Scientia Iranica E (2016) 23(1), 374–386

SCIENTIA IRANICA

Sharif University of Technology
Scientia Iranica
Transactions E: Industrial Engineering
www.scientiairanica.com

Population-based metaheuristics for R&D project scheduling problems under activity failure risk

M. Ranjbar*, H. Validi and R. Fakhimi

Department of Industrial Engineering, Faculty of Engineering, Ferdowsi University of Mashhad, Mashhad, Iran.

Received 20 May 2014; received in revised form 10 January 2015; accepted 15 June 2015

KEYWORDS
Project scheduling;
Risk;
Scatter search;
Genetic algorithm.

Abstract. In this paper, we study scheduling of R&D projects in which activities may to be failed due to the technological risks. We consider two introduced problems in the literature referred to as R&D Project Scheduling Problem (RDPSP) and Alternative Technologies Project Scheduling Problem (ATPSP). In both problems, the goal is maximization of the expected net present value of activities where activities are precedence related and each of them is accompanied with a cost, a duration, and a probability of technical success. In RDPSP, a project payoff is obtained if all activities are succeeded, while in ATPSP, if one of activities is implemented successfully, the project payoff is attained. We construct a solution representation for each of these problems and construct two population-based metaheuristics including scatter search algorithm and genetic algorithm as solution approaches. Computational experiments indicate scatter search outperforms genetic algorithm and also available exact solution algorithms.

1. Introduction

In a lot of industries, projects are subject to considerable uncertainty due to various causes. Factors influencing the completion date of a project include activities that are required but were not identified beforehand, activities taking longer than expected, activities that need to be redone, resources being unavailable when required, late deliveries, etc. In Research and Development (R&D) projects, activities may also fail altogether, for instance because the new technology under study does not perform as anticipated or because a toxicity test is not passed (in case of drug development). This risk is often referred to as technical risk. We consider the main sources of uncertainty in R&D projects, namely, the possibility of activity failure.

De Reyck et al. [1] introduced the modular R&D project scheduling as an R&D project consisting of several modules. In this problem, each module contains one or more activities that pursue a homogeneous target, for instance, representing repeated trials or technological alternatives. Each activity has a cost, a duration, and a Probability of Technical Success (PTS). It should be mentioned that in some references, such as [2], the term "alternative" has been used instead of "activity" in the ATPSP, but we use "activity" in this research. A module is successful when at least one of its included activities succeeds. Successful completion of the whole project requires successful completion of all the modules; project success equates with receiving a project payoff (cash inflow). In their model, they made a number of simplifying assumptions, including unlimited resources and no explicit consideration of the uncertainty in activity durations or project cash flows.

*. *Corresponding author. Tel.: +98 51 38805092*
E-mail addresses: m_ranjbar@um.ac.ir (M. Ranjbar); hamidreza.validi2@gmail.com (H. Validi); fakhimi.ramin@gmail.com (R. Fakhimi)

In this paper, we consider two problems: the RDPSP and the ATPSP. These two problems were introduced by De Reyck and Leus [3] and each of them is a special case of the modular R&D project scheduling problem. The goal is maximization of the expected Net Present Value (*eNPV*) in both the aforementioned problems. The RDPSP is a single-activity-module project scheduling problem in which each module consists of only one activity and successful completion of the whole project requires successful completion of all the activities. The ATPSP is a single-module project scheduling problem in which the project consists of one module and technical success of an activity leads to obtaining the project payoff.

The literature on project scheduling is vast and interested readers are referred to recent literature surveys in this field (e.g. [4,5]). In this study, we focus only on R&D project scheduling problems. The main topic of interest in the literature surveys on the scheduling under uncertainty is duration uncertainty, sometimes complemented with uncertain resource availabilities [6-8]. In this paper, we incorporate the concepts of activity success or failure into the scheduling decisions. A branch-abound (B&B) algorithm was developed for the RDPSP by De Reyck and Leus [3]. A similar model is tackled by [9,10] in which the scheduling of failure-prone new-product-development testing tasks is studied when non-sequential testing is admitted. The concept of modular projects is hinted at [1], but the authors do not develop any solution procedure. This research work has been continued by Creemers et al. [11] in which modular projects by considering the impact of activity duration variability on the project's value are studied. Also, Ranjbar and Davari [2] develops a B&B algorithm for the ATPSP. In all of the foregoing references, the resource constraint has not been taken into account, but recently, Coolen et al. [12] considered scheduling modular projects on a bottleneck resource. They described various policy classes, established the relations among them, and developed exact algorithms to optimize two different classes.

Closely related to the model developed in this paper is the work on sequential testing, in which a series of tests is to be performed to diagnose a system (i.e., to know its state, which usually is either 'working' or 'failing'). A solution in this setting is an inspection strategy, which is specified on the basis of the state of the already inspected components. Each component is to be inspected next, or halts if it is able to recognize the correct state of the system. Reviews of this body of literature can be found in [13,14]. The main differences with our scheduling problem is that the inspections will continue as long as the state of the system is not known, whereas we allow the project to be aborted preliminarily if this is better for the project's value.

One of the most important weaknesses in the previous research works on R&D project scheduling problems is that the developed solution approaches are applicable only for small-scale test instances. Thus, developing solution approaches which can solve large-scale and practical problems is really a research gap. To overcome this crisis, in this research, we construct efficient solution approaches.

The contributions of this article are twofold:

1. We develop representations of two solutions for the RDPSP and ATPSP, respectively;
2. We develop two population-based metaheuristic algorithms including two different combining methods and customized repairing and local search procedures to solve both problems.

The remainder of this paper is organized as follows. Section 2 contains formal definition and formulation of the RDPSP and ATPSP. Our scatter search algorithm along its components is described in Section 3 and computational performance is evaluated in Section 4. Finally, Section 5 contains a summary, some conclusions, and ideas for further research.

2. Formulation of the problems

The objective of RDPSP and ATPSP is to maximize *eNPV* of the project by constructing a project schedule specifying when to execute each activity. In the RDPSP (ATPSP), the final project payoff is only achieved when all activities are (one activity is) successful, and the project is terminated as soon as an activity fails (succeeds). Activity success or failure is revealed at the end of each activity. Consequently, in the RDPSP (ATPSP), each activity will start only if all the activities scheduled to finish earlier have a positive (negative) outcome and hence, in the objective function, the activity cash flows are weighted by the probability of joint success (failure) 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 1.

Without loss of generality, we assume activity 0 to be a dummy activity representing project initiation and be the predecessor of all other activities with $c_0 = d_0 = 0$ and $p_0 = 1$. Activity $n+1$ is also a dummy activity representing project completion and is a successor of all other activities. We assume that $d_i > 0$ for all non-dummy activities. A deadline δ is imposed on project completion because we require that $s_{n+1} \leq \delta$. This deadline is needed because optimization will try to push the start times of activities to infinity if the optimal *eNPV* of a particular problem instance is negative.

A schedule $\mathbf{s}$ is feasible if it respects the constraints imposed by A. In the RDPSP (ATPSP), the

Table 1. Definitions of parameters.

