Computers & Operations Research 40 (2013) 395–405

Contents lists available at SciVerse ScienceDirect

Computers & Operations Research

journal homepage: www.elsevier.com/locate/caor

An exact method for scheduling of the alternative technologies in R&D projects

Mohammad Ranjbar [a,*], Morteza Davari [b]

[a] Department of Industrial Engineering, Faculty of Engineering, Ferdowsi University of Mashhad, Mashhad, P.O. Box: 91775-1111, Iran
[b] Faculty of Business and Economics, KULeuven, Belgium

ARTICLE INFO

Available online 16 July 2012

Keywords:
Project scheduling
Technology
Research and development
Innovation

ABSTRACT

A fundamental challenge associated with research or new product development projects is identifying that innovative activity that will deliver success. In such projects, it is typically the case that innovative breakthroughs can be achieved by any of several possible alternative technologies, some of which may fail due to the technological risks involved. In some cases, the project payoff is obtained as soon as any single technology is completed successfully. We refer to such a project as alternative-technologies project and in this paper we consider the alternative-technologies project scheduling problem. We examine how to schedule alternative R&D activities in order to maximize the expected net present value, when each technology has a cost and a probability of failure. Although a branch-and-bound algorithm has been presented for this problem in the literature, we reformulate the problem and develop a new and improved branch-and-bound algorithm. We show using computational results that the new algorithm is much more efficient and outperforms the previous one.

1. Introduction

The development of complex and innovative products is characterized by much uncertainty. In order to deal with this uncertainty, it has been suggested that research and development (R&D) projects should pursue multiple alternative solutions for developing the new products (see, for instance, [1] and [2]). The scheduling of these attempts, hereafter referred to as alternatives, is crucial for increasing the likelihood of successfully developing a product, minimizing development time and obtaining revenues as early as possible. Consider, for instance, a software development firm that has the option to develop their web services using either a traditional Java SPRING framework or the pioneering Ruby-on-Rails framework. While both might achieve a similar functionality, the traditional Java SPRING framework will take longer to develop, but is more likely to handle the expected volume of users. A similar situation happens in the formulation, delivery and packaging development phase of the pharmaceutical drug-development process in which drug developers must devise a formulation that ensures the proper drug delivery parameters. It is critical to begin looking ahead to clinical trials at this phase of the drug development process. Drug formulation and delivery may be refined continuously until, and even after, the drug's final approval. Trials have different costs, durations and probability of success, and optimal scheduling of these trials saves a noticeable amount of money for the drug developer firm (see [3]).

In this paper, we focus on a single firm engaged in a single R&D or new product development (NPD) project. The project can be achieved by any one of several given alternatives. Each alternative is characterized by a cost, a duration and a probability of technical success (*PTS*). The successful completion of an alternative corresponds to the completion of the project and obtaining the project payoff. In other words, depending on the schedule and the realized successes of alternatives, some alternatives of the project will not be performed. Also, if in the time at which the success of an alternative is realized, there are some other alternatives in progress, they will be ignored. Since it is assumed that the cost of each alternative is incurred at the beginning of alternative while the project payoff will be obtained at the end of a successful alternative, there is the downside risk of disregarding some in progress alternatives. A serial schedule, in which alternatives are not attempted simultaneously, is conservative in terms of costs and minimizing the downside risk, but might result in the maximum project duration. On the other extreme, simultaneously developing all the potential alternative technologies, which could lead to the minimum project duration and an earlier launch date, carries a large downside risk and higher upfront costs (see, [4] and [5]). Our goal is to analyze such trade-offs and to solve the underlying optimization problem, which will be referred to as the

* Corresponding author. Tel.: +98 511 8805092.
E-mail addresses: m_ranjbar@um.ac.ir (M. Ranjbar), morteza.davari@student.kuleuven.be (M. Davari).

0305-0548/$ - see front matter © 2012 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.cor.2012.07.005

Alternative-Technologies Project Scheduling Problem (ATPSP). The goal of the problem is to determine optimal timing of the alternatives such that the project's expected net present value (*eNPV*) is maximized. The most related problem to the ATPSP has been proposed by Creemers et al. [6], who develop a dynamic programming approach to solve a modular R&D project scheduling problem (RDPSP). Although modular RDPSP and ATPSP both try to use the advantages of alternative technologies, they differ in their objectives. Unlike ATPSP in which the objective function directly relates to the scheduling of alternatives, in modular RDPSP, the objective function relates to alternatives only indirectly, and mostly depends on the probability of success of each module. In other words, a solution is feasible in modular RDPSP if all of the modules succeed.

Planning and scheduling of NPD activities has been a challenging subject of research in recent years. Dahan [7] examines the trade-off between parallel and sequential scheduling in alternative prototype development. Granot and Zuckerman [8] examine the sequencing of R&D projects with success or failure in individual activities. Ding and Eliashberg [9] examine the 'pipeline problem': since 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. Loch et al. [10] discuss the importance of exploratory learning and the value of partial information, thus highlighting the need for combined parallel and sequential planning. Also, Sobel et al. [11] consider the problem of scheduling projects with stochastic activity duration to maximize expected net present value.

De Reyck et al. [12] have presented a complete literature survey on project scheduling with activity failures. Following the classification introduced by De Reyck et al. [12], the project we consider in the ATPSP can be classified as a single-module project. Also, De Reyck and Leus [13] consider RDPSP in which project activities are interrelated by finish to start precedence relations. In their model, however, the project is successful only if all individual activities succeed. They develop a specialized branch-and-bound (B&B) algorithm for the RDPSP which includes two phases. In the first phase, a feasible extension for set of precedence relations is generated and in the second phase, each activity can be scheduled to end at the earliest of the start times of its successors in the extended set of precedence relations. Although there are similarities between ATPSP and RDPSP, the project success is achieved in the ATPSP if one alternative succeeds while this achievement is obtained in the RDPSP when no activity fails. Intuitively, we perceive that the number of situations in which the project is terminated in the ATPSP is much greater than in the RDPSP, and we conjecture that the ATPSP is harder to solve than the RDPSP. If we want to apply the B&B algorithm developed by De Reyck and Leus [13] for the RDPSP to the ATPSP, the second phase of this algorithm cannot be used because in the ATPSP, all intermediate alternative cash flows are not negative and, hence, in the optimal solution of this problem each alternative is not necessarily ended at the earliest of the start times of its successors. In other words, for the second phase, a new search methodology is required to find the optimal start time of each alternative.

Ranjbar [14] developed a two-phase solution procedure for the ATPSP which consists of a B&B algorithm that uses a recursive search procedure, developed by Vanhouck [15], as a subroutine to obtain an optimal solution. He presents each solution of the ATPSP as a sequence of start and finish events, and searches in the space of possible sequences for the optimal solution. The weakness point of his work was that the size of sequence is twice the size of alternatives; thus, his procedure is able to solve projects including at most eight alternatives in a reasonable time.

The contributions of this article are threefold: (1) we reformulate the ATPSP as a non-linear integer programming model; (2) we prove a property, referred to as the concurrency property, for the optimal solution of the ATPSP; and (3) we construct a new and improved B&B algorithm based on the concurrency property for the ATPSP.

The remainder of this paper is organized as follows. We illustrate an example in Section 2. The problem modeling and properties are presented in Section 3. In Section 4, we present the B&B algorithm. Computational results are discussed in Section 5. Finally, conclusions are given in Section 6.

2. An example

As an example, we consider a project with five alternative technologies for achieving a given breakthrough. The alternatives are represented by the nodes in Fig. 1, finish-to-start precedence constraints between the technologies are depicted by directed arcs.

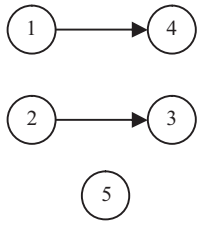

Fig. 1. Example project.

Table 1
Project data.

Alternative	Cost ($)	Duration (months)	*PTS*
1	−51	8	0.73
2	−31	6	0.62
3	−87	3	0.91
4	−28	7	0.57
5	−80	4	0.86

Other project data are given in Table 1. In this example, we assume a discount rate of 5% per month and a project payoff, achieved in case of technological success, is $2770. Also, we assume the project deadline is 29 months.

