al., 1997). In this case, the scheduling problem is easily solved because all intermediate cash flows are non-positive: each activity can be scheduled to end at the earliest of the starting times of its successors in E. Depending on whether the corresponding eNPV is positive or negative, we set $s_0 = 0$ or $s_n = \delta$, respectively. The resulting schedule is referred to as $\phi(E)$. Note that Schmidt and Grossmann (1996) opt for an early-start schedule rather than this late-start approach.

4.3 Properties

The following theorem allows us to establish ties with the literature on sequential testing.

Theorem 1. *If $r = 0$ and $\delta \geq \sum_{i \in N_n} d_i$ then an optimal feasible extension of A exists that is a complete order on N.*

The proofs of the theorems appear in the appendix. Intuitively, the theorem says that when money has no time value, it is a dominant choice to perform all tasks sequentially.

We define problem LCT as problem RDPSP whose solution space is restricted to schedules that impose a complete order on N; Monma and Sidney (1979) refer to this setting as the 'least-cost fault-detection problem'. Remark that LCT is not a sub-problem of RDPSP since we restrict the set of solutions and not the input parameters.

Without dummy start and end (and so without final project payoff), a number of special cases of LCT with $r = 0$ can be solved in polynomial time. If $A = \varnothing$ then each optimal complete order relation E sequences the activities in non-increasing order of $c_i/(1 - p_i)$, and each complete order that sequences the activities in non-increasing order of $c_i/(1 - p_i)$ is optimal. One of the earliest references for this result seems to be Mitten (1960), obtained in the context of 'least-cost testing'; another source is Butterworth (1972). A polynomial-time algorithm for LCT also exists when $G(N, A)$ consists of a number of *parallel chains* (see Chiu et al., 1999). Based on Monma and Sidney (1979) it can be shown that the problem is also solvable in polynomial time when $G(N, A)$ is *series-parallel*.

The foregoing results carry over to RDPSP when $\delta \geq \sum_{i \in N_n} d_i$ and $r = 0$. However, the incorporation of precedence constraints taking the form of an arbitrary acyclic digraph $G(N, A)$ results in an NP-hard problem:

Theorem 2. RDPSP *is* NP-*hard in the ordinary sense, even if* $r = 0$, $C = 0$, $\forall i \in N_{0n} : d_i = 1$, *and* $\delta \geq \sum_{i \in N_n} d_i$.

Corollary 1. LCT *is ordinarily* NP-*hard under the same conditions.*

This corollary settles what is said to be an open problem in Monma and Sidney (1979) and in Ünlüyurt (2004). In order to further examine the complexity status of LCT, we start with problem $1|prec|\Sigma w_j(1 - e^{-rC_j})$, the single-machine scheduling problem with discounted weighted-completion-time objective and general precedence constraints, with objective function to be *minimized* (see for instance Pinedo, 2002). The complexity status of this scheduling problem was considered to be open in Monma and Sidney (1979) (with max-objective, but this does not change the result), and has to the best of our knowledge since then not been treated in the scheduling literature.

Lemma 1. *Problem* $1|prec|\Sigma w_j(1 - e^{-rC_j})$ *is strongly* NP-*hard, even with unit durations.*

Based on this lemma we derive the following theorem.

Theorem 3. LCT *is* NP-*hard in the strong sense even if* $C = 0$ *and* $\forall i \in N_{0n} : p_i = d_i = 1$.

In the remaining sections of this text we deal only with problem RDPSP, not with LCT.

5. A branch-and-bound algorithm

In light of the NP-hardness of the RDPSP, an exact algorithm with better than exponential time complexity is unlikely to exist, and we will devise a branch-and-bound (B&B) algorithm to implicitly enumerate the solution space. The algorithm follows the intuitive approach described in Section 4.2, although the distinction between the two phases is less explicit.

We use the concept of a 'distance matrix' as described by Bartusch et al. (1988) to collect information about *minimal* differences between the starting times of all pairs of activities. Lower bounds l_{ij} are imposed on the differences between the starting times of activities:

$$l_{ij} \leq s_j - s_i \qquad\qquad \forall i, j \in N$$

At the root of the search tree, we initialize

$$l_{n0} = -\delta, \qquad \text{and for } (i,j) \neq (n,0): \quad l_{ij} = \begin{cases} 0 & \text{if } i = j \\ d_i & \text{if } (i,j) \in A \\ -\infty & \text{otherwise} \end{cases}$$

The $(n+1) \times (n+1)$-matrix D tightens the foregoing individual minimal distances: distance-matrix entry D_{ij} is the length of a longest path from i to j in the complete graph with node set N and distances l_{ij}. For a set of values l_{ij}, the distance matrix can be found in $O(n^3)$ time, for instance by means of the Floyd-Warshall algorithm (Lawler, 1976).

It can be shown (Bartusch et al., 1988) that a feasible schedule exists iff all $D_{ii} = 0$. If $D_{ii} > 0$ for some i, the corresponding graph contains a directed cycle with positive length. We observe that, when $D_{ij} \geq d_i$ for an arbitrary activity pair (i,j), then activity j will always start after activity i has finished, and so $D_{ij} \geq d_i$ implies the possibility of information flow from i to j, denoted as "$i \rightarrow j$". The conditions $s_i + d_i > s_j$ and $s_j + d_j > s_i$ jointly imply that i will be executed in parallel with j ("$i||j$"). Since we work with discrete durations and hence discrete starting times, these conditions are equivalent with $D_{ij} \geq -d_j + 1$ and $D_{ji} \geq -d_i + 1$.

In node h of the search tree, minimal distances are $l^{(h)}$ and the distance matrix is $D^{(h)}$. For each node h we distinguish set π_h, the set of (unordered) activity pairs $\{i,j\}$ for which $i||j$ holds according to $D^{(h)}$ (the activity pairs that need to overlap). Implied information flows $i \rightarrow j$ are gathered in order relation E_h. Finally, we also maintain set ν_h, the set of activity pairs that are not in π_h nor in E_h. Branching continues while

$\nu_h \neq \varnothing$; a branching decision consists of the selection of a set $\{i, j\} \in \nu_h$ and generates three branches: (1) $i \rightarrow j$; (2) $j \rightarrow i$; and (3) $i||j$. These branching options are mutually exclusive and jointly exhaustive.

Exploring a branch means that we update the distance matrix to incorporate the additional constraints that are imposed via $l^{(h)}$. Each update can be performed in $O(n^2)$ time (cfr. Bartusch et al., 1988). The recognition of additional implied parallel and serial relations (in π_h and E_h, respectively) is embedded in the distance updates and does not add to the $O(n^2)$ time complexity of these updates. Search nodes that no longer allow a feasible solution are immediately recognized when the distance updates trigger a change in $D^{(h)}_{ii}$ for some $i \in N$.

6. Upper bounds

Define $g^{(h)}$ to be the best objective value reachable from node h of the search tree. In other words,

$$g^{(h)} = \max_{\mathbf{s},E} \left\{ q_n C e^{-rs_n} + \sum_{i=1}^{n-1} q_i(E) c_i e^{-rs_i} \right\} \tag{1}$$

$$\text{subject to} \begin{cases} \mathbf{s} \in \mathcal{S}(E) \\ E \text{ is a feasible extension of } E_h \\ s_n \leq \delta \\ \mathbf{s} \text{ satisfies } D^{(h)} \end{cases} \tag{2}$$

In (2), '$\mathbf{s}$ satisfies $D^{(h)}$' is shorthand for '$\mathbf{s}$ satisfies the lower bounds on starting-time differences represented by $D^{(h)}$'. In the computation of upper bounds on $g^{(h)}$, we separate the determination of the values $\mathbf{q}$ and the discount factors.