Parameter	Definition
N	Set of activities, $N = \{0, 1, ..., n+1\}$
c_i	Cost of activity $i \in N$, a non-positive integer incurred at the start of the activity
C	End-of-project positive payoff, integer
d_i	Duration of activity $i \in N$, a positive integer
p_i	PTS of activity $i \in N$
r	Continuous discount rate
A	(Strict) partial order on N, an irreflexive and transitive relation imposing the constraints $s_i + d_i \leq s_j$ for all $(i, j) \in A$
δ	The project deadline

objective is to maximize the project's *eNPV*, so the cost of each activity is weighted by the probability of joint success (failure) of the already finished activities. We represent each schedule by a vector of start times $\mathbf{s} = (s_0, s_1, ..., s_{n+1})$ where s_i is a non-negative integer and indicates the start time of activity i. We also consider $\mathbf{f} = (f_0, f_1, ..., f_{n+1})$ as the vector of finish times, where $f_i = s_i + d_i$, indicates the finish time of activity i.

The RDPSP can now be formulated as the following non-linear integer programming model:

$$\max g_1(\mathbf{s}) = \sum_{i=1}^{n} \left(c_i \exp(-rs_i) \prod\nolimits_{j \in FBA(s_i)} p_j \right) + C \exp(-rs_{n+1}) \prod\nolimits_{j=1}^{n} p_j,$$

subject to:

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

$$s_{n+1} \leq \delta,$$

$$s_i \in \mathbb{Z}^+ \qquad \forall i \in N.$$

In the objective function $g_1(\mathbf{s})$ which indicates *eNPV* of schedule $\mathbf{s}$, $FBA(s_i)$ indicates the set of activities which are finished before or at the start time of activity i. $g_1(\mathbf{s})$ includes two main terms in which exp(.) shows exponential function. The first term indicates the expected value of costs while the second term shows the expected value of revenue. The first constraint of the model indicates the finish-to-start precedence relations among activities imposed by set A. The second constraint implies that dummy end activity $n+1$ must be finished before or at the given deadline. The last constraint shows that all start times are non-negative integers ($\mathbb{Z}^+$ indicates the set of non-negative integers).

The formulation of ATPSP is similar to that of RDPSP, but instead of $g_1(\mathbf{s})$, we use $g_2(\mathbf{s})$, where:

$$g_2(\mathbf{s}) = \sum_{i=1}^{n} \left(\exp(-rs_i) c_i \prod_{j \in FBA(s_i)} (1 - p_j) \right) + C \sum_{t \in \text{NUF}} \left(\exp(-rt) \left(1 - \prod_{k \in FA(t)} (1 - p_k) \right) \prod_{j \in FB(t)} (1 - p_j) \right) + C \sum_{i: f_i \notin \text{NUF}} \left(\exp(-r(s_i + d_i)) p_i \prod_{j \in FB(s_i + d_i)} (1 - p_j) \right).$$

In $g_2(\mathbf{s})$, $FB(f_i)$ shows the set of activities which are finished before finish time of activity i. In addition, NUF shows the set of non-unique finish times such that every $t \in$ NUF is equivalent to the finish time of at least two activities. The set of activities, which are finished at $t \in$ NUF, are shown by $FA(t)$. In addition, $FB(t)$ shows the set of activities which are finished before time instant t. $g_2(\mathbf{s})$ includes three main terms in which the first term represents the expected costs in which $\prod_{j \in FBA(s_i)}(1 - p_j)$ indicates the probability of joint failure of all activities finished before or at the start time of activity i. We assume that if in a schedule for two activities, $j \in N$, we have $f_j = s_i$, then activity i will start if activity j fails. The summation of two other terms in $g_2(\mathbf{s})$ indicates the expected revenue. The second term displays the expected revenue for the activities 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 activities finished at time instant t. Moreover, $\prod_{j \in FB(t)}(1 - p_j)$ indicates the probability of joint failure of all activities finished prior to the time instant t. Finally, the last term specifies the expected revenue for the activities 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 activities finished prior to the completion time of activity i.

3. Schedule representation

Our heuristic algorithms use a schedule representation to encode a schedule. Although there are various approaches for schedule representation of the classic resource-constrained project scheduling problem (see [15]), there is not any solution representation for RDPSP and ATPSP in the literature. We develop two schedule representations for the RDPSP and ATPSP based on the solution approaches made by [2,3], respectively.

3.1. Schedule representation for the RDPSP

Based upon solution of [3], RDPSP is solved in two phases. In the first phase, a feasible extension E of A, which updates sets $FBA(s_i)$ for some $i \in N$, is produced. Then, function $g_1(.)$ is optimized subject to new precedence constraints, which constitutes the second phase. If all feasible extensions of A will be implicitly or explicitly enumerated, finding an optimal schedule for RDPSP is guaranteed. The second phase (optimization for the given new precedence constraints) amounts to project scheduling with *eNPV* objective without resource constraints (see [16]). We consider the case in which all activity cash flows during the development phase are negative, which is typical for R&D projects. 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 start times of its successors in E. Depending on whether the corresponding *eNPV* is positive or negative, we set $s_0 = 0$ or $s_{n+1} = \delta$, respectively. Based upon this solution approach, we represent each schedule of the RDPSP by a set of pairs $\{(i,j) : \forall (i,j) \notin A\}$. In other words, we consider pairs of activities for which there is not any directed path between them in the project network. For each $(i,j) \notin A$, we consider three different scenarios: $(i \rightarrow j), (i \leftarrow j)$, and $(i||j)$, where $(i \rightarrow j)$ implies $s_j \geq s_i + d_i$ and $(i||j)$ implies jointly relations $s_i + d_i > s_j$ and $s_j + d_j > s_i$. In other words, relation $(i \rightarrow j)$ indicates that activity j can start after the end of activity i due to the information flow from i to j. Also, relation $(i||j)$ designates that activities i and j will overlap in execution. Each schedule representation is feasible *iff* there is no cycle in the corresponding project network. When a feasible schedule representation is generated, we schedule all activities based on their latest possible start times, calculated based on the Critical Path Method (CPM). Now, if there is a pair $(i,j) \notin A$ for which we have $(i||j)$ in our schedule representation, but we know $s_j \geq s_i + d_i$ on the basis of start times values obtained from the Latest Start Schedule (LSS), we change $(i||j)$ into $(i \rightarrow j)$.

As an example, we consider a project with 11 activities as represented in Figure 1.

Other project data are given in Table 2. In this example, we assume a discount rate of 5% per month and the project deadline is 67 months. Consider the following feasible schedule representation:

$$\{(1 \rightarrow 2), (1||4), (1||5), (1||6), (1||8), (2||3),$$
$$(2 \leftarrow 5), (2||6), (2||8), (2||9), (3||4), (3||5), (3||6),$$
$$(3||7), (3||8), (4||5), (4||6), (4||8), (4||9), (5||6),$$
$$(5||7), (5||9), (6||7), (6||9), (7||8), (7||9), (8||9)\}.$$

Table 2. Project data.

Activity	Cost ($)	PTS	Duration (months)
0	0	1.000	0
1	-5	0.811	11
2	-34	0.833	12
3	-38	0.915	4
4	-2	0.958	1
5	-8	0.857	9
6	-40	0.951	2
7	-1	0.893	9
8	-42	0.863	8
9	-15	0.972	11
10	802	1.000	0

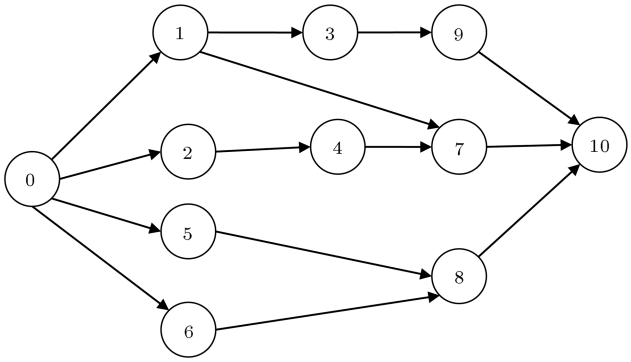

Figure 1. Network of the example project.