For scheduling these alternatives, several choices can be made. If we try to obtain the project payoff as soon as possible, we can execute the alternatives according to the early-start schedule determined by the Critical Path Method (CPM). This schedule, depicted in Fig. 2(a), results in an *eNPV* of $2058.96. In this schedule, if alternative 5 is successful, the firm must still pay the expenses associated with alternatives 1 and 2, which are planned to start prior to completion of alternative 5, since the discovery of success or failure of the alternatives takes place only at the end of the alternative.

Another option is to schedule the alternatives carrying technical risk in series, thereby avoiding unnecessary costs when an alternative succeeds. One such series schedule is depicted in Fig. 2(b); this schedule results in an *eNPV* of $2083.61. Finally, a schedule allowing for a partial overlap of alternatives is shown in Fig. 2(c), yielding an *eNPV* of $2104.16, which can be shown to be optimal. Finding such a schedule is the objective of the algorithms that will be presented in this paper.

For each of the three foregoing schedules, the cumulative distribution function of the project's NPV is depicted in Fig. 3. For this project we observe that, while the downside is the most limited in the serial schedule, the CPM schedule has the lowest variance and the optimal schedule has the highest *eNPV*.

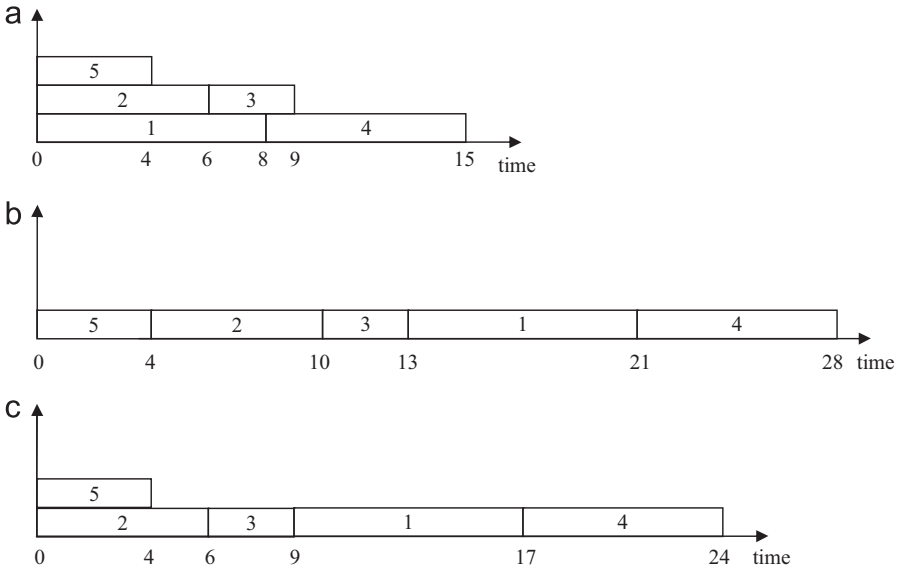

Fig. 2. Three possible schedules for the example project. (a) CPM early-start schedule, (b) Serial schedule and (c) Optimal schedule.

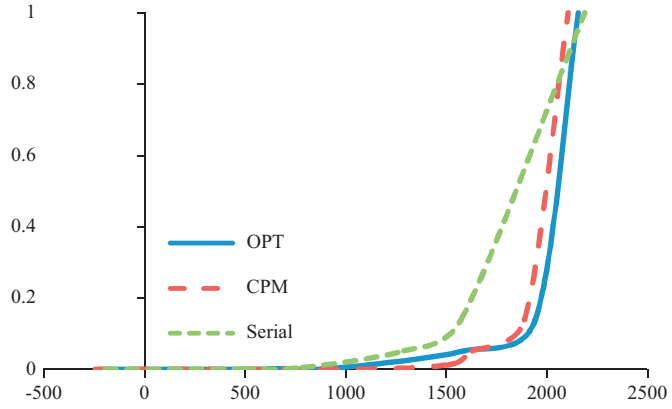

Fig. 3. The cumulative distribution functions of the project's NPV for each of the three schedules for the example project.

3. Problem modeling and properties

3.1. Definitions and problem modeling

We focus on a firm that is engaged in an NPD project and tries to maximize its project's *eNPV* by determining which alternative technologies to pursue and when. The development will end as soon as an alternative is successful, at which time the project terminates and the firm obtains the project payoff. Each alternative includes two time events: start event and finish event. The alternatives are executed without preemption from start to finish, and the alternative's outcome, which can be either success or failure, is revealed at the finish. Consequently, each start event will only occur if all alternatives with earlier finish events have had a negative outcome. We do not consider resource constraints and duration uncertainty and view the *PTS* of each of the alternatives as independent. The model's parameters are defined in Table 2.

Precedence constraints imposed by A might represent repetitive testing or trials, and also allow to model fallback options: alternative plans devised by management in the event the primary option falters (see [16]).

A schedule $\boldsymbol{s}$ is feasible if it respects the constraints imposed by A. The objective is to maximize the project's *eNPV*, and so each alternative's cost is weighted by the probability of joint failure of the already finished alternatives. We represent each schedule by a vector of start times $\boldsymbol{s}=(s_1,s_2,\ldots,s_n)$ where s_i is a non-negative integer and indicates the start time of alternative i. We also consider $\boldsymbol{f}=(f_1,f_2,\ldots f_n)$ as the vector of finish times where $f_i=s_i+d_i$ indicates the finish time of alternative i.

Table 2
Parameters.

Parameter	Definition
N	Set of alternatives, $N=\{1,2,\ldots,n\}$
c_i	Cost of alternative $i\in N$, a non-positive integer; incurred at the start of the alternative
C	End-of-project positive payoff, integer
d_i	Duration of alternative $i\in N$, a positive integer
p_i	*PTS* of alternative $i\in N$
r	Continuous discount rate
A	(Strict) partial order on N, an irreflexive and transitive relation imposing the constraints $s_i+d_i\le s_j$ for all $(i,j)\in A$
δ	The project deadline

The problem ATPSP can now be formulated as the following non-linear integer programming model.

$$\max g(\boldsymbol{s}) = \sum_{i=1}^{n}\left(\exp(-rs_i)c_i\prod_{j\in FBA(s_i)}(1-p_j)\right) + C\sum_{t\in NUF}\left(\exp(-rt)\left(1-\prod_{k\in FA(t)}(1-p_k)\right)\prod_{j\in FB(t)}(1-p_j)\right) + C\sum_{i:f_i\notin NUF}\left(\exp(-r(s_i+d_i))p_i\prod_{j\in FB(s_i+d_i)}(1-p_j)\right)$$

subject to

$$s_i+d_i\le s_j \quad \forall (i,j)\in A$$
$$s_i+d_i\le \delta \quad \forall i\in N$$
$$s_i\in\mathbb{Z}^+ \quad \forall i\in N$$

In the objective function $g(\mathbf{s})$ which indicates *eNPV* of schedule $\boldsymbol{s}$, $FBA(s_i)$ indicates the set of alternatives which are finished before or at start time of alternative i. Also, $FB(f_i)$ shows the set of alternatives which are finished before finish time of alternative i. In addition, *NUF* show the set of non-unique finish times such that every $t\in NUF$

is identical to the finish time of at least two alternatives. The set of alternatives which are finished at $t \in NUF$ are shown by $FA(t)$. In addition, $FB(t)$ shows the set of alternatives which are finished before time instant t. Objective function includes three main terms in which exp(.) shows exponential function. The first term indicates the expected costs in which $\prod_{j \in FBA(s_i)}(1-p_j)$ indicates the probability of joint failure of all alternatives finished before or at the start time of alternative i. We assume that if in a schedule for two alternatives $i,j \in N$, we have $f_j = s_i$, then alternative i will be started if alternative j has been failed. The summation of two other terms in the objective function indicates the expected revenue. The second term displays the expected revenue for the alternatives whose finish times belong to NUF. In this formula, $(1-\prod_{k \in FA(t)}(1-p_k))$ shows the probability of succeeding at least one of the alternatives finished at time instant t. Moreover, $\prod_{j \in FB(t)}(1-p_j)$ indicates the probability of joint failure of all alternatives finished prior to the time instant t. Finally, the last term specifies the expected revenue for the alternatives with unique finish times. In this formula, $\prod_{j \in FB(s_i+d_i)}(1-p_j)$ represents the probability of joint failure of all alternatives finished prior to the completion time of alternative i.