In a first approach, we start by underestimating the execution probabilities $q_i(E)$, $i \in N_{0n}$: lower bounds for these values are $\theta^{(h)}_i = \prod_{j \in N : D^{(h)}_{ij} \leq -d_j} p_j$. We substitute these values into (1) and relax constraint set (2) to

$$\begin{cases} s_n \leq \delta \\ \mathbf{s} \text{ satisfies } D^{(h)} \end{cases} \tag{3}$$

Note that if $\mathbf{s}$ satisfies $D^{(h)}$, it automatically holds that $\mathbf{s} \in \mathcal{S}(E_h)$. The problem has been reduced to scheduling the activities with NPV objective subject to the constraints (3) on $\mathbf{s}$. If $D^{(h)}_{0n} \leq \delta$, the solution can be seen to satisfy

$$s_i = s_n - D^{(h)}_{in} \qquad \forall i \in N_n \tag{4}$$

for a given value of s_n. In an optimal schedule either $s_0 = 0$ or $s_n = \delta$, depending on the sign of the resulting NPV. The optimal objective function of this relaxation is referred to as $UB^{(h)}$. When $D_{0n}^{(h)} > \delta$, no feasible schedule exists corresponding with all branching decisions that were made to reach search node h; this situation will be recognized during the distance-matrix updates. In non-dominated leaf nodes h, $UB^{(h)}$ equals the exact objective-function value corresponding with extension $E^{(h)}$ of A (the dominance rule is discussed in the Section 7).

For the determination of $UB^{(h)}$, we replaced the values $\mathbf{q}$ first. Alternatively, discount factors e^{-rs_i} could be fixed first by substituting for s_i as given by Eq. (4), after which remains the determination of s_n and values $q_i(E)$. This leads to a new RDPSP instance with zero discount rate and cash flows $c_i e^{rD_{in}^{(h)}}$ for intermediate activities i. This new problem is subjected to the general precedence constraints contained in E_h. An efficient upper bound on its objective function can be computed by (e.g. greedily) extracting sets of chains from E_h and imposing only those constraints on the auxiliary problem. Unfortunately, the resulting bound on $g^{(h)}$ turns out to be rather weak and is not retained in the final version of our algorithm.

7. Algorithmic structure and details

Overall structure of the algorithm. A general overview of the structure of the B&B algorithm is given in Figure 4. Further details on some of its aspects are provided below.

Branching choice. We explore different rules for the selection of an activity pair $\{i, j\} \in \nu_h$ to branch on. As a first possibility, rule 1 selects the first encountered activity pair $\{i, j\}$ in ν_h based on lexicographic ordering of the alternatives. From our experiments we have observed that the 'low-impact' choices typically concern activities with a lot of slack in their starting times. Therefore, we have implemented rule 1 with activity ordering based on (1) the activity index and (2) float values (*increasing* CPM-based total float in $G(N, A)$). The goal of this second option is to make decisions that strongly affect the bounds on lower-indexed levels in the search tree. We also examine a *decreasing* order. Alternatively, we order the candidate activity pairs in decreasing order of a 'pseudo-cost' of insertion, which is an estimate of their true impact. The role of this pseudo-cost is in guiding heuristic decisions in the algorithm, not in generating incumbent solutions or

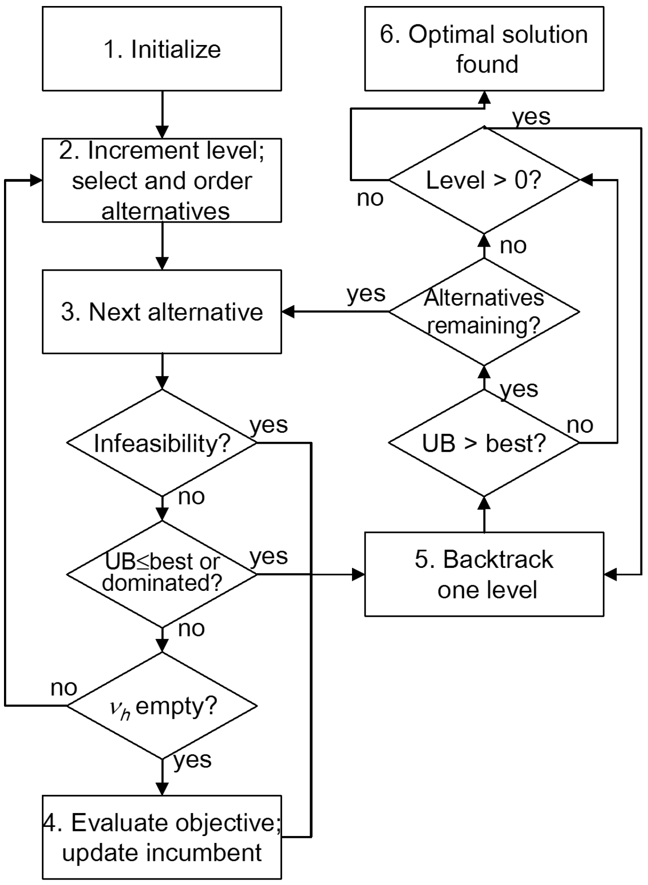


Figure 4: Flow chart of the algorithm.

in proving fathomability (Parker and Rardin, 1988). Rule 2 selects $\{i, j\} \in \nu_h$ with highest ratio $c_i/p_j + c_j/p_i$, in an attempt to make the most important decisions first. Rule 3 also tries to select the most influential activity pair $\{i, j\}$ first, by maximizing the difference between the latest ending time of the earliest starting activity (latest start times are given by $D^{(h)}_{0n} - D^{(h)}_{in}$) and the latest start of the other activity. Finally, rule 4 is a criterion that (approximately) minimizes the number of nodes in the search tree: we choose the activity pair that allows removing the most elements from ν_h, summed over its three emanating branches. An estimate of the number of elements removed by alternative $i \to j$ is $\#\{k \in N : ((j,k) \in E_h \wedge (i,k) \notin E_h) \vee ((k,i) \in E_h \wedge (k,j) \notin E_h)\}$; an estimate of the effect of $i||j$ is $\#\{k \in N : (\{j,k\} \in \pi_h \wedge \{i,k\} \notin \pi_h) \vee (\{k,i\} \in \pi_h \wedge \{k,j\} \notin \pi_h)\}$.

Branching order. We examine two different approaches with respect to the branching order, i.e. the order in which the three branches $i \to j$, $j \to i$ and $i||j$ are explored once a branching choice $\{i, j\}$ has been made. One possibility is to adhere to a *fixed* branching order; the actual order in this case turns out not to be decisive for algorithmic performance, we implement (1) $i \to j$, (2) $j \to i$ and (3) $i||j$. The second option is to use a variable order, in which we first select the branch that is compatible with the currently best known solution: if $s_i + d_i \leq s_j$ in this schedule, we first explore $i \to j$, then $i||j$ and finally $j \to i$. If i and j overlap in the incumbent, we first explore $i||j$; the second alternative is $i \to j$ if $s_i \leq s_j$.

Dominance rule. Consider the following lemma. A search node indexed h of the search tree is called a 'leaf node' if $\nu_h = \varnothing$.

Lemma 2. *A feasible solution in a leaf node h of the search tree can be discarded without loss of all optimal solutions if the following holds:*

$$\exists i \in N_{0n} : \forall (i,k) \in E_h : D_{ik}^{(h)} > d_i.$$

The proof of the lemma can be found in the appendix. The basic idea is that if parallelity constraints (constraints of the type $i||j$) are binding for a feasible solution, in the sense that at least one activity could be shifted later in time if such an (artificial) constraint were removed, then the solution is dominated. We underline that such parallelity constraints do remain useful for partitioning the search space. The lemma builds on the insight that distance-matrix entries can only increase, never decrease, when descending the search tree.

Based on Lemma 2 we have implemented a dominance rule. We dynamically maintain the cardinality of sets $S_i^1 = \{k \in N : \{i,k\} \in \nu_h\}$, $S_i^2 = \{k \in N : (i,k) \in E_h\}$ and $S_i^3 = \{k \in N : D_{ik}^{(h)} > d_i\}$ for each activity i, and we fathom a search node when $\exists i \in N_{0n} : |S_i^1| = 0$ and $|S_i^2| = |S_i^3|$.

A heuristic stand-alone procedure. We propose a heuristic that examines a set of order relations E, starting with $E = A$. We gradually append activity pairs to E until a full order is obtained; each solution $\phi(E)$ is evaluated and the best one retained. The procedure is described in pseudo-code as Algorithm 1. The output of this heuristic is used at the initialization phase of the B&B algorithm to produce a good lower bound LB. The procedure is interrupted when $s_n(\phi(E)) - s_0(\phi(E)) > \delta$, where $s_i(\phi(E))$ represents the $(1+i)$-th component of $\phi(E)$. Here and later, we write $g(\mathbf{s}, R(\mathbf{s}))$ as $g(\mathbf{s})$.