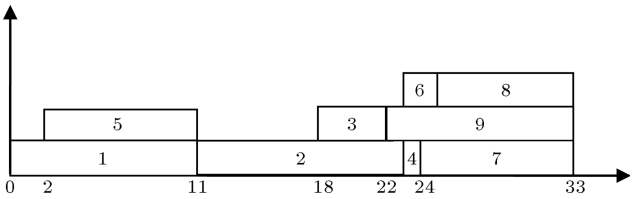

Figure 2. Optimal schedule of the RDPSP.

The corresponding LSS of this representation is presented in Figure 2, which is also optimal schedule with *eNPV* of $11.37. Using information obtained from the LSS, this representation will be changed as follows:

$$\{(1 \rightarrow 2), (1 \rightarrow 4), (1||5), (1 \rightarrow 6), (1 \rightarrow 8), (2||3),$$
$$(2 \leftarrow 5), (2 \rightarrow 6), (2 \rightarrow 8), (2||9), (3 \rightarrow 4), (3 \leftarrow 5),$$
$$(3 \rightarrow 6), (3 \rightarrow 7), (3 \rightarrow 8), (4 \leftarrow 5), (4||6), (4 \rightarrow 8),$$
$$(4||9), (5 \rightarrow 6), (5 \rightarrow 7), (5 \rightarrow 9), (6||7), (6||9),$$
$$(7||8), (7||9), (8||9)\}.$$

This change implies that the relation between solution space and solution representation space is not one to one, because several solution representations may correspond to an identical schedule.

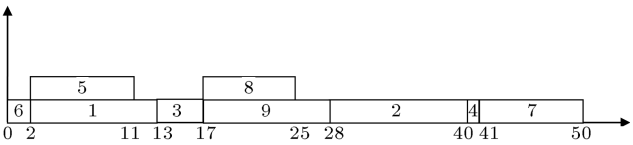

Figure 3. Optimal schedule of the ATPSP.

3.2. Schedule representation for the ATPSP

In the ATPSP, we use a direct representation and show each schedule by its vector of start time $\mathbf{s} = (s_0, s_1, ..., s_{n+1})$. Ranjbar and Davari [2] define the concurrency property and prove that there is an optimal solution for each instance of the ATPSP which is concurrent. Based on this property, the start time or finish time of each activity, which is defined as an event, is concurrent with the start time or finish time of at least another activity. Thus, our solution representation for the ATPSP is subject to the concurrency property. Figure 3 presents the optimal schedule of the ATPSP with *eNPV* of \$671.77 based on the example project depicted in Figure 1. We represent this schedule as $\mathbf{s} = (0, 2, 28, 13, 40, 2, 0, 41, 17, 17, 50)$.

4. Solution approaches

In this section, we develop two metaheuristics including scatter search and genetic algorithm. These two algorithms are both population-based and we have considered some similar or common operators for them, but their overall structures are different.

4.1. A scatter search algorithm

Scatter Search (SS) was developed by Glover and Laguna [17] as a heuristic algorithm for integer programming models. In this algorithm, solutions are generated randomly such that they cover most parts of the solution space. Similar to tabu search algorithm, SS has diversification and intensification procedures to enhance its efficiency [18]. For a brief and general introduction to SS, we refer the readers to [19]. Scatter search algorithm has been used in previous research on scheduling problems, such as [20-23].

Figure 4 shows the main structure of our SS algorithm as a flowchart for both the RDPSP and ATPSP in which some operators have different procedures for each problem. In the first step, an initial population P containing $|P|$ solutions is generated using the *initial population generation method*, described in Section 4.1. If the termination criterion, a given time limit, is met, the SS algorithm stops; otherwise, it continues. In the second step, the reference set *RefSet*, including b elite solutions of P, is constructed using the *reference set building method*, described in Section 4.2. The solutions of *RefSet* are called reference solutions. Next, the *NewSubsets*, each of them containing two reference solutions, are generated.

In this stage, the best solution of the current

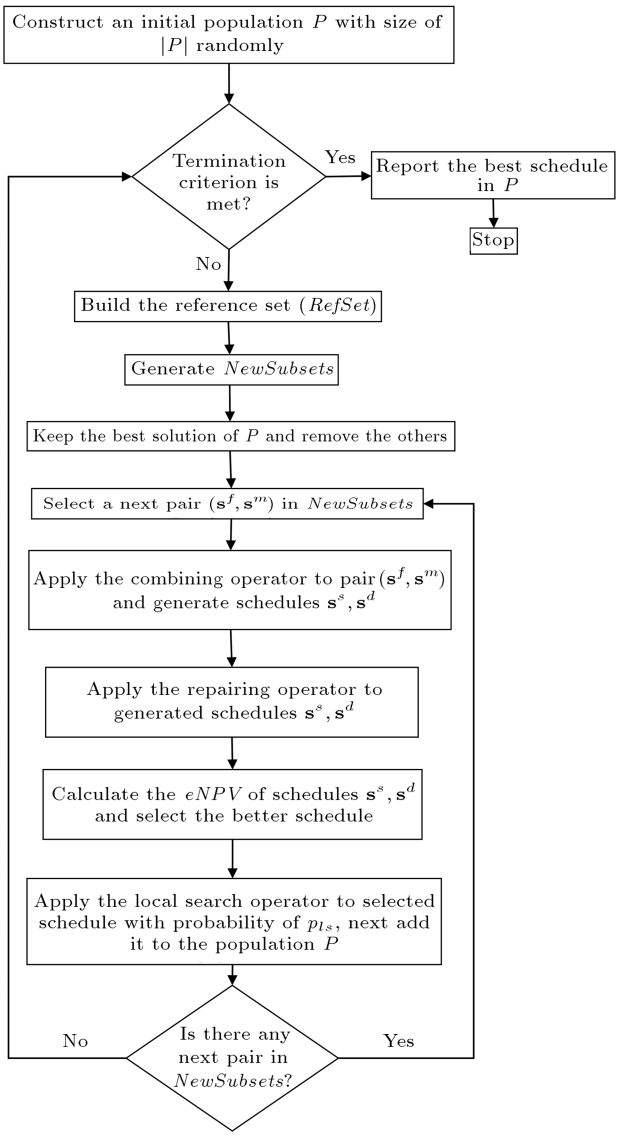

Figure 4. Flowchart of the SS algorithm.

population is kept and the other solutions are removed. Subsequently, two solutions of each subset, called parents solutions and shown by $(\mathbf{s}^f, \mathbf{s}^m)$, are selected and the combining operator, described in Section 4.3, is applied to them. We consider $\mathbf{s}^f$ and $\mathbf{s}^m$ as the father and mother solutions, respectively. Since it is possible that generated schedules $\mathbf{s}^s$ and $\mathbf{s}^d$ are infeasible, we apply the repairing operator, described in Section 4.5, to them. Then, the objective functions of schedules $\mathbf{s}^s$, and $\mathbf{s}^d$ are evaluated and the better schedule is selected to be improved by the local search operator, described in Section 4.6, with probability of p_{ls}. The selected schedule will be added to the population P after improvement by the local search operator. After selection of all pairs $(\mathbf{s}^f, \mathbf{s}^m)$ in *NewSubsets*, the size of population P will be $2|P|$. In this stage, population P will be updated by removing $|P|$ worse schedules from P.