The first constraint of the model indicates the finish-to-start precedence relations among alternatives imposed by set A. The second constraint implies that all alternatives must be finished before or at the given deadline. If this constraint is not considered, some alternatives may be finished at infinite. The last constraint shows that all start times are non-negative integers ($\mathbb{Z}^+$ indicates the set of non-negative integers).

The ATPSP can be shown to be NP-hard (Theorem 1 in [12]), even with unit durations and $r=0$. Consequently, an exact algorithm with better than exponential time complexity is unlikely to exist. A number of sub-problems with $r=0$ can be distinguished, however, that can be solved in polynomial time. One example is the case $A=\emptyset$, for which Price [17] shows that a serial schedule executing the activities in non-decreasing order of c_i/p_i is optimal. Other special cases (generalizing the result of Price) are surveyed by De Reyck and Leus [13].

3.2. An overall view of the solution approach

In this section, we draw an overall view of the solution approach. A detailed description of our solution approach is provided in Section 4.

Our solution algorithm is a depth-first B&B approach but in a primary phase, we try to reduce the size of the problem and facilitate the solution algorithm using a preprocessing procedure. The optimal schedule of each ATPSP divides alternatives into two subsets, i.e., favorable (F) and unfavorable (U). The former category consists of alternatives that have no negative impact on the objective function $g(\boldsymbol{s})$ while the alternatives of latter category have negative impact. Generally, there is a time interval in which no alternative is in progress, referred to as a "*gap*", between these two sets of alternatives in the optimal schedule. The preprocessing procedure tries to determine that each alternative is favorable or unfavorable but unfortunately, it cannot decide about all alternatives. In other words, the output of the preprocessing procedure partitions all alternatives into three categories: favorable, unfavorable and undecided. The effectiveness of the preprocessing procedure depends on the number of alternatives determined to be favorable or unfavorable. After this primary phase, the size of the problem is reduced by removing unfavorable alternatives. Also, the size of the solution space of the new problem inversely depends on the number of favorable alternatives.

Our B&B algorithm relies on the concurrency property of the ATPSP that was conjectured by De Reyck et al. [12]. In a given schedule, we call an alternative $i \in N$ concurrent if its start event or its finish event is concurrent with at least another (start or finish) event. If all alternatives of a given schedule are concurrent, then the concurrency property is satisfied for the schedule and it is called concurrent schedule. We prove that there is at least one optimal concurrent schedule for each instance of the ATPSP.

Theorem 1. For each instance of the ATPSP, there is a concurrent optimal schedule.

Proof. Appendix A.

In our B&B algorithm, we limit ourselves to enumerating only concurrent schedules. This algorithm adds alternatives one-by-one to a partial schedule until a complete schedule is obtained. For each new (non-scheduled) alternative, it considers different start times subject to precedence relations. Each start time of a new alternative is determined based upon concurrency of the start time or the finish time of the alternative with available events in the partial schedule. Since there is a gap between sets F and U, for favorable (unfavorable) alternatives we consider only possible start times before (after) the gap while for undecided alternatives both cases must be examined. We consider two different variations of B&B algorithm, F-B&B and FB-B&B. The F-B&B algorithm assumes that all alternatives are favorable, while FB-B&B does not. The F-B&B is faster in average than the FB-B&B and works based on forward scheduling while the FB-B&B works based upon forward-backward scheduling.

3.3. Preprocessing

In the preprocessing procedure, we try to determine as many elements of the sets F and U as possible. In order to better understand the concept of favorable and unfavorable alternatives, we draw the overall sketch of an optimal schedule as shown in Fig. 4. In the optimal schedule of ATPSP, a set of alternatives (F) such that at least one of them is started at time instant $t=0$ is executed. Assume the latest finished alternative from this set completes at time instant t_1. In each time interval $[t,t+1]$ where $0 \le t \le t_1-1$, at least one of the favorable alternatives is scheduled to be executed. The earliest start time among alternatives of set U is shown by t_2 where $t_2 > t_1$ and the latest finish time among these alternatives is $t=\delta$. Also, in each time interval $[t,t+1]$ where $t_2 \le t \le \delta-1$, at least one of the unfavorable alternatives is scheduled. Thus, each unfavorable alternative is scheduled to be started after all favorable alternatives and the gap between F and U. Obviously, if $t_2=\delta$, then $U=\emptyset$ and all alternatives will be favorable. Similarly, if $t_1=0$, then $F=\emptyset$ and all alternatives will be unfavorable. Since unfavorable alternatives have a negative impact on the objective function, for the risk aversion firms, it is not recommended to carry them out in practice. For the risk seeking firms which are interested in testing all of their chances of obtaining the project payoff, the set of unfavorable alternatives must be scheduled optimally.

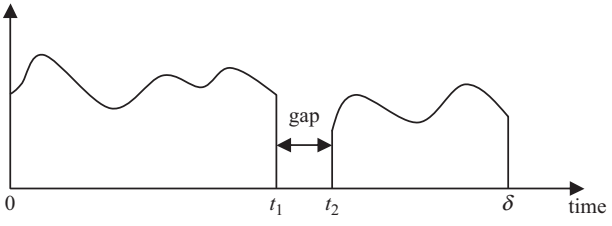

Fig. 4. Overall sketch of an optimal solution of ATPSP.

It is easily concluded that if alternative $i \in U$, then all successors (direct and indirect) of alternative i, referred to as $Suc(i)$, must be members of U as well. Similarly, if an alternative $i \in F$, then all predecessors (direct and indirect) of alternative i, referred to as $Pred(i)$, must be included in F.

In order to describe the preprocessing procedure, we first need to define the expected net present value for a single alternative $i \in N$ as $eNPV_i = c_i + p_i C\exp(-rd_i)$. It is not possible to easily judge about the non-negative or negative impact of a single alternative i by considering only either $eNPV_i \geq 0$ or $eNPV_i < 0$. For instance, if $eNPV_i \geq 0$ but all members of $Pred(i)$ have negative expected net present value, then inclusion of alternative i and consequently $Pred(i)$ in the schedule may have an overall negative impact on the objective function. Also, if $eNPV_i < 0$ but all members of $Suc(i)$ have positive expected net present value, then inclusion of alternative i along with some alternatives of $Suc(i)$ in the schedule may have in overall a non-negative impact on the objective function. In this confusing situation, in order to determine whether an alternative is favorable or unfavorable, a good idea is to use lower bound and upper bound of impacts of each alternative. We know that if an alternative i and all of its successors are unfavorable while all of its predecessors are favorable, then there is a gap between alternative i and $Pred(i)$. Likewise, if an alternative i and all of its predecessors are favorable while all of its successors are unfavorable, then there is a gap between alternative i and $Suc(i)$. In each of the two foregoing situations, we call alternative i as a head alternative. The union of a head alternative i and $Suc(i)(Pred(i))$ constitutes a set shown by $H_i^{Suc}(H_i^{Pred})$. Assume $UB_{H_i^{Suc}}(LB_{H_i^{Pred}})$ indicates the upper (lower) bound of impact of $H_i^{Suc}(H_i^{Pred})$ on the objective function of a feasible schedule. If we consider es_i, ls_i, ef_i and lf_i as the earliest start time, the latest start time, the earliest finish time and the latest finish time of alternative $i \in N$, where all of them are computed based on the critical path method, we have

$$UB_{H_i^{Suc}} = \sum_{k \in H_i^{Suc}} \left(\exp(-r(ls_k))c_k \prod_{j:ef_j \leq ls_k} (1-p_j) + C\exp(-r(ef_k))p_k \prod_{j \in Pred(k)} (1-p_j) \right) \text{ and}$$

$$LB_{H_i^{Pred}} = \sum_{k \in H_i^{Pred}} \left(\exp(-r(es_k))c_k \prod_{j \in Pred(k)} (1-p_j) + C\exp(-r(lf_k))p_k \prod_{j:ef_j \leq lf_k} (1-p_j) \right).$$

Now, we use these bounds to decide about alternatives as follows. For each alternative $i \in N$, if $UB_{H_i^{Suc}} < 0$, then all alternatives of H_i^{Suc} are candidates to be unfavorable. The relation $UB_{H_i^{Suc}} < 0$ means that alternatives of the set H_i^{Suc} together will have an overall negative impact in each feasible schedule but we cannot consider all of them as unfavorable alternatives. This discrepancy arises because members of set H_i^{Suc} may have predecessors with positive impacts on the objective function while these impacts are not considered in calculation of $UB_{H_i^{Suc}}$. For instance, assume there is an alternative $k \in H_i^{Suc}$ where $k \neq i$ and $LB_{H_k^{Pred}} \geq 0$. In this case, we cannot decide that alternative k is favorable or unfavorable. Similarly, for each alternative $i \in N$, if $LB_{H_i^{Pred}} \geq 0$, then all alternatives of H_i^{Pred} are candidates to be favorable alternatives. We consider the alternatives which have membership of both sets F and U as undecided alternatives and remove them from both sets. Certainly, when we remove an alternative from F (U), we have to remove all of its successors (predecessors) as well. The pseudo-code of the preprocessing is described in Fig. 5.