Algorithm 1 A heuristic procedure

```
$\mathbf{s}_{best} := \phi(A)$; $E := A$
construct full order $F$ extending $A$, sequencing incomparable activities in non-increasing order of $c_i/(1-p_i)$
for $d = (n-2)$ downto 1 do
  $S$ is the set of ordered activity pairs $(i,j)$ for which the difference between the rank order of $i$ and $j$ in $F$ equals $d$ and $j$ comes after $i$ in $F$
  order the elements $(i,j) \in S$ in decreasing $-c_j/p_i$
  for ordered $(i,j) \in S$ do
    $E := E \cup \{(i,j)\}$
    if $g(\phi(E)) > g(\mathbf{s}_{best})$ then
      $\mathbf{s}_{best} := \phi(E)$
    end if
  end for
end for
return $\mathbf{s}_{best}$
```

8. Incorporating risk preferences

The objective of RDPSP is to maximize the *expected* NPV, but this does not preclude actual project realizations from resulting in higher or lower NPV values. In order to evaluate the entire risk profile associated with a schedule, a representation of all possible NPV realizations together with the probability of each realization would be desirable (which was illustrated at the end of Section 3 for the example project). In the literature on project networks with stochastic activity durations, it is shown (Hagstrom, 1988; Möhring, 2001) that even with independent processing times, the determination of a single point of the cumulative distribution function of project completion time is #P-complete, and thus in particular NP-hard. As noted by Adlakha and Kulkarni (1989), the difficulty arises from two sources: (1) the number of paths grows exponentially in the number of activities, and (2) even when the activity durations are independent, the path lengths are generally dependent, as several paths have one or more activities in common.

Fortunately, our setting of stochastic-success activities does not suffer the same difficulties. In spite of the fact that $O(2^n)$ different realizations are possible of success or failure of the individual activities, the knowledge that activity failure leads to immediate project termination permits an efficient determination of the pmf (probability-mass function) of the NPV of an arbitrary schedule. With each schedule $\mathbf{s}$ we associate a set $\tau(\mathbf{s})$ of decision points corresponding with the (intermediate) activity start and finish times: $t \in \tau(\mathbf{s}) \Leftrightarrow \exists i \in N_{0n} : (t = s_i) \vee (t = s_i + d_i)$.

Algorithm 2 Computation of expectation and pmf of NPV for a schedule $\mathbf{s}$

```
$prob = 1; cost = 0$
$f_{\mathbf{s}}(\cdot) = 0; g(\mathbf{s}) = 0$
**for** increasing $t$ in $\tau(\mathbf{s})$ **do**
  **if** $\exists i \in N : t = s_i + d_i$ **then**
    $successpr = \prod_{i \in N | t = s_i + d_i} p_i$
    **if** $successpr < 1$ **then**
      $f_{\mathbf{s}}(cost) := f_{\mathbf{s}}(cost) + prob * (1 - successpr)$
      $g(\mathbf{s}) := g(\mathbf{s}) + cost * prob * (1 - successpr)$
      $prob := prob * succespr$
    **end if**
  **end if**
  **for all** $i \in N | s_i = t$ **do**
    $cost := cost + c_i e^{-rt}$
  **end for**
**end for**
$cost := cost + Ce^{-rs_n}$
$f_{\mathbf{s}}(cost) := f_{\mathbf{s}}(cost) + prob$
$g(\mathbf{s}) := g(\mathbf{s}) + cost * prob$
```

The procedure named Algorithm 2 determines the NPV-pmf of $\mathbf{s}$, denoted $f_{\mathbf{s}}(\cdot)$, and its expected NPV $g(\mathbf{s})$; it can be implemented in $O(n \log n)$ time. In the code, *prob* and *cost* respectively monitor the probability of reaching the different $t \in \tau(\mathbf{s})$ and the cost incurred up until that time. *successpr* represents the probability that all activities ending at the considered time instance succeed. For easy access $\tau(\mathbf{s})$ can be conceived as a multi-set (which is *not* explicitly taken into account in the code description). A bifurcation of probability mass occurs each time when fallible activities ($p_i < 1$) end.

The NPV-pmf can be used by the decision maker to evaluate the downside risk, e.g. the probability that the NPV is lower than a threshold value, or the upside potential, e.g. the probability that NPV is larger than or equal to a threshold. This gives the decision maker a number of additional options: (1) it allows for the specification of a constraint on downside risk and/or upside potential, which could be imposed during the search for schedules with maximum eNPV, and (2) the approach permits to generate the efficient frontier showing the trade-off between return and risk.

9. Computational performance

We have performed a series of computational experiments using randomly generated test problems in order to examine and enhance the performance of the B&B algorithm.

9.1 Experimental setup

Random test sets have been generated for various values of n using the random network generator RanGen (Demeulemeester et al., 2003). Each dataset contains 20 instances for each of the values 0.25, 0.50 and 0.75 of the network-shape parameter *order strength*[1] OS, resulting in 60 instances per set. Unless mentioned otherwise, we set $r = 0.05$. Cash flows for each activity in N_{0n} are generated as independent realizations of a discrete uniform random variable on $[-50; 0]$, durations for these activities are discrete values in $[1; 15]$, and success probabilities are, unless stated otherwise, chosen randomly from $[80\%; 100\%]$. Deadline δ is set at the non-restrictive value $\sum_{i \in N_n} d_i$.

The end-of-project payoff value C is an integer randomly selected from the interval $[0.5a; 2a]$ with

$$a = -(1/q_n) \sum_{i=1}^{n-1} c_i q_i^{(0)} \exp(0.05 D_{in}^{(0)}),$$

with distance matrix $D^{(0)}$ based on the initial order relation A and probabilities $q_i^{(0)}$ based on starting times $(D_{0n}^{(0)} - D_{in}^{(0)})$. Note that when $C \geq a$ the optimal project's eNPV is guaranteed to be non-negative. The algorithm can easily be adapted to exclude negative-eNPV schedules by exploring only search nodes with positive upper bounds. This would speed up the algorithm's running time for some of the test instances. We have not implemented this enhancement, since the value C is generated arbitrarily and its selection would allow for manipulation of the computational efficiency.

In order to compare the quality of schedules, we define the function

$$I(\mathbf{s}_1, \mathbf{s}_2) = (g(\mathbf{s}_2) - g(\mathbf{s}_1))/|g(\mathbf{s}_1)|,$$

which measures the improvement in the objective function $g()$ of a schedule $\mathbf{s}_2$ compared with a schedule $\mathbf{s}_1$. In the (rare) cases when $g(\mathbf{s}_1) = 0$, the instance is skipped when computing averages for a dataset.

The algorithms were coded in C using Microsoft Visual C++ 6.0. The experiments were run on a Dell OptiPlex GX620 PC with an Intel Pentium-4 2.80-GHz processor and 1 GB RAM, equipped with Windows XP Professional. Unless stated otherwise, a time limit of two minutes is imposed on the running time of the algorithms.

[1]The order strength is the number of comparable intermediate activity pairs divided by the maximum number $n(n-1)/2$ of such pairs, and is a measure for the closeness to a linear order of the technological precedence constraints in A (cfr. Mastor, 1970).

9.2 Parameter settings

For the dataset with $n = 20$, Table 3 shows the improvements in the performance of the B&B algorithm starting from the base case (1), which relates to the following settings: lexicographic branching choice (rule 1) using index order, fixed order branching, no dominance rule, no initial solution and upper bound UB. Settings (2) and (3) illustrate the successive improvements by using the schedule produced by the heuristic described in Section 7 as initial incumbent, and by resorting to a variable branching order. The table shows the number of instances solved to guaranteed optimality within the time limit, two efficiency measures (the average running time and the average number of nodes in the search tree) expressed as percentage of the best setting (3), and two efficacy measures (improvement from the initial solution to the output of setting (i), and improvement upon setting (i) by setting (3)). $\mathbf{s}_{(i)}$ is the output of the procedure run in setting (i); $\mathbf{s}_{(0)}$ refers to the schedule produced by Algorithm 1. The efficiency measures are computed only for the instances that are solved to guaranteed optimality by setting (3). Efficacy measure $I(\mathbf{s}_{(i)}, \mathbf{s}_{(3)})$ is computed only for the instances that are *not* solved to optimality by the setting (i).