1. Let $EA(\mathbf{s}) = \{i | i \in N, Pred(i) = \varnothing\}$ and $SA(\mathbf{s}) = \varnothing$
2. **while** $EA(\mathbf{s}) \neq \varnothing$ **do**
3. Select activity $j \in EA(\mathbf{s})$ randomly
4. Calculate $ES_j(\mathbf{s})$, select $t \in ES_j(\mathbf{s})$ randomly and let $s_j = t$
5. Update $EA(\mathbf{s})$ and $SA(\mathbf{s})$
6. **end while**

Figure 5. Pseudo-code of the initial population generation method for the ATPSP.

4.1.1. Initial population generation method

Each member of the initial population P in the RDPSP is generated randomly. For this purpose and for each pair of activities $(i, j) \notin A$, we select one of the three cases $(i \rightarrow j), (i \leftarrow j)$, or $(i||j)$, randomly. Since the generated project network may include cycle and be infeasible, we use the repairing procedure expressed in Section 4.4.

Also, each member in the initial population of ATPSP is a random solution and is created by the method, illustrated in Figure 5, by pseudo-code. In this approach, activities are added one by one to a partial schedule $\mathbf{s}$. $EA(\mathbf{s})$ indicates the set of eligible activities, initialized by all activities which do not have any predecessor. Also, $SA(\mathbf{s})$ represents the set of scheduled activities initialized as an empty set.

After the initialization step, we follow an iterative while-loop to construct a random solution. In each iteration, an individual activity is added to the partial schedule $\mathbf{s}$. At step 3, one of the eligible activities, shown by j, is selected randomly. Next, eligible start times for activity j in schedule s, noted by $ES_j(\mathbf{s})$, are calculated based upon concurrency property. Next, one of the eligible events for the start time of activity j is chosen, randomly, and activity j starts in that time. It should be noticed that for each activity j which does not have any predecessor (successor), $t = 0 (t = \delta - d_j)$ is always a valid start time. However, in the next step, sets $EA(\mathbf{s})$ and $SA(\mathbf{s})$ are updated. In order to update set $EA(\mathbf{s})$, activity j should be removed from $EA(\mathbf{s})$ and all activities $i \in Suc(j)$ for which $Pred(i) \subseteq SA(\mathbf{s})$ should be added to $EA(\mathbf{s})$, where $Suc(j)$ and $Pred(j)$ indicate the set of successors and predecessors of activity j, respectively. Also, set $SA(\mathbf{s})$ is updated simply as $SA(\mathbf{s}) = SA(\mathbf{s}) \cup j$.

4.1.2. Reference set building and subset generation methods

The reference set, *RefSet*, is a collection of both high quality solutions and diverse solutions that are used to generate new solutions by way of applying the combining, repairing, and local search operators. We construct *RefSet* based on the method applied for construction of $RefSet_1$ of scatter search in [22]. We select the solution with the smallest objective function, shown as $\mathbf{s}_1$, as the first member of *RefSet* and remove it from P. The next best solution $\mathbf{s}$ in P is chosen and added to *RefSet* only if $D_{\min}(\mathbf{s}) \geq th\_dist$, where $D_{\min}(\mathbf{s})$ is the minimum of the distances of solution $\mathbf{s}$ to the all solutions currently in *RefSet*. Also, *th_dist* indicates a threshold distance. In the RDPSP, the difference between two solutions equals the number of different ordered activities $(i, j) \notin A$ in two solutions divided by the number of unordered pairs of activities. Also, in the ATPSP, the difference between two solutions equals the number of different start times for identical non-dummy activities in two solutions divided by the number of non-dummy activities. Thus, the difference between every two solutions changes in range [0,1]. This process is repeated until b members are chosen for *RefSet*. Whenever no qualified solution can be found in the population, the *RefSet* is completed with random solutions generated based on the *initial population generation method*. For these members of *RefSet*, the condition of minimum threshold distance is ignored.

In the next step, *NewSubsets* are generated consisting of all pairs of reference solutions. The pairs in *NewSubsets* are selected at a time in lexicographical order and the combining, repairing, and local search operators are applied to generate one trial solution. Thus, the size of P will be always $(b^2 - b)/2 + 1$, where one more solution shows the best solution so far.

4.1.3. Combining operators

After the selection of parents $\mathbf{s}^f$ and $\mathbf{s}^m$ from *NewSubsets*, a combining operator combines these two schedules to generate children $\mathbf{s}^s$ and $\mathbf{s}^d$, respectively. We use two different methods as combining operators. We consider the Combining Operator 1 (CO1) as the well-known two-point crossover operator in which the crossing points are determined randomly. If we consider cp_1 and cp_2 as the crossing points where $cp_1 < cp_2$, the elements of $\mathbf{s}^s$ are identical to the elements of $\mathbf{s}^f$, except those placed in positions $[cp_1, cp_2]$ for which elements of $\mathbf{s}^m$ are used to construct $\mathbf{s}^s$. Generation of $\mathbf{s}^d$ is similar to generation of $\mathbf{s}^s$ but with exchanging the roles of $\mathbf{s}^f$ and $\mathbf{s}^m$. For example, consider the two following schedule representations $\mathbf{s}^f$ and $\mathbf{s}^m$ for the RDPSP and assume $cp_1 = 8$ and $cp_2 = 19$.

$$\begin{aligned}\mathbf{s}^f = \{&(1||2), (1 \rightarrow 4), (1||5), (1 \rightarrow 6), (1 \rightarrow 8),\\ &(2 \rightarrow 3), (2 \rightarrow 5), (2 \rightarrow 6), (2 \rightarrow 8), (2 \rightarrow 9),\\ &(3||4), (3||5), (3 \rightarrow 6), (3 \rightarrow 7), (3 \rightarrow 8),\\ &(4||5), (4 \rightarrow 6), (4 \rightarrow 8), (4 \rightarrow 9), (5 \rightarrow 6),\\ &(5 \rightarrow 7), (5||9), (6||7), (6||9), (7||8), (7||9),\\ &(8||9)\},\end{aligned}$$

$$\mathbf{s}^m = \{(1||2),(1||4),(1||5),(1 \leftarrow 6),(1 \rightarrow 8),$$
$$(2 \rightarrow 3),(2 \rightarrow 5),(2 \leftarrow 6),(2 \rightarrow 8),(2 \rightarrow 9),$$
$$(3 \leftarrow 4),(3||5),(3 \leftarrow 6),(3||7),(3 \rightarrow 8),$$
$$(4 \rightarrow 5),(4 \leftarrow 6),(4 \rightarrow 8),(4 \rightarrow 9),(5 \leftarrow 6),$$
$$(5||7),(5||9),(6 \rightarrow 7),(6 \rightarrow 9),(7 \rightarrow 8),$$
$$(7 \rightarrow 9),(8||9)\}.$$

By applying the two-point crossover operator to $\mathbf{s}^f$ and $\mathbf{s}^m$, the two following schedule representations $\mathbf{s}^m$ and $\mathbf{s}^s$ are obtained.