1. Let $F = U = \emptyset$
2. For i=1 to n do
    3. If $eNPV_i \geq 0$, then
        4. Calculate $LB_{H_i^{Pred}}$.
        5. If $LB_{H_i^{Pred}} \geq 0$, then let $F = F \cup H_i^{Pred}$.
6. End for
7. For i=n down to 1 do
    8. If $eNPV_i < 0$, then
        9. Calculate $UB_{H_i^{Suc}}$.
        10. If $UB_{H_i^{Suc}} < 0$, then let $U = U \cup H_i^{Suc}$.
11. End for
12. For i=1 to n do
    13. If $i \in F \cap U$, then let $F = F \backslash H_i^{Suc}$ and $U = U \backslash H_i^{Pred}$
14. End for

Fig. 5. Pseudo-code of the preprocessing procedure.

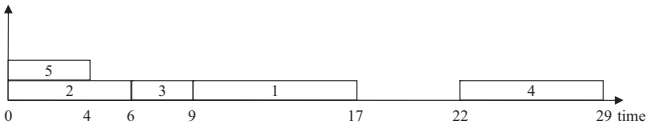

Fig. 6. An optimal schedule including gap.

Table 3
Implementation of the preprocessing procedure.

i	eNPV$_i$	$LB_{H_i^{Pred}}$	$UB_{H_i^{Suc}}$
1	1304.45	−48.71	1355.34
2	−31.00	−31.00	1607.15
3	2082.59	−93.44	1607.19
4	−28.00	−53.78	−0.03
5	1870.38	−78.10	1950.30

Since $UB_{H_i^{Suc}}$ and $LB_{H_i^{Pred}}$ are dependent on the latest start times or the latest finish times of some alternatives, the project deadline affects the effectiveness of the preprocessing procedure. Computational results demonstrate that the preprocessing procedure is more effective when the deadline is tighter. In other words, when the deadline is tighter, the number of undecided alternatives will be decreased. On the other hand, in order to separate favorable and unfavorable alternatives, we must have $\delta > \sum_{i=1}^{n} d_i$. Thus, we set $\delta = 1 + \Sigma_{i=1}^{n} d_i$.

Consider the example project but assume $p_2 = p_4 = 0.00$. The optimal schedule of this new instance with $eNPV = 2065.82$ is depicted in Fig. 6 in which there is a gap between alternatives 1 and 4. In other words, alternative 4 is unfavorable while other ones are favorable. This schedule implies that even if alternatives 1, 2, 3 and 5 fail, alternative 4 should not be executed because it will

surely fail ($p_4=0.00$). Even if $p_4>0$ but $eNPV_4<0$, a risk aversion firm would prefer to not execute alternative 4 because it has a negative impact on its *eNPV*. Of course, if $p_4>0$ but $eNPV_4<0$, it is recommended to a risk-seeking firm to start alternative 4 as late as possible in order to reduce its negative impact.

For each alternative $i\in N$, $eNPV_i$, $LB_{H_i^{Pred}}$ and $UB_{H_i^{Suc}}$ are displayed in Table 3. If we follow steps 1 to 11 of the preprocessing procedure, we find $F=\{1,2,3,5\}$ and $U=\{1,2,3,4,5\}$ but steps 12 to 14 result in $F=\emptyset$ and $U=\{4\}$.

It should be mentioned here that in a special case in which for every alternative $i\in N$, we know $eNPV_i\geq 0$, the preprocessing procedure is not required because all alternatives are favorable. For this case, we have designed a faster version of B&B algorithm that will be described in Section 4.1.1.

4. Branch-and-bound algorithm

In our B&B algorithm, each node of the search tree corresponds to a partial schedule $\boldsymbol{s}$ where set $E(\boldsymbol{s})$ indicates the set of different events in this schedule. Each event $e\in E(\boldsymbol{s})$ corresponds to a time instant at which at least one alternative is started or finished.

4.1. The branching strategy

4.1.1. The branching strategy of F-B&B

Assume all alternatives are favorable. In this case, in the root node, set $E(\boldsymbol{s})$ is initialized as $E(\boldsymbol{s})=\{0\}$. For each node v which corresponds to a partial schedule $\boldsymbol{s}$, we consider a set of eligible alternatives EL_v including alternatives all of whose predecessors are scheduled already. For each alternative $i\in EL_v$, we consider a set of possible start times based upon available time events in $E(\boldsymbol{s})$ and precedence constraints. In other words, regarding to the precedence constraints, branches of node v are created by considering the start time or the finish time of each alternative $i\in EL_v$ equal to each event $e\in E(\boldsymbol{s})$. For this purpose, the possible start times and finish times of each alternative i are calculated as $\{t\in E(\boldsymbol{s})|es_i\leq t\leq ls_i\}$ and $\{t\in E(\boldsymbol{s})|ef_i\leq t\leq lf_i\}$, respectively. Since each alternative is added to a partial schedule when all of its predecessors are already scheduled, in each node, the earliest start times and consequently the earliest finish times of alternatives may be updated while the latest start times and the latest finish times will not be updated. For example, consider the partial schedule depicted in Fig. 7 including alternatives 1 and 5. In the node corresponding to this partial schedule, we have $EL=\{2,4\}$ and $E(\boldsymbol{s})=\{0,4,8\}$. The offspring of this node are depicted in Fig. 8.

Since alternative 1 is the predecessor of alternative 4, alternative 4 can be started only at time instant $t=8$ (Fig. 8(a)). There is not any event in $E(\boldsymbol{s})$ that can be considered as the finish time of alternative 4. Also, there are three events in $E(\boldsymbol{s})$ that can be considered as the start time of alternative 2, i.e., $t=0$, 4 and 8 (Fig. 7(b–d)). In addition, there is only one event in $E(\boldsymbol{s})$ that can be considered as the finish time of alternative 2, i.e., $t=8$ (Fig. 8(e)).

Property 1. The complexity order of F-B&B algorithm is $O((2n)!/2^n)$.

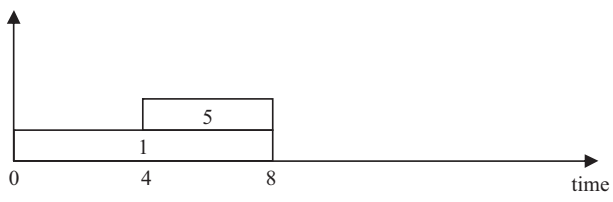

Fig. 7. A Partial schedule.

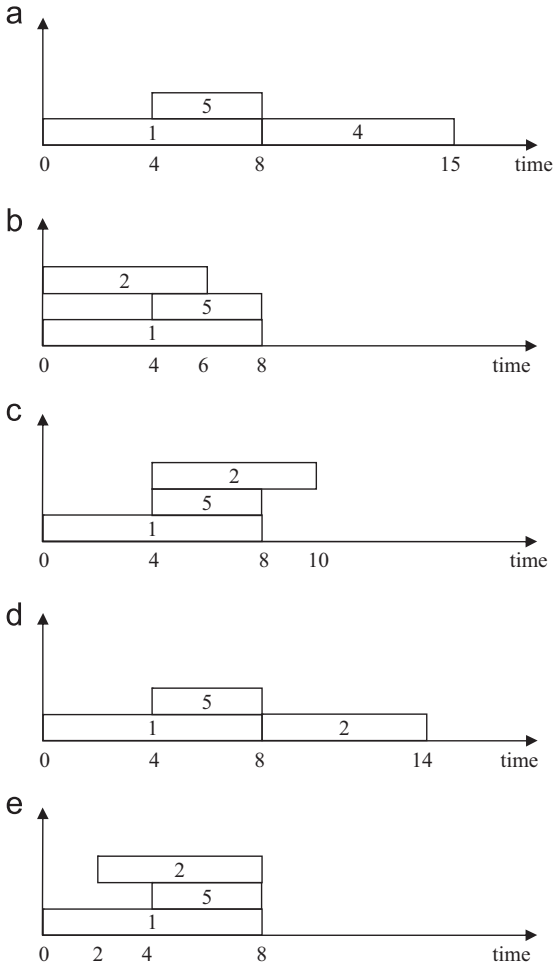

Fig. 8. Illustration of the branching strategy for the F-B&B.