The performance of the B&B algorithm without an initial solution (setting (1)) is rather poor. The incorporation of a variable branching order (from (2) to (3)) allows to solve more instances to optimality (from 45 to 52 out 60) and yields a 53% gain in CPU time for these 52 instances. Efficacy-wise, for the 15 instances not solved to optimality in

	opt	efficiency		efficacy	
	(/60)	nodes	CPU time	$I(\mathbf{s}_{(0)}, \mathbf{s}_{(i)})$	$I(\mathbf{s}_{(i)}, \mathbf{s}_{(3)})$
(1) base	40	1362%	1282%	+3.84%	+3627.16%
(2) = (1) + initial LB	45	146%	153%	+23.64%	+26.43%
(3) = (2) + var. br. order	52	100%	100%	+26.10%	0.00%
(4) = (3) with rule 2, index	24	269%	258%	+20.01%	+15.70%
(5) = (3) with rule 3, index	19	33939%	30191%	+19.51%	+32.36%
(6) = (3) with rule 4, index	50	163%	172%	+25.67%	+2.44%
(7) = (3) with rule 1, incr. float	48	1651%	1555%	+12.15%	+41.86%
(8) = (3) with rule 1, decr. float	23	39656%	34801%	+23.14%	+7.93%

Table 3: Computational results for different versions of the B&B algorithm. Efficiency measures are averaged only over the 52 instances solved by (3). Efficacy measure $I(\mathbf{s}_{(i)}, \mathbf{s}_{(3)})$ for setting (i) is averaged only over the instances not solved to guaranteed optimality by (i). $I(\mathbf{s}_{(0)}, \mathbf{s}_{(i)})$ is averaged over the entire dataset.

$n =$	10	15	20	25	30	35	40
opt (/60) ND	60	60	52	36	25	20	14
opt (/60) D	60	60	52	36	25	20	14
CPU time ND *	110%	106%	99%	106%	107%	114%	113%
$I(\mathbf{s}_{ND}, \mathbf{s}_D)$ **	-	-	+3.33%	+1.46%	+1.68%	+3.02%	+2.99%

Table 4: The impact of incorporating the dominance rule for different values of n.
* averaged only over the instances solved to optimality by D, and expressed as a percentage of the result for D.
** averaged only for the instances *not* solved to optimality by ND.

setting (2), the variable branching order achieves an average improvement in the objective function of some 26%.

As for the branching choice, we find that the simplest option possible is also the best: lexicographic branching choice (rule 1) using index order significantly outperforms all other branching rules ((4)–(8)). Various other rules have also been tried and found not to improve upon setting (3), including some more accurate (but considerably more time-intensive) estimates of the change in $UB^{(h)}$ for each alternative. We conjecture that this simple branching rule fits best into our overall search approach because it allows to examine each search node in the most efficient manner: the computational effort for processing each node is very low. We should point out that activity ordering in increasing or decreasing float values for (7) and (8) takes place only once at the beginning of the B&B algorithm, so that the time spent on sorting is negligible.

Table 4 examines the influence of the dominance rule. 'ND' refers to setting (3) in Table 3, 'D' adds the dominance rule to this setting. We observe that both for efficiency and for efficacy, the dominance rule improves the performance of the algorithm (except for a small dip for CPU time for $n = 20$). For the instances *not* solved to optimality, an average improvement in the objective function of between 1.5% and 3% is achieved. In the remainder of this text, when reference is made to 'the B&B algorithm', we always mean the algorithm corresponding with setting (3) and with the dominance rule.

9.3 Time limit

Since the problem at hand is NP-hard, no optimal polynomial-time algorithms are likely to exist so that we have to impose a time limit on our (exponential-time) B&B algorithm because it may otherwise take an inordinately long amount of time before terminating,

time limit	opt (/60)	nodes	$I(\mathbf{s}_{(0)}, \mathbf{s}_{(i)})$	$I(\mathbf{s}_{(i)}, \mathbf{s}_{(1000)})$
0	0	0	0.00%	+31.59%
1	20	62,240	+21.67%	+30.50%
5	23	274,900	+23.61%	+28.42%
20	28	996,001	+25.33%	+25.39%
100	35	4,045,108	+27.70%	+19.06%
250	38	8,958,914	+28.60%	+17.83%
1000	41	31,498,214	+31.59%	0.00%

Table 5: Performance of the B&B algorithm with varying time limits (in seconds), for $n = 25$. $I(\mathbf{s}_{(0)}, \mathbf{s}_{(i)})$ is computed for the entire dataset, $I(\mathbf{s}_{(i)}, \mathbf{s}_{(1000)})$ only for the instances *not* solved to guaranteed optimality by setting (i).

especially for large scheduling instances.

Table 5 examines the performance of the B&B procedure for various time limits with $n = 25$; $\mathbf{s}_{(i)}$ is the output of the procedure run with time limit i. A time limit of zero means that the actual branching procedure is never entered so that $\mathbf{s}_{(0)}$ is the output of Algorithm 1. When a time limit of 1000 seconds is imposed, 41 instances are solved to guaranteed optimality; this number gradually increases with the time limit from zero onwards. A running time of 1000 seconds allows to considerably improve the objective function of a number of instances, even when compared with 100 and 250 seconds. This result is a very strong indication that 'sophisticated' scheduling methods, such as our B&B algorithm, are valuable.

10. Insights

In this section we run a number of experiments in order to derive managerial insights. We find (Section 10.1) that adopting a simplistic schedule (e.g., doing all activities in series) may result in negative eNPV at time 0 and abandonment of a project, while using a more sophisticated scheduling approach such as our (truncated) B&B may result in a positive eNPV and the project being pursued. The benefits of advanced scheduling procedures will be significant especially for medium- to high-risk projects. Another valuable insight (discussed in Sections 10.1 and 10.4) is the fact that CPM-based schedules are good when the probability of failure is small and when the decision-maker is risk-seeking; longer schedules (with less parallel activities) tend to be better when the probabilities of failure are significant or when the decision-makers are risk-averse. These limits to the benefits of parallelization are also considered in Section 10.2. The non-intuitive behavior of the optimal schedule length as a function of the discount rate is treated in Section 10.3.

10.1 Benefits of advanced scheduling procedures

Table 6 presents results of the B&B algorithm for the dataset $n = 25$ (again with a two-minute time limit). The table contains a column 'uncertainty', in which we account for different levels of technical risk: 'medium' uncertainty refers to the situation where each activity's PTS is in interval [80%; 100%], which was the case in Section 9. 'high' and 'low' uncertainty relates to success probabilities within [60%; 100%] and [95%; 100%], respectively. The length $\lambda(\mathbf{s})$ of a schedule $\mathbf{s}$ is defined to be $s_n - s_0$; the 'length ratio' in Table 6 is zero if the length of the best-found schedule equals the critical-path length (remember that $\phi(A)$ is the CPM-based late-start schedule). E_C is the complete order on N that is obtained at the end of Algorithm 1.