$$\mathbf{s}^s = \{(1||2),(1 \rightarrow 4),(1||5),(1 \rightarrow 6),(1 \rightarrow 8),$$
$$(2 \rightarrow 3),(2 \rightarrow 5),(2 \leftarrow 6),(2 \rightarrow 8),(2 \rightarrow 9),$$
$$(3 \leftarrow 4),(3||5),(3 \leftarrow 6),(3||7),(3 \rightarrow 8),$$
$$(4 \rightarrow 5),(4 \leftarrow 6),(4 \rightarrow 8),(4 \rightarrow 9),(5 \rightarrow 6),$$
$$(5 \rightarrow 7),(5||9),(6||7),(6||9),(7||8),(7||9),$$
$$(8||9)\},$$

$$\mathbf{s}^d = \{(1||2),(1||4),(1||5),(1 \leftarrow 6),(1 \rightarrow 8),$$
$$(2 \rightarrow 3),(2 \rightarrow 5),(2 \rightarrow 6),(2 \rightarrow 8),(2 \rightarrow 9),$$
$$(3||4),(3||5),(3 \rightarrow 6),(3 \rightarrow 7),(3 \rightarrow 8),$$
$$(4||5),(4 \rightarrow 6),(4 \rightarrow 8),(4 \rightarrow 9),(5 \leftarrow 6),$$
$$(5||7),(5||9),(6 \rightarrow 7),(6 \rightarrow 9),(7 \rightarrow 8),$$
$$(7 \rightarrow 9),(8||9)\}.$$

For the ATPSP, if we refer to parents as $\mathbf{s'}^f$ and $\mathbf{s'}^m$ and assume $cp_1 = 4$ and $cp_2 = 7$, by applying the two-point crossover operator to $\mathbf{s'}^f$ and $\mathbf{s'}^m$, we obtain offspring schedules $\mathbf{s'}^s$ and $\mathbf{s'}^d$ as follows.

$$\mathbf{s'}^f = (0,0,11,18,23,2,23,24,25,22,33) \quad \text{and}$$

$$\mathbf{s'}^m = (0,2,0,17,21,13,0,24,25,22,35),$$

$$\mathbf{s'}^s = (0,0,11,17,21,13,0,24,25,22,33) \quad \text{and}$$

$$\mathbf{s'}^d = (0,2,0,18,23,2,23,24,25,22,35).$$

Also, we design the Combining Operator 2 (CO2) based on a path-relinking algorithm developed by Ranjbar et al. [22]. The CO2 gets two parent solutions as inputs and generates a set of child solutions as follows. In this method, the elements of $\mathbf{s}^f$ and $\mathbf{s}^m$ are compared from left to right respectively. The first element of $\mathbf{s}^f$ which differs from the corresponding element of $\mathbf{s}^m$ is changed such that to be identical with $\mathbf{s}^m$. This change creates a new solution ($\mathbf{s}^1$). For example, in the RDPSP, the two following solutions $\mathbf{s}^f$ and $\mathbf{s}^m$ differ in the second element, indicating the precedence relation between activities 1 and 4. Thus, $\mathbf{s}^1$ is a copy of $\mathbf{s}^f$, but in which relation $(1 \rightarrow 4)$ has been changed to $(1||4)$, taken from $\mathbf{s}^m$. Next, we find the first different element between $\mathbf{s}^1$ and $\mathbf{s}^m$ which is related to activities 1 and 6. Using changing relation $(1 \rightarrow 6)$ to $(1 \leftarrow 6)$, $\mathbf{s}^2$ is generated from $\mathbf{s}^1$. This process is repeated until we reach $\mathbf{s}^m$. In this example, 13 different solutions are generated. The CO2 randomly selects two solutions, one solution from the first half of all the generated solutions and another one from the second half. These two solutions are considered as children solutions.

$$\mathbf{s}^f = \{(1||2),(1 \rightarrow 4),(1||5),(1 \rightarrow 6),(1 \rightarrow 8),$$
$$(2 \rightarrow 3),(2 \rightarrow 5),(2 \rightarrow 6),(2 \rightarrow 8),(2 \rightarrow 9),$$
$$(3||4),(3||5),(3 \rightarrow 6),(3 \rightarrow 7),(3 \rightarrow 8),(4||5),$$
$$(4 \rightarrow 6),(4 \rightarrow 8),(4 \rightarrow 9),(5 \rightarrow 6),(5 \rightarrow 7),$$
$$(5||9),(6||7),(6||9),(7||8),(7||9),(8||9)\},$$

$$\mathbf{s}^1 = \{(1||2),(1||4),(1||5),(1 \rightarrow 6),(1 \rightarrow 8),(2 \rightarrow 3),$$
$$(2 \rightarrow 5),(2 \rightarrow 6),(2 \rightarrow 8),(2 \rightarrow 9),(3||4),$$
$$(3||5),(3 \rightarrow 6),(3 \rightarrow 7),(3 \rightarrow 8),(4||5),(4 \rightarrow 6),$$
$$(4 \rightarrow 8),(4 \rightarrow 9),(5 \rightarrow 6),(5 \rightarrow 7),(5||9),$$
$$(6||7),(6||9),(7||8),(7||9),(8||9)\},$$

$$\mathbf{s}^2 = \{(1||2),(1||4),(1||5),(1 \leftarrow 6),(1 \rightarrow 8),(2 \rightarrow 3),$$
$$(2 \rightarrow 5),(2 \rightarrow 6),(2 \rightarrow 8),(2 \rightarrow 9),(3||4),$$
$$(3||5),(3 \rightarrow 6),(3 \rightarrow 7),(3 \rightarrow 8),(4||5),(4 \rightarrow 6),$$
$$(4 \rightarrow 8),(4 \rightarrow 9),(5 \rightarrow 6),(5 \rightarrow 7),(5||9),$$
$$(6||7),(6||9),(7||8),(7||9),(8||9)\},$$

$$\mathbf{s}^3 = \{(1||2),(1||4),(1||5),(1 \leftarrow 6),(1 \rightarrow 8),(2 \rightarrow 3),$$
$$(2 \rightarrow 5),(2 \leftarrow 6),(2 \rightarrow 8),(2 \rightarrow 9),(3||4),(3||5),$$
$$(3 \rightarrow 6),(3 \rightarrow 7),(3 \rightarrow 8),(4||5),(4 \rightarrow 6),$$

$$(4\rightarrow 8),(4\rightarrow 9),(5\rightarrow 6),(5\rightarrow 7),(5||9),(6||7),$$
$$(6||9),(7||8),(7||9),(8||9)\},$$
$$\vdots$$
$$\mathbf{s}^{13}=\{(1||2),(1||4),(1||5),(1\leftarrow 6),(1\rightarrow 8),(2\rightarrow 3),$$
$$(2\rightarrow 5),(2\leftarrow 6),(2\rightarrow 8),(2\rightarrow 9),(3\leftarrow 4),$$
$$(3||5),(3\leftarrow 6),(3||7),(3\rightarrow 8),(4\rightarrow 5),$$
$$(4\leftarrow 6),(4\rightarrow 8),(4\rightarrow 9),(5\leftarrow 6),(5||7),(5||9),$$
$$(6\rightarrow 7),(6\rightarrow 9),(7\rightarrow 8),(7\rightarrow 9),(8||9)\},$$
$$\mathbf{s}^{m}=\{(1||2),(1||4),(1||5),(1\leftarrow 6),(1\rightarrow 8),(2\rightarrow 3),$$
$$(2\rightarrow 5),(2\leftarrow 6),(2\rightarrow 8),(2\rightarrow 9),(3\leftarrow 4),$$
$$(3||5),(3\leftarrow 6),(3||7),(3\rightarrow 8),(4\rightarrow 5),(4\leftarrow 6),$$
$$(4\rightarrow 8),(4\rightarrow 9),(5\leftarrow 6),(5||7),(5||9),(6\rightarrow 7),$$
$$(6\rightarrow 9),(7\rightarrow 8),(7\rightarrow 9),(8||9)\}.$$

The CO2 method follows the same strategy for the ATPSP, but it deals with numbers instead of precedence relations.