Proof. In F-B&B algorithm, when k ($k<n$) alternatives have been scheduled, there are at most $2k+1$ different start times for $k+1^{th}$ alternative which is selected from $(n-k)$ remaining alternatives and the maximum number of possible schedules is $1(n)\times 3(n-1)\times 5(n-2)\cdots\times(2n-1)(1)=n!\times 1\times 3\times\cdots\times(2n-1)=n!\times((2n)!/2^n\times n!)=(2n)!/2^n$. □

4.1.2. The branching strategy of FB-B&B

If the F-B&B algorithm is applied to an instance of the ATPSP in which there is unfavorable alternatives that are not determined by the preprocessing procedure, the obtained solution is not optimal. In order to find the optimal solution in such instances, we use FB-B&B algorithm in which $E(\boldsymbol{s})$ is initialized as $E(\boldsymbol{s})=\{0,\delta\}$. We also redefine EL_v as the set of alternatives whose all predecessors or all successors are already scheduled. Also, regarding to finish-to-start precedence relations, for each alternative $i\in EL_v$, we consider a set of possible start times based upon available events in $E(\boldsymbol{s})$. In FB-B&B algorithm, both earliest start times and latest start times of alternatives and consequently, the earliest finish times and the latest finish times may be updated in each node.

For example, consider again the node corresponding to the partial schedule depicted in Fig. 7. If FB-B&B is applied, in addition of offspring shown in Fig. 8, two other offspring, shown in Fig. 9, are generated.

Property 2. The complexity order of FB-B&B algorithm is $O((n!)^2 2^n)$.

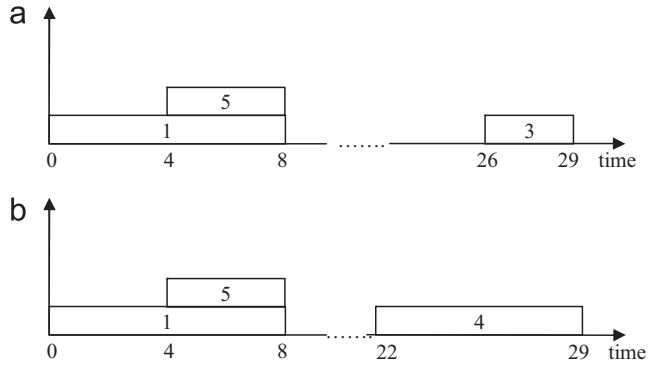

Fig. 9. Illustration of the branching strategy of FB-B&B.

Proof. In FB-B&B algorithm, when k $(k<n)$ alternatives have been scheduled, there are at most $2k+2$ different start times for $k+1^{th}$ alternative which is selected from $(n-k)$ remaining alternatives and the maximum number of possible schedules is $2(n)\times 4(n-1)\times 6(n-2)\cdots\times 2(n)(1)=(n!)^2 2^n$. □

4.2. The bounding strategy

In this section, we develop three dominance rules to fathom unvalued nodes. The first and the second dominance rules prevent duplication of schedules while the third one fathoms low quality schedules. When an alternative is added to a partial schedule, the set of events $E(\boldsymbol{s})$ may be updated by the addition of a new event e'. If it happens, event e is called the parent of event e' and event e' is called the child of event e. For example, consider Fig. 7 as a partial schedule involving alternatives 1 and 5. If the alternative 4 is added to this partial schedule as it was shown in Fig. 8(a), the event $e=8$ is the parent of event $e'=15$. Now consider Fig. 8(e) in which $e'=2$ is the child of $e'=8$. In this figure, if we assume $d_2=8$ and $f_2=8$, then the set of events will not be updated and hence, the only child of event $e=8$ is event $e'=4$.

Dominance rule 1: Consider a schedule $\boldsymbol{s}$ and event $e\in E(\boldsymbol{s})$. If C_e indicates the set of children of e, all members of C_e should be added to $\boldsymbol{s}$ in increasing order of index number of their corresponding alternatives.

Proof. Assume we follow dominance rule 1 and we obtain schedule $\boldsymbol{s}$ as the best found schedule. Also, suppose a better schedule $\boldsymbol{s}'$ $(g(\boldsymbol{s}')>g(\boldsymbol{s}))$ does still exist. If schedule $\boldsymbol{s}'$ is not concurrent, we can easily create concurrent schedule $\boldsymbol{s}''$ with $g(\boldsymbol{s}'')\geq g(\boldsymbol{s}')$ from schedule $\boldsymbol{s}'$ by shifting some alternatives to the right or to the left. On the other hand, if all members of C_e are added to each partial schedule in increasing order of index number of their corresponding alternatives, obviously all concurrent schedules, such as optimal schedule, are investigated. It should be noticed that different permutations of all events occurring in a time instant result in an identical value of the objective function. Thus, if schedule $\boldsymbol{s}''$ is optimal, then $g(\boldsymbol{s}'')=g(\boldsymbol{s})$,otherwise $g(\boldsymbol{s}'')<g(\boldsymbol{s})$ and consequently $g(\boldsymbol{s}')\leq g(\boldsymbol{s})$ that is a contradiction by the first assumption. Also, if schedule $\boldsymbol{s}'$ is concurrent, the contradiction is proved due to this fact that different permutations of all events occurring in a time instant result in an identical value of the objective function. □

For example, consider the partial schedule of Fig. 8(d). Obviously, the first alternative added to this partial schedule has been alternative 1. Thus, the first added event to E has been $e=8$. If we consider event $e=8$ as a parent, its children are $e'=4$ and $e''=14$, corresponding to the alternatives 5 and 2, respectively. There are two different paths in the search tree from the node corresponding to the partial schedule involving only alternative 1 towards the node corresponding to the partial schedule depicted in Fig. 8(d). In the first path, alternative 2 is added before alternative 5 to the partial schedule while in the second path, alternative 5 is added earlier than alternative 2 to the partial schedule. The dominance rule 1 allows only the former path be extended in the search tree and fathoms the latter one.

Dominance rule 2: In each partial schedule $\boldsymbol{s}$, if event e has been created before event e' (which is not necessarily the child of e), all members of C_e must be added to $\boldsymbol{s}$ before any member of $C_{e'}$.

Proof. Dominance rule 2 is proved similarly to dominance rule 1. It suffices to notice that if we consider two different time instants in a solution of ATPSP where each one corresponds to a set of events, different permutations of all of these events result in an identical value of the objective function. □

For example, consider the partial schedule depicted in Fig. 10(a). In order to obtain the complete schedule shown in Fig. 10(b), there are different paths in the search tree. Based upon dominance rule 1, alternative 3 must be added before alternative 4 but, there are three different paths yet. In the first path, alternatives 3, 4 and 5 are added in order of 3, 4 and 5 while in the second and the third paths they are added to the partial schedule in order of 3, 5, 4 and 5, 3, 4, respectively. According to dominance rule 1, alternative 1 is added before alternative 2, hence, event $e=8$ is created before event $e'=6$. Now, on the basis of dominance rule 2, children of e must be created before children of e'. In other words, from the above three mentioned paths, only the first path is investigated and two other ones are fathomed.

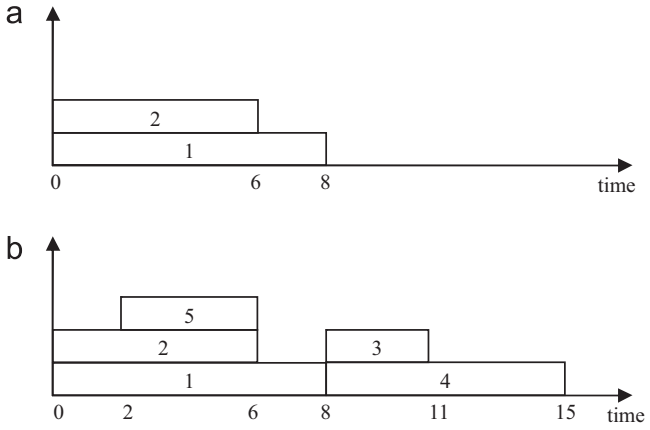

Fig. 10. Illustration of the dominance rule.