We observe that, when the risk level of the project is relatively low, the CPM schedule performs quite well. Therefore, for low-risk projects the use of a simple CPM scheduling scheme seems to be warranted. One might intuitively expect a similar good performance for a serial schedule when project risks are high, but our results show that this is not the case: although for high-risk projects a serial schedule typically performs better than CPM, a further substantial improvement can be obtained by using the exact algorithm; B&B also does significantly better than our 'greedy' heuristic (see the final column of

	uncertainty	opt (/20)	length ratio	$I(\phi(A), \mathbf{s}_{BB})$	$I(\phi(E_C), \mathbf{s}_{BB})$	$I(\mathbf{s}_{(0)}, \mathbf{s}_{BB})$
	low	3	0.00%	0.17%	2318.00%	0.09%
$OS = 0.25$	medium	0	24.21%	69.00%	330.39%	30.35%
	high	0	122.07%	99.89%	99.24%	97.23%
	low	19	0.00%	0.09%	1662.17%	0.08%
$OS = 0.50$	medium	16	13.31%	57.56%	200.86%	38.99%
	high	16	25.99%	99.90%	99.65%	99.51%
	low	20	0.00%	0.02%	1552.52%	0.02%
$OS = 0.75$	medium	20	3.30%	48.34%	180.85%	15.80%
	high	20	7.42%	99.38%	98.65%	98.04%

Table 6: Investigation of the influence of OS and the degree of uncertainty for $n = 25$. 'length ratio' refers to the average of ratio $(\lambda(\mathbf{s}_{BB}) - \lambda(\phi(A)))/\lambda(\phi(A))$, with $\mathbf{s}_{BB}$ the schedule produced by our B&B algorithm after two minutes of running time; $\mathbf{s}_{(0)}$ is the schedule used to produce the initial lower bound.

Table 6). We conclude that optimally scheduling R&D projects, i.e. obtaining an optimal degree of parallelization, can result in a significantly higher project eNPV when compared to the CPM or serial schedule, and these benefits of advanced scheduling procedures will be significant especially for medium- to high-risk projects.

Even more important, perhaps, is the fact that the B&B algorithm is sometimes able to produce a positive-eNPV schedule for a project where both the CPM and serial approaches fail to do so. This would result in the project being cut from the portfolio using simple scheduling, whereas it would be able to add value given an optimal schedule to carry out the project. Dependent on the parameter settings (in particular on the risk level and on the value of r), this was the case for up to five out of the 60 instances in each dataset. Although the complexity of a project's structure, as measured by OS, also has an impact on the benefit of optimal scheduling, this effect is not as pronounced. Our results suggest that these benefits are higher when OS is relatively low, i.e. when there is more freedom in scheduling the activities: the number of 'undecided' activity pairs in ν_0 is higher for lower OS values. On a separate note, we observe (from the number of guaranteed optimal solutions) that problem difficulty is inversely related to OS; this goes hand in hand with the previous observation.

10.2 Limits to the benefits of parallelization

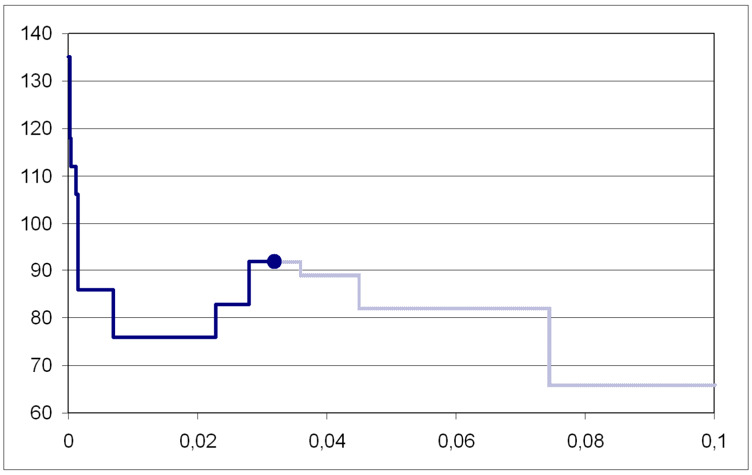
From Table 6 we can also observe that the schedule length $\lambda(\mathbf{s}_{BB})$ is often higher than the critical-path length $\lambda(\phi(A))$; the dependence on OS is important in this case. Although the present value of the project payoff decreases with increasing $\lambda(\mathbf{s})$ (at least for positive eNPV), performing certain activities in series rather than in parallel will sometimes allow to decrease the expected development cost of a project, resulting in an overall improvement in the project's eNPV. An optimal project schedule will need to balance information flows between activities against delays in final project payoff.

In line with the findings of Hoedemaker et al. (1999) but from a different perspective, we find that there are limits to the benefits of parallelization in R&D projects. This is especially so for highly uncertain environments: as randomness increases, good schedules become increasingly longer. This observation should be contrasted with projects without technical uncertainty (equivalent with the limit case where all $p_i = 1$), for which the CPM late-start schedule is optimal. Again we conclude that the use of 'simple' heuristics such as CPM is recommendable only when the degree of variability in the environment is very low: only in such cases, the added benefit of advanced scheduling procedures is marginal.

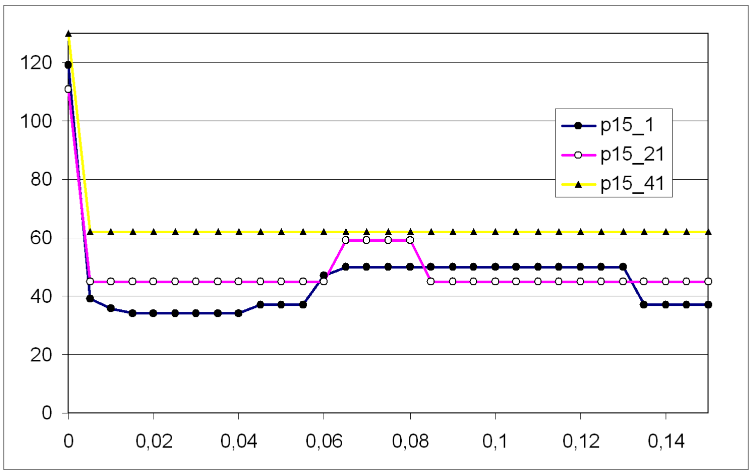
10.3 The influence of the discount rate

Intuitively, one would expect the incentive for parallelization to increase with increasing cost associated with project delay. We examine this behavior by means of varying the interest rate, representing the time value of money, for a constant project payoff.

Figure 5(a) contains the results for the example problem of Section 3: the graph shows the optimal schedule length as a function of the interest rate. With zero discount rate, the value of the project payoff is constant over time and the project schedule can take full advantage of information flows: no activity with PTS < 1 will be in parallel with other activities, which leads to maximum total length (this insight led to Theorem



(a) Schedule length versus interest rate for the example problem. The schedules corresponding with r-values beyond the dot have negative eNPV.



(b) Schedule length versus interest rate for three more instances.

Figure 5: The influence of the discount rate on schedule length.

1). As the interest rate goes up, we observe a reduction in the optimal schedule length: some activities are overlapped, forfeiting information flows for the sake of earlier project completion (with positive eNPV). Interestingly, however, from a certain point onwards a further increase in the interest rate induces an increase in the optimal project schedule length. The reason for this phenomenon is that the magnitude of the present-value change due to an incremental delay in a cash flow decreases as the interest rate becomes larger, and this effect is more marked for cash flows that occur later in time. As a consequence, cost savings early on in the project due to higher project duration may more than offset the associated decrease in the present value of the project payoff. Subsequently, once the objective function becomes negative (when $r = 0.03195$, indicated with a dot in the graph), all activities are scheduled against the deadline and the benefits of information exchange are again dominated by the discounting effect: schedule length decreases again.

Our observations are also illustrated by Figure 5(b), which shows the optimal schedule length for different values of the interest rate for three arbitrary instances of the dataset with $n = 15$: problems p15_1, p15_21 and p15_41 have $OS = 0.25, 0.5$ and 0.75, respectively. The pattern observed above is more markedly present for lower OS, presumably because this corresponds with the existence of more feasible solutions. As a comparison, the order strength of the example project of Section 3 also equals 0.5 – but graph 5(a) is based on a finer discretization of the values of r. The objective function becomes negative for a value of r in $[0.06; 0.065]$ for p15_1 and p15_21, and in $[0.04; 0.045]$ for p15_41.