4.1.4. Repairing operator

It is possible that the offspring schedules generated by *the initial population generation method* or by *the crossover operator* are infeasible. In order to overcome this problem, we repair each schedule representation after its generation. For this purpose, we generate a random sequence of all pairs $(i,j)\notin A$ and then check the feasibility of the network based on this sequence. Whenever the infeasibility condition is found due to a relation like $(i\rightarrow j)$, we change it as $(i\leftarrow j)$. As an example, consider the schedule representation $\mathbf{s}^s$ introduced in Section 4.3. Since there is loop $2\rightarrow 5\rightarrow 6\rightarrow 2$ in the corresponding project network of this schedule representation, we should change precedence relation $(5\rightarrow 6)$ as $(5\leftarrow 6)$.

The repairing operator for the ATPSP includes two steps. The first step is applied to infeasible solutions for the purpose of making them feasible while the second step is applied to the solutions which are not subject to the concurrency property. In the first step in which the precedence constraints are investigated, for each activity $i\in N$, if $s_i<\max_{j\in Pred(i)}\{f_j\}$, we set $s_i=\max_{j\in Pred(i)}\{f_j\}$. In the second step of the repairing method, for each activity $i\in N$ in which $s_i\notin ES_i$, we shift the activity to the direction that has more positive or less negative impact on the objective function by the minimum amount required such that we have $s_i\in ES_i$. For example, consider schedule representation $\mathbf{s}'^d$, shown in Section 4.3 and depicted in Figure 6.

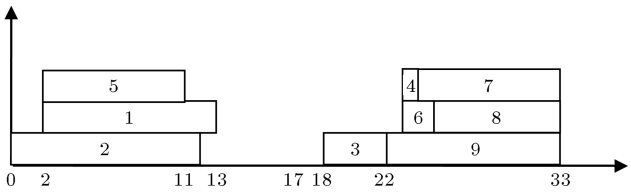

Figure 6. Gantt chart of schedule $\mathbf{s}'^d$.

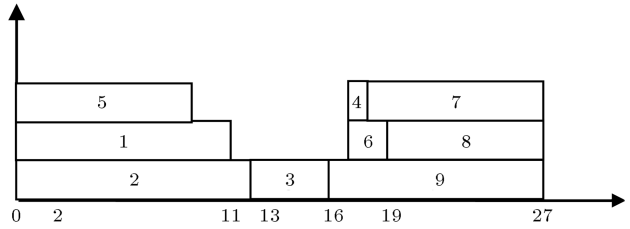

Figure 7. Gantt chart of repaired schedule $\mathbf{s}'^d$.

In this schedule, at first, activity 2 is scheduled which is concurrent with event $t=0$. Then, activities 1 or 5 are scheduled where none of them is concurrent with available events. Thus, we change $s_1=0$ and $s_5=0$. Then, activities 3, 4, 5, 6, 7, 8, and 9 are shifted to the left such that the precedence relations among them is kept and $s_3=f_2=12$. Finally, we obtain schedule representation $\mathbf{s}'^d=(0,0,0,12,17,0,17,18,19,16,27)$, depicted in Figure 7.

4.1.5. Local search operator

A local search operator is applied to each children schedule with the probability of p_{ls} to enforce search intensification. For the RDPSP, this operator changes each unordered pair $(i||j)$ as $(i\rightarrow j)$ and $(i\leftarrow j)$ if each of these directions does not make the project network infeasible. For each direction of each unordered pair of activities, the *eNPV* is calculated. It should be noticed that the impact of each change is evaluated while other unordered pairs of activities are not changed. The current schedule is replaced with the best schedules found in the aforementioned alterations and the local search procedure is repeated.

Similarly, for the ATPSP, we change the start times of activities instead of direction of the unordered pair of activities. For this purpose and for a given schedule $\mathbf{s}$, we should first calculate the $ES_i(\mathbf{s})$ based upon concurrency property for each activity $i\in N$. Then, s_i is changed to each time $t\in ES_i(\mathbf{s})$ while start times of other activities are kept the same. Finally, the impact of each change is evaluated and the best found solution is replaced with the current solution and this procedure is repeated. Both the foregoing local search procedures are stopped whenever no improvement is obtained anymore.

4.2. A genetic algorithm

The Genetic Algorithm (GA) was developed by [24] based on the biological evolution idea for solving compound optimization problems. GA has been widely used in optimization problems [25]. Our GA starts with generation of the initial population P with size of $|P|$. Each element (solution) of the P is generated by the initial population generation method, described in Section 4.1.1. In order to develop a new element P, each element of the current population, which is considered as the father schedule (s^f), is mated randomly with other element of the population, considered as the mother schedule (s^m). They constitute a couple and each couple generates two child schedules, considered as the son schedule (s^s) and the daughter schedule (s^d). The child generation process is performed using three operators, i.e. combining, mutation, and local search. The combining and local search operators have been presented in Sections 4.1.3 and 4.1.5. The mutation operator selects $100p_{\text{mut}}$ percent of cells of each solution and changes them randomly to other defined states. Since the generated child solutions may be infeasible, the repairing operator, described in Section 4.1.4, is also applied in the necessary cases to make the generated solutions feasible.

The child with the better *eNPV* is chosen as the child ($\mathbf{s}^c$) of the parents and is improved by the local search operator with probability of p_{ls}. Next, it is added to the end of P. Then, the population, consisting of the $2|P|$ elements, is updated by deleting half of its elements in such a way that the best individuals will appear in the population more than those with a worse *eNPV* value, emulating the "survival of the fittest" principle of nature. To that purpose, we delete population elements using a biased random sampling method in which the deletion probability of each element is inversely proportional to its *eNPV*. In order to prevent drop of the best-found schedule so far, we will never delete it from the population. This procedure will be continued until the termination criterion is met.

5. Computational results

5.1. Benchmark problem set

All coding was implemented in the Visual C++ 6.0 environment; all experiments were run on a PC Pentium IV 3 GHz processor with 1024 MB of internal memory. For the RDPSP, we used the test sets of [3] including 420 test instances in which $n = 10, 15, 20, 25, 30, 35, 40$ and $OS = 0.25, 0.5$, and 0.75 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 [26]. For each combination of n and OS, 20 test instances were generated varying in the durations of activities, costs, and PTSs: durations and costs are realizations of (independent) discrete uniform random variables on the intervals $[1; 15]$ and $[-50; 0]$, respectively, and the PTS-values are with uniform probability chosen from $[80\%; 100\%]$.

Also, for the ATPSP, we used test sets generated by De Reyck and Leus [3] including 60 test instances in which they chose $n = 6, 8, 10, 12$ and $OS = 0.4, 0.6$, and 0.8. For each combination of n and OS, they generate five test instances. Moreover, durations and costs are realizations of (independent) discrete uniform random variables on the intervals $[1; 10]$ and $[-100; -10]$, respectively, and the values of PTS are with uniform probability chosen from $[50\%; 100\%]$. Unless mentioned otherwise, we set $r = 0.05$.