For each node v, consider SA_v and RA_v as the set of scheduled alternatives and the set of remaining alternatives of a partial schedule, respectively. Also, UB_v indicates the upper bound of the partial schedule corresponding to node v and is calculated as follows.

$$UB_v=\sum_{i\in SA_v}\left(\exp(-rs_i)c_i\prod_{j\in(SA_v\cap FBA(s_i))}(1-p_j)\prod_{k\in RA_v\&ef_k\leq s_i}(1-p_k)\right)$$

$$+\sum_{i\in RA_v}\left(\exp(-r(ls_i))c_i\prod_{j\in SA_v\&f_j\leq ls_i}(1-p_j)\prod_{k\in RA_v\&ef_k\leq ls_i}(1-p_k)\right)$$

$$+C\sum_{t\in NUF}\left(\exp(-rt)\left(1-\prod_{k\in(SA_v\cap FA(t))}(1-p_k)\right)\right)$$

$$\times \prod_{j \in (SA_v \cap FB(t))} (1-p_j) \prod_{l \in RA_v \& lf_l < t} (1-p_l) \Bigg)$$

$$+C \sum_{i \in SA_v \& f_i \notin NUF} \Bigg(\exp(-r(s_i+d_i))p_i \prod_{j \in (SA_v \cap FB(f_i))} (1-p_j)$$

$$\times \prod_{k \in RA_v \& lf_k < f_i} (1-p_k) \Bigg) + C \sum_{i \in RA_v} \Big(\exp(-r(es_i+d_i))p_i$$

$$\times \prod_{j \in SA_v \& f_j < ef_i} (1-p_j) \prod_{k \in RA_v \& lf_k < ef_i} (1-p_k) \Bigg)$$

In order to have a more efficient upper bound, we update earliest start times, earliest finish times, latest start times and latest finish times in each node of the search tree. If we consider *LB* as the lower bound of the optimal solution (the best found solution so far), initialized by a schedule in which all alternatives are started at their earliest possible start times, the dominance rule 3 is described as follows.

Dominance rule 3: In each node ν of the search tree, if $UB_v \leq LB$, then fathom node *v*.

5. Computational performance

We have performed a series of computational experiments using randomly generated test sets in order to examine the performance of F-B&B and FB-B&B and analyze the effects of different parameter choices.

5.1. Experimental setup

All coding was done in the Visual C++ 6.0 environment and all experiments were run on a PC Pentium IV 3 GHz processor with 1024 MB of internal memory. We have generated a test set using the random network generator RanGen (Demeulemeester et al. [18]), including 60 test instances. We choose $n=6$, 8, 10, 12 and $OS=0.4$, 0.6 and 0.8, where *OS* is the order strength, the number of precedence-related activity pairs divided by the theoretically maximum number of such pairs in the network (Mastor [19]). For each combination of *n* and *OS* we generate five test instances, varying the alternatives' duration, cost and *PTS*: durations and costs are realizations of (independent) discrete uniform random variables on the intervals [1;10] and [−100; −10], respectively, and the *PTS*-values are, unless stated otherwise, with uniform probability chosen from [50%;100%]. We refer to the ratio of project payoff to sum of the alternatives cost, $C/\sum_{i \in N} c_i$ as the payoff-to-cost (*PTC*) index, which is a characteristic of a given project. Unless mentioned otherwise, we set $PTC=5$ and $r=0.05$.

Table 4
Average T_{Total} (in seconds) for F-B&B, FB-B&B and O-B&B.

			N			
			6	8	10	12
OS	0.4	F-B&B	0.003	0.269	16.512	1162.850
		FB-B&B	0.028	2.384	143.909	6267.659
		O-B&B	0.078	27.147	7529.770	–
	0.6	F-B&B	0.000	0.041	0.956	35.441
		FB-B&B	0.003	0.309	7.194	177.750
		O-B&B	0.012	1.794	251.062	–
	0.8	F-B&B	0.000	0.000	0.013	0.422
		FB-B&B	0.003	0.013	0.153	3.053
		O-B&B	0.003	0.035	1.119	53.791

5.2. Detailed comparison

In order to compare the performance of the F-B&B and FB-B&B algorithms with old B&B developed by Ranjbar [14] (referred to as O-B&B), we take the average total CPU run time, shown as T_{Total}, as the comparison criterion. Table 4 shows the average T_{Total} in seconds for F-B&B, FB-B&B and O-B&B for different numbers of alternatives and different values of *OS*. The empty cells of Table 4 indicate that O-B&B could not be finished in 5 hours. It should be noticed that the ATPSP is very hard and numerous feasible solutions are available for each instance of this problem, especially when *OS* is small. When the *OS* is decreased, the number of precedence relations (the size of set *A*) is reduced and consequently, the number of possible start times for each alternative will be increased. Thus, the ATPSP for small values of *OS* and large values of *n* is really an intractable problem. On the other hand, existence of more than 12 alternatives for execution of an R&D project usually does not usually happen in practice.

It should be mentioned that we applied FB-B&B without preprocessing procedure because it has a negative effect on the average T_{Total}. We will elaborate more on the impact of the preprocessing procedure in Section 5.3.2.

For F-B&B and FB-B&B, in average we obtained $T_{Total}=101.38$ seconds and $T_{Total}=550.20$ seconds, respectively. Also, for O-B&B, if we ignore test problems with $n=12$ and $OS=0.4$ or 0.6 which could not be solved in 5 hours, in average we have $T_{Total}=786.48$ seconds. Although F-B&B is the fastest algorithm, it should be noticed that this algorithm could not find the optimal solution for two out of 60 test instances because in the optimal solutions of these two test instances, some alternatives have been unfavorable. For these two test instances, the average percent deviation of the obtained solutions from the corresponding optimal solutions is 1.24%. Of course, if *PTC* is decreased, the number of unfavorable alternatives will be increased and consequently, the average percent deviation between the best solutions found by F-B&B and the optimal solutions will be increased.

Moreover, we run each B&B algorithm for limited CPU times: 1, 10 and 100 seconds. In each case, the average percent deviation from optimal solutions for different values of *n* is reported in Table 5. Table 6 is constructed similar to Table 5 but in which different values of *OS* are considered. Since all of the optimal

Table 5
Average percent deviation from optimal solution for different values of *n* in the limited CPU times.

	Time limit											
	1				10				100			
	n				*n*				*n*			
	6	8	10	12	6	8	10	12	6	8	10	12
F-B&B	0.00	0.00	0.78	2.35	0.00	0.00	0.05	1.94	0.00	0.00	0.00	0.73
FB-B&B	0.00	0.17	2.44	3.35	0.00	0.00	0.59	2.32	0.00	0.00	0.05	1.86
O-B&B	0.00	0.97	3.68	3.98	0.00	0.55	2.75	3.48	0.00	0.00	1.32	3.11

Table 6
Average percent deviation from optimal solution for different values of *OS* in the limited CPU times.

	Time limit								
	1			10			100		
	OS			*OS*			*OS*		
	0.4	0.6	0.8	0.4	0.6	0.8	0.4	0.6	0.8
F-B&B	0.61	0.00	0.00	0.42	0.00	0.00	0.18	0.00	0.00
FB-B&B	0.83	0.00	0.00	0.56	0.00	0.00	0.42	0.00	0.00
O-B&B	1.37	0.00	0.00	1.12	0.00	0.00	0.81	0.00	0.00

solutions were positive, we did not face with the problem of "division by zero". Also, in the case that each of the B&B algorithms could not find any solution in the given CPU time limit, the *eNPV* of the initial solution in which all alternatives are started at their earliest start time is considered.

The results of Tables 5 and 6 indicate that F-B&B has again the best performance in the limited CPU run time.

We know that in the optimal solution there is a gap between favorable and unfavorable alternatives, and the unfavorable alternatives are scheduled on the right side of the gap. On the other hand, the F-B&B algorithm considers all alternatives as favorable and schedules them in left side. Thus, the best obtained solution of the F-B&B algorithm may not be optimal and its deviation from the optimal solution will be increased when the number of unfavorable alternatives is increased.