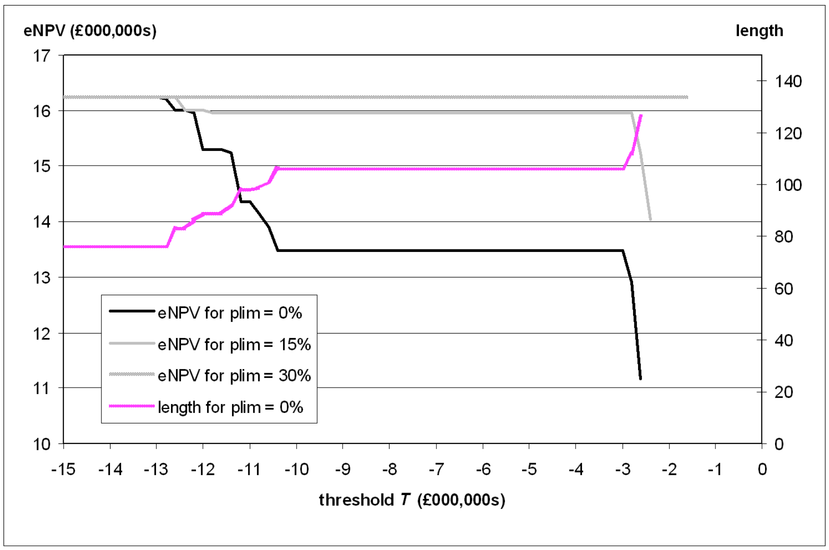
10.4 Impact of risk preferences

In order to examine the impact of risk preferences, we have performed a detailed analysis of the example problem that was presented in Section 3: we have opted for an illustration of risk preferences on one example project rather than on an entire dataset because, in our approach to dealing with risk preferences, appropriate thresholds are set manually.

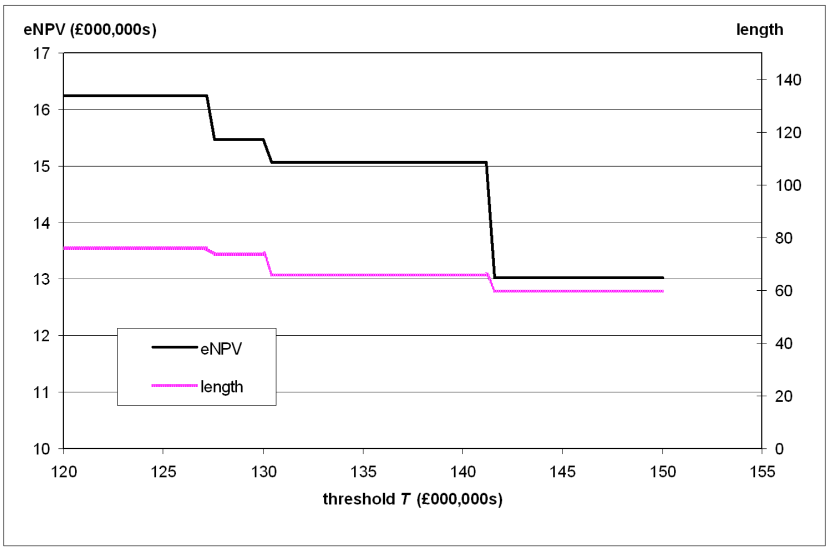
If $Z(\mathbf{s})$ denotes the random variable representing the NPV of schedule $\mathbf{s}$, then the pmf of $Z(\mathbf{s})$ is $f_{\mathbf{s}}(z)$. We model downside-risk preferences in the following way: for a given probability p_{lim} and a threshold T, the constraint is imposed that the probability that Z is lower than T should not be higher than p_{lim}. In other words, a candidate schedule $\mathbf{s}$ is acceptable only if

$$\int_{-\infty}^{T-\epsilon} f_{\mathbf{s}}(z)dz \leq p_{lim},$$

for any $\epsilon > 0$. In a similar way, we might implement a constraint on upside potential of



(a) Downside risk versus eNPV. The three curves associated with the first ordinate 'eNPV' represent $g(\mathbf{s}^D(p_{lim}, T))$ for three probability limits p_{lim} (which is written as 'plim'); the threshold T is on the abscissa. The second ordinate represents the length $\lambda(\mathbf{s}^D(0, T))$ of the optimal schedule for $p_{lim} = 0\%$ and applies to the only increasing curve.



(b) Upside potential versus eNPV. The two curves represent $g(\mathbf{s}^U(T))$ and $\lambda(\mathbf{s}^U(T))$, each with its own ordinate. The threshold T is on the abscissa.

Figure 6: Trade-off curves for downside risk and upside potential versus eNPV.

the form

$$\int_T^{+\infty} f_{\mathbf{s}}(z)dz \geq p_{lim},$$

but it can be seen that

$$\int_T^{+\infty} f_{\mathbf{s}}(z)dz = \begin{cases} q_n & \text{if } T \leq z^*(\mathbf{s}) \\ 0 & \text{otherwise} \end{cases}$$

with $z^*(\mathbf{s})$ representing the NPV of $\mathbf{s}$ in case of project success (the only positive realization of $Z(\mathbf{s})$), so that a value p_{lim} is not really useful in this case. We call $\mathbf{s}^D(p_{lim}, T)$ the schedule that optimizes $g(\mathbf{s})$ subject to the downside-risk constraint represented by p_{lim} and T, and similarly $\mathbf{s}^U(T)$ the schedule with highest value $g(\mathbf{s})$ subject to the upside-potential requirement with parameter T that $z^*(\mathbf{s}) \geq T$.

Figure 6(a) illustrates the trade-off between downside-risk preferences and expected return (the optimal eNPV without a risk constraint is £16.244 million). We observe decreasing eNPV-values with tighter value-at-risk constraints. As explained, in the particular setting of RDPSP, constraints on upside potential offer less freedom to the decision maker than down-risk preferences: positive cash flows are always obtained with probability q_n. Figure 6(b) describes the trade-off between *minimum* NPV in case of project success on the one hand, and expected NPV on the other hand.

Both graphs in Figure 6 were produced by invoking Algorithm 2 each time the B&B procedure found a new incumbent, and by accepting only those schedules that comply with the risk-preference constraint under consideration; the initial lower bound is not used. In both graphs, when the threshold is set too high, no feasible schedule can be found that meets the demands (which is where the curves end). Both graphs also depict the evolution of the optimal schedule length as a function of the risk preferences. We observe that a longer schedule (more activities in series) tends to be better if the decision maker is more risk averse, and that scheduling more activities in parallel becomes preferable when the decision maker is more risk seeking.

11. Summary and outlook on further research

In this article we have presented a model and algorithms for scheduling R&D projects to maximize the expected net present value of a project when the activities have an inherent possibility of failure and when individual activity default causes overall project failure. We have shown that this problem, referred to as the R&D-Project Scheduling Problem or RDPSP, is NP-hard and have developed a branch-and-bound algorithm that is able to produce optimal project schedules. As a side result, we have established a complexity result for an open problem in single-machine scheduling (the discounted weighted-completion-time objective with general precedence constraints).

We have observed that R&D project scheduling requires balancing early project completion with minimizing expected expenditures, and this balance is influenced by the degree of randomness in the planning environment. The benefits of advanced scheduling procedures turn out to be significant especially for medium- to high-risk projects. Other insights include the fact that CPM-based schedules are good when the probability of failure is small and when the decision-maker is risk-seeking; longer schedules (with less parallel activities) tend to be better when the probabilities of failure are significant and when the decision-makers are risk-averse.

The model we have presented and analyzed is rather stylized, and will not always be of immediate use for decision support. Decision makers faced with planning R&D projects in industry will often be confronted with resource constraints and duration uncertainty, an observation that was also made by Schmidt and Grossmann (1996) and Jain and Grossmann (1999). Further research is needed if optimal scheduling solutions are to be developed for realistically-sized scheduling problems with such additional complications. However, we are convinced that the insights and results provided in this paper can serve as guidelines in this process.

Another practically-relevant generalization of RDPSP is to make project payoff a function of the project completion time. The choice for a non-increasing function would be appropriate for most innovative projects: the earlier a new product enters the market, the longer it can benefit from a monopoly position and first-mover advantages, or the longer it can exploit a patent. A further open option for model extension is correlated activity success, an inherent characteristic of many R&D projects. Quantifying correlations may be difficult, however. The model can also be altered to include alternative sets of activities for which success is required for only one set, allowing to model the pursuit of alternative technologies. Finally, decision makers may also desire to take into account that some R&D activities can be performed in different ways, e.g. by allocating more or less money, resulting in different success probabilities associated with these multiple activity execution modes.