In order to compare the efficiency of our developed algorithms, we generated two large-scale test sets including 120 test instances of RDPSP with $n = 50, 100$ and 30 test instances of ATPSP with $n = 12, 15$.

5.2. Setting of parameters

Using Design Of Experiments (DOE) technique and based on hard test instances of the RDPSP and ATPSP, we set the SS's parameters p_{ls}, *th-dist*, $|P|$, and b. For each time limit, we considered three levels for the first three parameters as follows: $p_{ls} = 0.01, 0.03$, and 0.05; *th-dist* $= 0.2, 0.4$, and 0.6; $|P| = 10b, 15b$, and $20b$. Also, for each of time limits, $TL = 1$, 10, and 100, we considered three levels for parameter b as shown in Table 3.

For each time limit, we ran a 3^4 full factorial design and found the results of Table 4.

Similarly, we set the parameters of GA as $|P|$, p_{ls}, and p_{mut}, as shown in Table 5.

Table 3. Levels of parameter b.

TL	b
1	4, 6, 8
10	12, 14, 16
100	20, 30, 40

Table 4. Setting of parameter.

TL	b	p_{ls}	*th-dist*	$\|P\|$
1	6	0.03	0.2	$10b$
10	12	0.03	0.4	$10b$
100	30	0.05	0.4	$15b$

Table 5. Levels of the GA's parameter.

TL	p_{ls}	$\|P\|$
1	0.02	50
10	0.03	100
100	0.05	500

Table 6. Comparative results for combining methods.

Time limit		$TL = 1$ second				$TL = 10$ seconds				$TL = 100$ seconds			
Solution method		SS+ CO1	SS+ CO2	GA+ CO1	GA+ CO2	SS+ CO1	SS+ CO2	GA+ CO1	GA+ CO2	SS+ CO1	SS+ CO2	GA+ CO1	GA+ CO2
RDPSP	$n = 50$	23.81	21.44	26.46	24.67	8.18	5.61	9.00	6.14	0.16	0.04	0.20	0.14
	$n = 100$	64.15	56.31	72.48	60.71	12.45	9.10	14.15	11.07	3.61	1.53	4.08	2.23
ATPSP	$n = 15$	19.07	18.50	27.53	19.13	5.87	4.98	7.15	5.21	0.01	0.00	0.02	0.00
	$n = 20$	29.64	24.71	36.38	26.09	8.13	7.13	8.92	7.18	2.07	1.81	2.37	1.96
TAPD		34.17	30.24	40.71	32.65	8.66	6.71	9.81	7.40	1.46	0.85	1.67	1.08

Table 7. Comparative results for the RDPSP.

		TL					
		$TL = 1$ second		$TL = 10$ seconds		$TL = 100$ seconds	
		SS+CO2	B&B	SS+CO2	B&B	SS+CO2	B&B
n	10	1.41(59)	0.00(60)	0.00(60)	0.00(60)	0.00(60)	0.00(60)
	15	8.97(51)	6.34(53)	1.07(58)	1.04(58)	0.00(60)	0.00(60)
	20	14.21(47)	8.94(51)	4.91(54)	2.84(57)	0.00(60)	0.00(60)
	25	17.86(35)	29.94(20)	13.84(45)	26.84(25)	5.143(53)	19.02(35)
	30	26.74(19)	35.83(13)	18.27(33)	31.11(19)	17.94(34)	28.60(23)
	35	45.12(9)	51.59(5)	34.27(20)	37.61(16)	30.12(18)	34.08(20)
	40	50.64(0)	59.71(0)	44.07(8)	54.88(3)	35.61(17)	41.37(14)
	TAPD. (Avg#*opt*)	23.56(31.43)	27.48 (28.86)	16.63(39.71)	22.05(34)	12.69(43.14)	17.58(38.86)

5.3. Comparative computational results

As the termination criterion, we consider three time limits $TL = 1$, 10, and 100 seconds and run our SS, GA, and B&B procedures of [2,3] in the given time limits. The performance of our SS and GA for the RDPSP and ATPSP with two different combining methods have been summarized in Tables 5, 6, and 7, respectively. We report the Average Percent Deviation (*APD*) from the optimal solutions (or best known solutions). The best known solutions for the RDPSP with $n = 10, 15, 20, 25, 30, 35, 40$ and for the ATPSP with $n = 6, 8, 10, 12$ have been determined using B&B algorithms of [2,3]. For other test instances (large-scale test instance), the best found solutions have been obtained by SS or GA. The result of Table 6 shows the performance of SS and GA based on *APD* with two different combining methods of CO1 and CO2. This comparison has been made on the basis of large scale test instances which are more applicable for meta-heuristic algorithms. From Table 6 and based on the Total Average Percent Deviation (*TAPD*), we conclude that SS algorithms combined with CO2 has the best performance in both the RDPSP and the ATPSP. Also, it can be seen that with identical combining operator, SS outperforms GA in average. Thus, in the rest of this paper, we compare only performance of SS+CO2 with exact solution algorithms.

The result of Table 7 shows that the developed SS+CO2 algorithm for the RDPSP dominates the B&B algorithms of [3]. In addition to *APD*, in each cell of this table, the number of times the optimal solution (#*opt*) is obtained is reported inside parenthesis. For test instances with $n = 10$, 15, and 20, the B&B algorithm has better performance than that of our developed SS+CO2, but for larger test instances, our SS+CO2 algorithm outperforms the B&B algorithm (in terms of both *APD* and #*opt*). The last row of Table 7 indicates the summarized results in terms of Total Average Percent Deviation (*TAPD*) and average number of optimal solutions (*Avg*#*opt*). This row indicates that SS+CO2 algorithm surpasses the B&B algorithm.

Similar to those of Table 7, the results of Table 8 indicate that our SS+CO2 algorithm outperforms the B&B algorithm of [2] in average and also for larger test instances ($n = 10$ and 12).

5.4. Sensitivity analysis

In this section, we analyze the impact of repairing and local search operators. On the basis of the

Table 8. Comparative results for the ATPSP.

		TL					
		$TL = 1$ second		$TL = 10$ seconds		$TL = 100$ seconds	
		SS+CO2	B&B	SS+CO2	B&B	SS+CO2	B&B
n	6	0.00(15)	0.00(15)	0.00(15)	0.00(15)	0.00(15)	0.00(15)
	8	0.20(14)	0.17(13)	0.00(15)	0.00(15)	0.00(15)	0.00(15)
	10	0.84(10)	2.44(4)	0.21(13)	0.59(8)	0.00(15)	0.05(14)
	12	2.46(4)	3.35(0)	0.38(9)	2.32(3)	0.21(12)	1.86(5)
	TAPD. (Avg#*opt*)	0.88(10.75)	1.49(8)	0.15(13)	0.73(10.25)	0.05(14.25)	0.48(12.25)

Table 9. Impact of repairing and local search operators on the performance of SS+CO2.

n	RDPSP							ATPSP			
	10	15	20	25	30	35	40	6	8	10	12
SS+CO2/repairing	0.67	2.84	7.14	18.91	25.14	41.57	58.10	0.01	0.08	0.35	0.72
SS+CO2/local search	0.96	2.34	6.18	20.41	25.37	48.63	60.72	0.03	1.38	2.54	2.0

performance of SS+CO2, around 25% of precedence relations generated in RDPSP schedules were repaired (inversed). Also, around 35% of start times generated in ATPSP schedules were changed to find feasible solutions. Moreover, in the second phase of the repairing operator of ATPSP, 18.5% of start times were changed because of the concurrency property. These results indicate that if we do not apply the repairing operator, a noticeable percent of solutions must be regenerated. To show efficiency of the repairing operator, we run the SS+CO2 algorithm over both RDPSP and ATPSP without repairing operator by considering $TL = 10$ seconds. In this case, instead of each infeasible solution, a new random solution is generated until a feasible solution is found. In Table 9, the row titled by "SS+CO2/repairing" indicates the *APD* values while repairing operators have not been applied in the SS+CO2 algorithm. By comparing these outcomes with similar values reported in Tables 5 and 6, we find that performance of the algorithm has decreased by as many as 6.65 and 4.8 percent for RDPSP and ATPSP, respectively.