5.3. Impact of components

5.3.1. Impact of the upper bound

In order to evaluate impact of the developed upper bound, we run both F-B&B and FB-B&B without upper bound for 10 seconds over all test instances. The results show that for $n=8$ and 10, the average percent deviation from optimal solutions increases in both F-B&B and FB-B&B algorithm and consequently, the upper bound has positive impact on the performance of the B&B algorithm. We predict this increment will be observed also for $n>10$.

Consider the difference between the average percent deviation from optimal when upper bound is excluded from the algorithm and the case that the upper bound is included. Based on the results illustrated in Fig. 11, this difference criterion is in average 0.22% for F-B&B algorithm and 0.45% for the FB-B&B algorithm. It implies that the upper bound has more impact on the FB-B&B algorithm rather than the F-B&B algorithm.

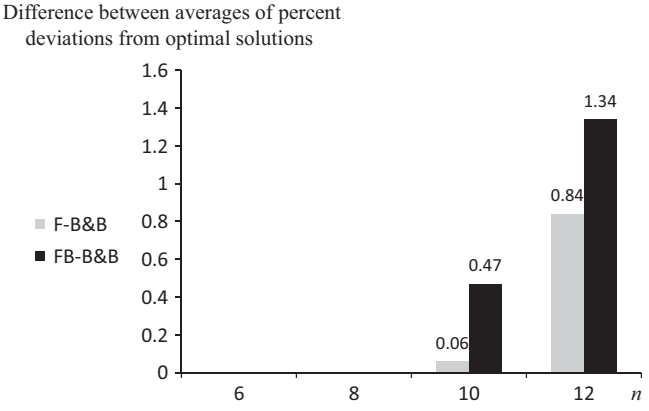

Fig. 11. Impact of the upper.

5.3.2. Impact of the preprocessing procedure

The preprocessing procedure may have negative effect on the performance of FB-B&B if number of favorable and unfavorable alternatives determined by this procedure is not noticeable. For the generated test instances, the preprocessing procedure determines that 2.2% and 0.0% of alternatives belongs to the sets *F* and *U*, respectively, while on the basis of optimal solutions, 94.5% of these alternatives are favorable. In this situation, the preprocessing procedure is not worthwhile because it results in an increase in CPU run time, without helping to solve the problem.

In order to show a case in which the preprocessing procedure has positive effect, we consider $PTC=50$ instead of $PTC=5$. Now, the preprocessing procedure determines that 11.68% and 0.0% of alternatives belong to the sets *F* and *U*, respectively, while based upon optimal solutions, 95.8% of these alternatives are favorable. We run FB-B&B with and without preprocessing procedure for 10 seconds over this new test set. The average percent deviation from optimal solutions for different values of *n* is depicted in Fig. 12. The results show that when the preprocessing procedure is included in the FB-B&B algorithm the average percent deviation from optimal solution is 0.17% while when it is excluded the average percent deviation will be 0.20%.

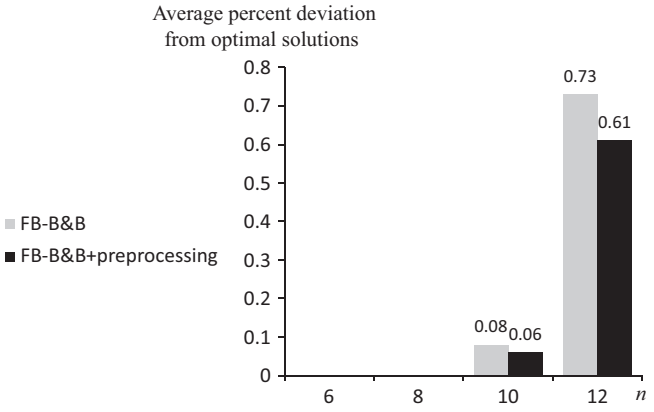

Fig. 12. Impact of the preprocessing.

6. Conclusions

In this paper, we reformulated the ATPSP as a non-linear integer programming model and proved that there is an optimal concurrent schedule for each instance of the ATPSP. Also, we developed a new branch-and-bound algorithm for the problem which relies on the concurrency property. The computational results indicate that the F-B&B algorithm is faster than FB-B&B and O-B&B algorithms but is not able to find optimal solutions in the case there are unfavorable alternatives. Also, for $PTC=5$, the preprocessing procedure was not efficient but becomes more effective as *PTC* increases. In addition, the developed upper bound has positive impact on the performance of the B&B algorithms especially for larger test instances.

Since exact algorithms are not able to solve large scale instances of the ATPSP, as a future research opportunity, we recommend to develop heuristic or metaheuristic algorithms for the problem. We also suggest to develop the setting of this problem to more practical assumptions like stochastic durations for alternatives (see Creemers et al. [6]) or resource constraints.

Table 7
Different situations for $g(\boldsymbol{s}')-g(\boldsymbol{s})$ in case (a).

Case number	Δ_1	Other conditions	$FBA'(s_i')$	NUF'	FA'	FB'	$g(\boldsymbol{s}')-g(\boldsymbol{s})$
a-1)	s_i-lebs_i	–	$FBA(s_i)$	NUF	FA	FB	I
a-2)	f_i-lebf_i	$\nexists j\in N: f_j=lebf_i$	$FBA(s_i)$	NUF	FA	FB	I
a-3)	f_i-lebf_i	$\exists j\in N: f_j=lebf_i,\ f_j\notin NUF$	$FBA(s_i)$	$NUF\cup lebf_i$	$FA'(lebf_i)=\{i,j\}$	$FB'(lebfi)=FB(fi)\backslash\{j\}$	I
a-4)	f_i-lebf_i	$\exists j\in N: f_j=lebf_i,\ f_j\in NUF$	$FBA(s_i)$	NUF	$FA'(lebf_i)=FA(lebf_i)\cup i$	$FB'(lebfi)=FB(fi)\backslash\{j:fj=lebfi\}$	I

Table 8
Different situations for $g(\boldsymbol{s}')-g(\boldsymbol{s})$ in case (b).

Case number	Δ_2	Other conditions	$FBA'(s_i')$	NUF'	FA'	FB'	$g(\boldsymbol{s}')-g(\boldsymbol{s})$
b-1)	$eeas_i-s_i$	$\nexists j\in N: f_j=eeas_i$	$FBA(s_i)$	NUF	FA	FB	II
b-2)	$eeas_i-s_i$	$\exists j\in N: f_j=eeas_i$	$FBA(s_i)\cup FA(eeas_i)$	NUF	FA	FB	II'
b-3)	$eeaf_i-f_i$	$\nexists j\in N: f_j=eeaf_i$	$FBA(s_i)$	NUF	FA	FB	II
b-4)	$eeaf_i-f_i$	$\exists j\in N: f_j=eeaf_i, f_j\notin NUF$	$FBA(s_i)$	$NUF\cup eeaf_i$	$FA'(eeaf_i)=\{i,j\}$	$FB'(eeafi)=FB(eeafi)\backslash\{i\}$	II
b-5)	$eeaf_i-f_i$	$\exists j\in N: f_j=eeaf_i, f_j\in NUF$	$FBA(s_i)$	NUF	$FA'(eeaf_i)=FA(eeaf_i)\cup i$	$FB'(eeafi)=FB(eeafi)\backslash\{i\}$	II

Acknowledgments

This work is supported by Ferdowsi University of Mashhad as a research project with number 6741. Also the authors would like to thank Professor Dr. Roel Leus at KULeuven (Belgium), Professor Dr. Bert De Reyck at University College London (UK) and Professor Dr. Yael Grushka-Cockayne at University of Virginia (USA) for their valuable comments and suggestions.

Appendix A

Theorem 1. For each instance of the ATPSP, there is a concurrent optimal schedule.