Appendix: proofs

Proof (Theorem 1): When $r = 0$, the objective function corresponding with feasible extension E is $q_n C + \sum_{i=1}^{n-1} q_i(E)c_i$, with q_n independent of the information-flow decisions; we also omit the argument to q_i in this proof. Each optimal feasible extension minimizes $\sum_{i=1}^{n-1} q_i|c_i|$. Consider an optimal feasible extension $E^{(0)}$ that is not a complete order. If no such relation exists, the theorem holds, otherwise, take an arbitrary activity $k \in N$ that is incomparable with at least one other element in N according to $E^{(0)}$. The expression to be minimized can now be written as follows, in which $\bar{E}^{(0)}$ is the set of unordered activity pairs that are incomparable according to $E^{(0)}$:

$$\sum_{i=1}^{n-1} q_i|c_i| = q_k|c_k| + \sum_{(i,k)\in E^{(0)}} q_i|c_i| + \sum_{\{i,k\}\in \bar{E}^{(0)}} q_i|c_i| + \sum_{(k,i)\in E^{(0)}} q_i|c_i|.$$

If we extend $E^{(0)}$ to $E^{(1)} = E^{(0)} \cup \{(k,i) : \{k,i\} \in \bar{E}^{(0)}\}$, the only term changing in the right-hand side of the foregoing equation is the third (it is multiplied by p_k). We conclude that the objective-function value associated with $E^{(1)}$ is at least as good as the value for $E^{(0)}$. Continuing in this way, we obtain a complete order E^* after at most $(n-2)$ iterations, whose objective-function value is at least as high as that of $E^{(0)}$. □

Proof (Theorem 2): We consider the following problem:

Problem Π
Instance: directed precedence graph $G(V, F)$, non-negative integer job durations d'_i and non-negative integer weights w_i for each $i \in V$.
Goal: find a single-machine schedule that is contiguous from time 0 for the jobs in V such that the precedence constraints are respected and $\Sigma_j w_j C_j$ is maximized, with C_j the completion time of job j.

Lenstra and Rinnooy Kan (1978) show that problem $1|prec|\Sigma w_j C_j$ (the single-machine scheduling problem with general precedence constraints and weighted-completion-time objective) is strongly NP-hard by means of a reduction from OPTIMAL LINEAR ARRANGEMENT. From this result, NP-hardness of the *maximization* of the weighted sum of completion times $\Sigma_j w_j C_j$, i.e. of problem Π, is immediate, as can be seen by reversing the precedence constraints.

For an arbitrary instance of Π, we construct an instance of RDPSP, as follows. The set of activities $N = V \cup \{0, n\}$, with $n = |V| + 1$. We have non-positive activity cash flows $c_i = -w_i$ and durations $d_i = 1, \forall i \in V$. $C = 0$ and $(i, n) \in A, \forall i \in N_n$; $c_0 = d_0 = 0$, $p_0 = 1$ and $(0, i) \in A, \forall i \in N_0$. For each activity $i \in V$, we set probability $p_i = (1 - d'_i/M)$ with non-negative integer $M \geq d'_{\max} = \max_{i \in V}\{d'_i\}$ (further specification of M follows). Let $\delta = |V|$ and $r = 0$, so that an optimal solution to RDPSP that does not correspond with a complete order on N can be re-arranged in polynomial time into a complete order with equal objective function (as outlined in the proof of Theorem 1). Consider such an optimal complete order and let $[i]$ represent the job from V in the i-th position. Since $s_{[1]} = 0$, the objective-function value of the thus-built RDPSP-instance equals

$$c_{[1]} + \sum_{j=2}^{n-1} c_{[j]} \prod_{i=1}^{j-1} p_{[i]} = c_{[1]} + \sum_{j=2}^{n-1} c_{[j]} \prod_{i=1}^{j-1} \left(1 - \frac{d'_{[i]}}{M}\right), \tag{A1}$$

with

$$\prod_{i=1}^{j-1} \left(1 - \frac{d'_{[i]}}{M}\right) = 1 - \frac{1}{M} \sum_{k=1}^{j-1} d'_{[k]} + \frac{1}{M^2} \sum_{k=1}^{j-2} \sum_{l=k+1}^{j-1} d'_{[k]} d'_{[l]} - \ldots$$

so that the objective function (A1) becomes

$$\sum_{j=1}^{n-1} c_j - \frac{1}{M} \sum_{j=2}^{n-1} c_{[j]} \sum_{k=1}^{j-1} d'_{[k]} + \sum_{j=2}^{n-1} c_{[j]} \sum_{i=2}^{j-1} \left(\frac{-1}{M}\right)^i \sum_{\substack{\text{all sets } K: |K| = i, \\ K \subseteq \{1, 2, \ldots, j-1\}}} \prod_{k \in K} d'_{[k]}.$$

The first term in this expression is a constant. We want the impact of a change of a single unit in quantity $\sum_{j=2}^{|V|} c_{[j]} \sum_{k=1}^{j-1} d'_{[k]}$ (the weighted sum of the starting times in Π) to be larger than the largest possible change in all remaining terms, so that any optimal solution to the RDPSP-instance automatically optimizes this weighted sum. We impose

$$\frac{1}{M} > \sum_{j=2}^{n-1} c_{\max} \sum_{i=2}^{j-1} \frac{1}{M^i} \sum_{\substack{\text{all sets } K: |K| = i, \\ K \subseteq \{1, 2, \ldots, j-1\}}} \prod_{k \in K} d'_{\max}, \tag{A2}$$

with $c_{\max} = \max_{i \in V}\{|c_i|\}$. The right-hand side of Eq. (A2) is smaller than or equal to

$$\sum_{j=2}^{n-1} c_{\max} \sum_{i=2}^{j-1} \left(\frac{d'_{\max}}{M}\right)^i \binom{j-1}{i}$$

and this expression in turn is strictly smaller than

$$c_{\max} \left(\frac{d'_{\max}}{M}\right)^2 2^{n-1}$$

since $d'_{\max} \leq M$, $\sum_{i=2}^{j-1} \binom{j-1}{i} < 2^{j-1}$ and $\sum_{j=2}^{n-1} 2^{j-1} < 2^{n-1}$. This leads us to the conclusion that Eq. (A2) holds when

$$M = c_{\max} d'^2_{\max} 2^{n-1}.$$

For this M-value, we have shown that any job sequence maximizing Eq. (A1) also maximizes the weighted sum of completion times in Π minus constant $\sum_{i\in V} w_i d'_i$. This description establishes a polynomial-time transformation from Π to RDPSP. Since this proof of intractability of RDPSP clearly depends on the fact that large (exponential) input numbers are allowed, we can only conclude NP-hardness in the ordinary sense. □

Proof (Lemma 1): We use Γ to refer to the problem under study; C_j again represents the completion time of job j. Consider an instance of Γ for set of activities V to be scheduled with weights w_i and durations d'_i, $i \in V$. Knowing that

$$e^{-rC_j} = 1 - rC_j + (rC_j)^2/2 - (rC_j)^3/3! + (rC_j)^4/4! - \ldots,$$

we see that

$$\sum_{j\in V} w_j(1 - e^{-rC_j}) = r\sum_{j\in V} w_j C_j + \sum_{j\in V} w_j \left(-\frac{(rC_j)^2}{2} + \frac{(rC_j)^3}{3!} - \ldots\right). \tag{A3}$$

We examine under which conditions the effect of the change of a single unit in the weighted sum of completion times (the first term in the right-hand side of (A3)) is larger than the largest possible joint effect of all remaining terms. This is true when

$$r > w_{\max} \sum_{j\in V} \left(+ \left(\frac{rC_j}{2}\right)^2 + \left(\frac{rC_j}{3!}\right)^3 + \ldots \right)$$

with $w_{\max} = \max_{j\in V}\{w_j\}$, which is certainly true under the stronger condition

$$r > w_{\max}|V| \left((rT)^2 + (rT)^3 + \ldots\right),$$

where $T = \sum_{j\in V} d'_j$. If we impose $rT < 1$, this is equivalent with

$$r > w_{\max}|V| \left((rT)^2 / (1 - rT)\right)$$

or (if $r > 0$)

$$r < \left(w_{\max}|V|T^2 + T\right)^{-1}.$$

The foregoing provides all the necessary elements for the construction of a polynomial-time reduction from the strongly NP-hard problem $1|prec|\Sigma w_j C_j$ to Γ, and the maximum number in the resulting instance of Γ is polynomially bounded such that strong NP-hardness of Γ is established. $1|prec|\Sigma w_j C_j$ remains strongly NP-hard in case of unit durations (Lenstra and Rinnooy Kan, 1978), which concludes the proof of the lemma. □