Also, we run the SS+CO2 algorithm over both RDPSP and ATPSP without their corresponding local search procedures. The last row of Table 9 shows that the performance of SS+CO2 algorithm has decreased by around 7.8 and 6.47 percent for RDPSP and ATPSP, respectively.

6. Summary, conclusions, and further research

In this paper we studied the project scheduling problem under the risk of failure of activities. We considered two introduced problems in the literature: the RDPSP and ATPSP. We developed a scatter search metaheuristic for the mentioned problems with especial components. Using available test sets in the literature, we compared the performance of our developed scatter search algorithm with those of the branch-and-bound algorithms, developed by Ranjbar and Davari [2] and De Reyck and Leus [3] for the ATPSP and RDPSP, respectively. The computational experiments indicated that our algorithm has better performance, especially for larger test instances.

For future research, we propose to consider the modular project scheduling and extend our algorithm for this new problem. In addition, developing new exact and metaheuristic algorithms with more efficient operators for the RDPSP and ATPSP can be interesting research topics.

Acknowledgments

This work is supported by Ferdowsi University of Mashhad as a research project with number 26821 and date 21-5-2013.

References

1. De Reyck, B., Grushka-Cockayne, Y. and Leus, R. "A new challenge in project scheduling: The incorporation of alternative failures", *Tijdschrift voor Economie en Management*, **LII**(3), pp. 411-434 (2007).
2. Ranjbar, M. and Davari, M. "An exact method for scheduling of the alternative technologies in R&D projects", *Computers & Operations Research*, **40**(1), pp. 395-405 (2013).
3. De Reyck, B. and Leus, R. "R&D-project scheduling when activities may fail", *IIE Transactions*, **40**(4), pp. 367-384 (2008).

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

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

6. Herroelen, W. and Leus, R. "Project scheduling under uncertainty: survey and research potentials", *European Journal of Operational Research*, **165**(2), pp. 289-306 (2005).

7. Sabuncuoglu, I. and Goren, S. "Hedging production schedules against uncertainty in manufacturing environment with a review of robustness and stability research", *International Journal of Computer Integrated Manufacturing*, **22**(2), pp. 138-157 (2009).

8. Vieira, G.E., Herrmann, J.W. and Lin, E. "Rescheduling manufacturing systems: a framework of strategies, policies, and methods", *Journal of Scheduling*, **6**(1), pp. 39-62 (2003).

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

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

11. Creemers, S., Leus, R. and De Reyck, B. "Project scheduling with alternative technologies: incorporating varying activity duration variability", In *Proceedings of the IEEE International Conference on Industrial Engineering and Engineering Management (IEEM 2010)*, Macau, pp. 7-10 (December 7-10, 2010).

12. Coolen, K., Wei, W., Talla Nobibon, F. and Leus, R. "Project scheduling with modular project completion on a bottleneck resource", In *Proceeding of the 13th International Conference on Project Management and Scheduling (PMS2012)*, Belgium, pp. 125-128 (April 1-4, 2012).

13. Boros, E. and Ünlüyurt, T. "Diagnosing double regular systems", *Technical Report RRR 30-97, RUTCOR-Rutgers Center for Operations Research*, Rutgers University, US (1997).

14. Ünlüyurt, T. "Sequential testing of complex systems: a review", *Discrete Applied Mathematics*, **142**, pp. 189-205 (2004).

15. Kolisch, R. and Hartmann, S. "Heuristic algorithms for solving the resource-constrained project scheduling problem: classification and computational analysis", In *Project Scheduling - Recent Models, Algorithms and Applications*, Weglarz, J., Ed., Kluwer Academic Publishers, Boston, pp. 147-178 (1999).

16. Herroelen, W.S., Van Dommelen, P. and Demeulemeester, E.L. "Project network models with discounted cash flows a guided tour through recent developments", *European Journal of Operational Research*, **100**(1), pp. 97-121 (1997).

17. Glover, F. and Laguna, M., *Tabu Search*, Kluwer Academic Publishers, Boston (1997).

18. Laguna, M. and Marti, R., *Scatter Search: Methodology and Implementations in C*, Kluwer Academic Press (2003).

19. Martí, R., Laguna, M. and Glover, F. "Principles of scatter search", *European Journal of Operational Research*, **169**(2), pp. 359-372 (2006).

20. Alvarez-Valdes, A., Crespo, E., Tamarit, J.M. and Villa, F. "A scatter search algorithm for project scheduling under partially renewable resources", *Journal of Heuristics*, **40**, pp. 95-113 (2006).

21. Debels, D., De Reyck, B., Leus, R. and Vanhoucke, M. "A hybrid scatter search/electromagnetism metaheuristic for project scheduling", *European Journal of Operational Research*, **169**, pp. 638-653 (2006).

22. Ranjbar, M., De Reyck, B. and Kianfar, F. "A hybrid scatter search for the discrete time/resource trade-off problem in project scheduling", *European Journal of Operational Research*, **193**(1), pp. 35-48 (2009).

23. Yamashita, D.S., Armentano, V.A. and Laguna, M. "Scatter search for project scheduling with resource availability cost", *European Journal of Operational Research*, **169**, pp. 623-637.

24. Holland, H.J., *Adaptation in Natural and Artificial Systems*, The University of Michigan Press, Ann Arbor (1975).

25. Gen, M. and Cheng, R., *Genetic Algorithms and Engineering Optimization*, Wiley (2007).

26. Mastor, A.A. "An experimental investigation and comparative evaluation of production line balancing techniques", *Management Science*, **16**(11), pp. 728-746 (1970).

Biographies

Mohammad Ranjbar was born in 1978 in Mashad, Iran. He received his BSc and PhD degrees from the Department of Industrial Engineering at Sharif university of Technology in 2000 and 2007, respectively. Also, he received his MSc degree from University of Tehran, Faculty of Engineering. In addition, he was a visiting PhD student at London Business School, Department of Operations Management, from October 2005 to March 2006. On February 2008, he joined Ferdowsi University of Mashhad and presently, he is an Associate Professor of Industrial Engineering at this university. Dr. Ranjbar is the author of 22 papers published in national and international journals, 10 papers presented in the international conferences.

Hamidreza Validi was born in 1989 in Mashhad. He received his BSc degree in Industrial Engineering from Ferdowsi University of Mashhad in 2012 and is currently an MSc student of Industrial Engineering at Sharif University of Technology.

Ramin Fakhimi was born in 1990 in Birjand. He received his BSc degree in Industrial Engineering from Ferdowsi University of Mashhad in 2012 and is currently an MSc student of Industrial Engineering at Sharif University of Technology.