Proof. Consider an optimal solution $\boldsymbol{s}=(s_1,..,s_i,..,s_n)$ of the ATPSP in which alternative $i\in N$ is not concurrent. We show that the solution $\boldsymbol{s}'=(s_1',..,s_i',..,s_n')$ with $g(\boldsymbol{s}')\geq g(\boldsymbol{s})$ can be found in which $\forall j\in N, j\neq i: s_j'=s_j$ and alternative i is concurrent. For this purpose, we define the conditional expected net present value for each alternative $i\in N$ as $CeNPV_i=c_i+Cp_i\exp(-rd_i)\prod_{j\in FBSF_i}(1-p_j)$ in which $FBSF_i=\{j\in N|s_i<f_j<f_i\}$ indicates the set of alternatives that are finished between start time and finish time of alternative i. The term of *conditional* is used because we have assumed all alternatives $j\in FBSF_i$ have been failed. Consider $lebs_i$, $eeas_i$, $lebf_i$ and $eeaf_i$ as the latest event (time instant) before the start time of alternative i, the earliest event after the start time of alternative i, the latest event before the finish time of alternative i and the earliest event after the finish time of alternative i, respectively. In this proof, we consider two general cases: (a) $CeNPV_i\geq 0$ and (b) $CeNPV_i<0$. In case (a), we show that solution $\boldsymbol{s}'$ is created by shifting alternative i to the left while in case (b) the solution $\boldsymbol{s}'$ is created by shifting alternative i to the right. In each case, due to the changes created in sets $FBA(s_i)$, NUF, $FA(t)$ and $FB(t)$, different conditions are formed for calculation of $g(\boldsymbol{s}')$. We use notations $FBA'(s_i')$, NUF', $FA'(t)$, $FB'(t)$ to show the sets contributed in calculation of $g(\boldsymbol{s}')$.

Case (a) $CeNPV_i\geq 0$

In this case, we construct solution $\boldsymbol{s}'$ from solution $\boldsymbol{s}$ using shifting alternative i to the left by the amount of $\Delta_1=\min\{s_i-lebs_i,\ f_i-lebf_i\}$. Four different situations in case (a) are listed in Table 7. In the following, we interpret, for instance, the case $(a$-3$)$ of Table 7. The notation $FBA(s_i)$ in column $FBA'(s_i')$ implies that the set of alternatives finished at or before alternative i in schedule $\boldsymbol{s}'$ is identical to $FBA(s_i)$. Also, $NUF'=NUF\cup\{lebf_i\}$ means set of non-unique finish times in schedule $\boldsymbol{s}'$ consists of time instant $t=lebf_i$ and set of non-unique finish times in schedule $\boldsymbol{s}$. In addition, $FA'(lebf_i)=\{i,j\}$ means that the set of alternatives which are finished at $t=lebf_i$, are $\{i,j\}$ while set FA' for each of the other time instants $t\in NUF'$ is identical to set FA for the same time instant. Moreover, $FB'(lebf_i)=FB(f_i)\backslash\{j\}$ denotes the set of alternatives finished before $t=lebf_i$ in schedule $\boldsymbol{s}'$ equals to the set of alternatives finished before $t=f_i$ in schedule $\boldsymbol{s}$ except alternative j where $f_j=lebf_i$. Elementary algebra then shows that the value of $g(\boldsymbol{s}')-g(\boldsymbol{s})$ results an identical value, shown as (I), in all four cases $(a$-1$)$ to $(a$-4$)$.

$$I=g(\boldsymbol{s}')-g(\boldsymbol{s})=\exp(-rs_i)(\exp(r\Delta_1)-1)\left(\prod_{j\in FBA(s_i)}(1-p_j)\right)\times\left(c_i+Cp_i\exp(-rd_i)\prod_{j\in FBSF_i}(1-p_j)\right)$$

Since $\exp(-rs_i)\geq 0$, $\exp(r\Delta_1)-1\geq 0$, $\prod_{j\in FBA(s_i)}(1-p_j)\geq 0$ and we have assumed $c_i+Cp_i\ \exp(-rd_i)\prod_{j\in FBSF_i}(1-p_j)=CeNPV_i\geq 0$, thus $I\geq 0$.

Case (b) $CeNPV_i<0$

In this case, we construct solution $\boldsymbol{s}'$ from solution $\boldsymbol{s}$ using shifting alternative i to the right by the amount of $\Delta_2=\min\{eeas_i-s_i,\ eeaf_i-f_i\}$. Similar to case (a), we have five different situations in case (b), shown by $(b$-1$)$ to $(b$-5$)$ and listed in Table 8. In all cases except case $(b$-2$)$, elementary algebra shows that the value of $g(\boldsymbol{s}')-g(\boldsymbol{s})$ results an identical value, shown as (II).

$$II=g(\boldsymbol{s}')-g(\boldsymbol{s})=\exp(-rs_i)(\exp(-r\Delta_2)-1)\left(\prod_{j\in FBA(s_i)}(1-p_j)\right)\times\left(c_i+Cp_i\exp(-rd_i)\prod_{j\in FBSF_i}(1-p_j)\right)$$

Since $\exp(-rs_i)\geq 0$, $\exp(-r\Delta_1)-1\leq 0$, $\prod_{j\in FBA(s_i)}(1-p_j)\geq 0$ and we have assume $c_i+Cp_i\exp(-rd_i)\prod_{j\in FBSF_i}(1-p_j)=CeNPV_i\leq 0$, thus $II\geq 0$. For the case $(b$-2$)$, the value of $g(\boldsymbol{s}')-g(\boldsymbol{s})$ is shown as II'.

$$II'=g(\boldsymbol{s}')-g(\boldsymbol{s})=\exp(-rs_i)\left(\prod_{j\in FBA(s_i)}(1-p_j)\right)$$

$$\times\left(c_i\left(\exp(-r\Delta_2)\prod_{j\in FA(eeas_i)}(1-p_j)-1\right)\right.$$

$$\left.+Cp_i\exp(-rd_i)(\exp(-r\Delta_2)-1)\prod_{j\in FBSF_i}(1-p_j)\right)$$

Since, $\prod_{j\in FA(eeas_i)}(1-p_j)\le 1$ and consequently $\exp(-r\Delta_2)\prod_{j\in FA(eeas_i)}(1-p_j)-1\le\exp(-r\Delta_2)-1$, it is concluded that $II'\ge II\ge 0$. □

References

[1] Pich MT, Loch CH, de Meyer A. On uncertainty, ambiguity, and complexity in project management. Management Science 2002;48(8):1008–23.

[2] Sommer SC, Loch CH. Selectionism and learning in projects with complexity and unforeseeable uncertainty. Management Science 2004;50(10):1334–47.

[3] Drug Development Process: Stages of Drug Development.(2008). Retrieved May 24, 2012, from 〈http://pacificbiolabs.com/drug_stages.asp〉.

[4] Cohen MA, Eliashberg J, Ho TH. New product development: the performance and time-to-market tradeoff. Management Science 1996;42(2):17–186.

[5] Hendricks KB, Singhal VR. The effect of product introduction delays on operating performance. Management Science 2008;54(5):878–92.

[6] Creemers S, De Reyck B, Leus, RR.&D project planning with multiple trials in uncertain environments. In: IEEE international conference on industrial engineering and engineering management; 2009. p. 1–4: 325–329.

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

[8] Granot D, Zuckerman D. Optimal sequencing and resource allocation in research and development projects. Management Science 1991;37(2): 140–56.

[9] Ding M, Eliashberg J. Structuring the new product development pipeline. Management Science 2002;48(3):343–63.

[10] Loch CH, De Meyer A, Pich MT. Managing the unknown: a new approach to managing high uncertainty and risk in projects. London: J. Wiley and Sons; 2006.

[11] Sobel MJ, Szmerekovsky JG, Tilson V. Scheduling projects with stochastic activity duration to maximize expected net present value. European Journal of Operational Research 2009;198(3) 697–670.

[12] De Reyck B, Grushka-Cockayne Y, Leus R. A new challenge in project scheduling: the incorporation of alternative failures. Tijdschrift voor Economie en Management 2007;LII(3):411–34.

[13] De Reyck B, Leus R. R.&D-project scheduling when activities may fail. IIE Transactions 2008;40(4):367–84.

[14] Ranjbar M A branch-and-bound algorithm for scheduling of new product development projects. Technical report, faculty of engineering, Ferdowsi University of Mashhad, 2010.

[15] Vanhoucke M. An efficient hybrid search procedure for various optimization problems. Lecture Notes in Computer Science 2006;3906:272–83.

[16] Ford DN, Sobek DK. Adapting real options to new product development by modeling the second Toyota paradox. IEEE Transactions on Engineering Management 2005;52(2):175–85.

[17] Price HW. Least-cost testing sequence. Journal of Industrial Engineering 1959:278–9 July–August:.

[18] Demeulemeester E, Vanhoucke M, Herroelen WA. Random generator for activity-on-the-node networks. Journal of Scheduling 2003;6:13–34.

[19] Mastor AA. An experimental investigation and comparative evaluation of production line balancing techniques. Management Science 1970;16(11): 728–46.