Proof (Theorem 3): The proof of Lemma 1 is easily adapted to show that single-machine scheduling with general precedence constraints and discounted weighted-*starting*-time objective ($1|prec|\Sigma w_j(1 - e^{-rs_j})$ in the standard three-field notation), with the objective to be *maximized*, is also strongly NP-hard. The result is then straightforward. □

Proof (Lemma 2): All intermediate activities are started as late as possible. The only reason why an activity would not end *exactly* at the start of its earliest starting successor in E_h, is because it needs to be in parallel with some other activity. If we iteratively remove all parallelity constraints for this activity and shift it later in time until it ends exactly at its earliest successor starting time, there is no effect on the contribution to the objective function of any of the other activities. On the other hand, the (negative) contribution of the activity itself to the objective function goes down, first of all because of the discounting effect, and second also because additional activities may now end before or at the starting time of the activity itself, which would allow a further reduction of its expected NPV via extra information flows. □

References

Abernathy, W.J., R.S. Rosenbloom. 1969. Parallel strategies in development projects. *Management Science* **15** B486–B505.

Adlakha, V.G., V.G. Kulkarni. 1989. A classified bibliography of research on stochastic PERT networks: 1966-1987. *INFOR* **27**(3) 272–296.

Bard, J.F. 1985. Parallel funding of R&D tasks with probabilistic outcomes. *Management Science* **31**(7) 814–828.

Bartusch, M., R.H. Möhring, F.J. Radermacher. 1988. Scheduling project networks with resource constraints and time windows. *Annals of Operations Research* **16** 201–240.

Blau, G., B. Mehta, S. Bose, J. Pekny, G. Sinclair, K. Keunker, P. Bunch. 2000. Risk management in the development of new products in highly regulated industries. *Computers and Chemical Engineering* **24** 659–664.

Boros, E., T. Ünlüyurt. 1999. Diagnosing double regular systems. *Annals of Mathematics and Artificial Intelligence* **26** 171–191.

Butterworth, R. 1972. Some reliability fault-testing models. *Operations Research* **20** 335–343.

Chiu, S.Y., L.A. Cox Jr., X. Sun. 1999. Optimal sequential inspections of reliability systems subject to parallel-chain precedence constraints. *Discrete Applied Mathematics* **96–97** 327–336.

Crama, P., B. De Reyck, Z. Degraeve, W. Chong. 2006. R&D project valuation and licensing negotiations at Phytopharm plc. *Interfaces*, forthcoming.

Dahan, E. 1998. Reducing technical uncertainty in product and process development through parallel design of prototypes. *Working Paper, Graduate School of Business, Stanford University.*

Demeulemeester, E., M. Vanhoucke, W. Herroelen. 2003. A random generator for activity-on-the-node networks. *Journal of Scheduling* **6** 13–34.

Denardo, E.V., U.G. Rothblum, L. Van der Heyden. 2004. Index policies for stochastic search in a forest with an application to R&D project management. *Mathematics of Operations Research* **29** 162–181.

DiMasi, J.A. 2001. Risks in new drug development: approval success rates for investigational drugs. *Clinical Pharmacology and Therapeutics* **69** 297–307.

Ding, M., J. Eliashberg. 2002. Structuring the new product development pipeline. *Management Science* **48** 343–363.

Eppinger, S.D., D.E. Whitney, R.P. Smith, D.A. Gebala. 1994. A model-based method for organizing tasks in product development. *Research in Engineering Design* **6** 1–13.

Gassmann, O., G. Reepmeyer, M. von Zedtwitz. 2004. *Leading Pharmaceutical Innovation. Trends and Drivers for Growth in the Pharmaceutical Industry.* Springer-Verlag, Berlin Heidelberg New York.

Granot, D., D. Zuckerman. 1991. Optimal sequencing and resource allocation in research and development projects. *Management Science* **37** 140–156.

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

Herroelen, W.S., P. Van Dommelen, E.L. Demeulemeester. 1997. Project network models with discounted cash flows: a guided tour through recent developments. *European Journal of Operational Research* **100** 97–121.

Herroelen, W.S., R. Leus. 2005. Project scheduling under uncertainty, survey and research potentials. *European Journal of Operational Research* **165**(2) 289–306.

Hill, A.V. 2002. *The Encyclopedia of Operations Management Terms.* Available on-line as: http://www.poms.org/POMSWebsite/EducationResources/omencyclopedia.pdf.

Hoedemaker, G.M., J.D. Blackburn, L.N. Van Wassenhove. 1999. Limits to concurrency. *Decision Sciences* **30**(1) 1–18.

Jain, V., I.E. Grossmann. 1999. Resource-constrained scheduling of tests in new product development. *Industrial and Engineering Chemistry Research* **38** 3013–3026.

Krishnan, V., S. Bhattacharya. 2002. Technology selection and commitment in new product development: the role of uncertainty and design flexibility. *Management Science* **48** 313–327.

Krishnan, V., S.D. Eppinger, D.E. Whitney. 1997. A model-based framework to overlap product development activities. *Management Science* **43**(4) 437–451.

Lawler, E.L. 1976. *Combinatorial Optimization: Networks and Matroids.* Holt, Rinehart and Winston, New York.

Lenstra, J.K., A.H.G. Rinnooy Kan. 1978. Complexity of scheduling under precedence constraints. *Operations Research* **26** 22–35.

Lockett, A.G., A.E. Gear. 1973. Representation and analysis of multi-stage problems in R&D. *Management Science* **19**(8) 947–960.

Mastor, A.A. 1970. An experimental and comparative evaluation of production line balancing techniques. *Management Science* **16** 728–746.

Mitten, L.G. 1960. An analytic solution to the least cost testing sequence problem. *Journal of Industrial Engineering* **11** 17–17.

Möhring, R.H. 2001. Scheduling under uncertainty: bounding the makespan distribution. *In*: Alt, H. (ed.). *Computational Discrete Mathematics, Advanced Lectures.* Lecture Notes in Computer Science 2122, Springer.

Monma, C.L., J.B. Sidney. 1979. Sequencing with series-parallel precedence constraints. *Mathematics of Operations Research* **4** 215–224.

Neumann, K., C. Schwindt, J. Zimmermann. 2003. *Project Scheduling with Time Win-*

dows and Scarce Resources: Temporal and Resource-Constrained Project Scheduling with Regular and Nonregular Objective Functions. Springer-Verlag, 2nd edition.

Padman, R., D.E. Smith-Daniels, V.L. Smith-Daniels. 1997. Heuristic scheduling of resource-constrained projects with cash flows. *Naval Research Logistics* **44**(4) 365–381.

Parker, R., R. Rardin. 1988. *Discrete Optimization.* Academic Press.

Pinedo, M. 2002. *Scheduling. Theory, Algorithms, and Systems.* Prentice-Hall, Inc., Upper Saddle River, NJ.

Schmidt, C.W., I.E. Grossmann. 1996. Optimization models for the scheduling of testing tasks in new product development. *Industrial and Engineering Chemistry Research* **35** 3498–3510.

Shah, N. 2004. Pharmaceutical supply chains: key issues and strategies for optimisation. *Computers and Chemical Engineering* **28** 929–941.

Ünlüyurt, T. 2004. Sequential testing of complex systems: a review. *Discrete Applied Mathematics* **142** 189–205.

Weitzman, M.L. 1979. Optimal search for the best alternative. *Econometrica* **47** 641–654.

Zemel, D., I. David, A. Mehrez. 2001. On conducting simultaneous versus sequential engineering activities in risky R&D. *International Transactions in Operational Research* **8** 585–601.

Zipfel, A. 2003. Modeling the probability-cost-profitability architecture of portfolio management in the pharmaceutical industry. *Drug Information Journal* **37** 185–205.