Advances in Engineering Software 41 (2010) 966–975

Contents lists available at ScienceDirect

Advances in Engineering Software

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

A hybrid scatter search approach for resource-constrained project scheduling problem in PERT-type networks

Siamak Baradaran [a], S.M.T. Fatemi Ghomi [b,*], Mahdi Mobini [c], S.S. Hashemin [d]

[a] Department of Industrial Engineering, Islamic Azad University, Damavand Branch, Damavand, Iran
[b] Department of Industrial Engineering, Amirkabir University of Technology, 424 Hafez Avenue, Tehran, Iran
[c] Industrial Engineering Group, Faculty of Forestry, University of British Columbia, 2424 Main Mall, Vancouver, Canada V6T 1Z4
[d] Department of Industrial Engineering, Islamic Azad University, Ardabil Branch, Ardabil, Iran

ARTICLE INFO

Article history:
Received 25 November 2008
Received in revised form 15 September 2009
Accepted 20 May 2010
Available online 29 June 2010

Keywords:
PERT networks
Resource-constrained
Project scheduling
Renewable resources
Scatter search algorithm
Path relinking algorithm

ABSTRACT

This paper presents a metaheuristic algorithm for resource-constrained project scheduling problem (RCPSP) in PERT networks. A PERT-type project, where activities require resources of various types with random duration, is considered. The problem is to minimize the regular criterion namely project's makespan. The resource project scheduling model is an NP-hard, therefore to obtain a precise solution, a metaheuristic algorithm is suggested namely hybrid scatter search (HSS). The path relinking algorithm and two operators like crossover and prominent permutation-based are applied to solve the problem. The problem has to be solved at each decision point, when at least more than one activity is ready to be operated but the available amount of resources is limited. The metaheuristic model is illustrated by a numerical example. In order to validate the performance of new hybrid metaheuristic algorithm, solutions are compared with "optimal solution" for small networks. Also the efficiency of the proposed algorithm, for real world problems, in terms of solution quality, is compared with well-reported benchmark test problems available on the PSPLIB. The computational results reveal that the proposed algorithm has appropriate results for small networks and real world problems.

1. Introduction

Project scheduling problem, which has been widely studied since early 1960's, is to determine the schedule of allocating resources so as to balance the total cost and the completion time. The general resource-constrained project scheduling problem (RCPSP) arises when a set of interrelated activities (precedence relations) is given and when each activity can be performed in one of the several ways (modes). Questions arise regarding which resource-duration mode should be adopted, and when should each activity begin so as to optimize some pre-specified managerial goal. The general version of the problem is that each activity could be performed in one of the several ways. For simplicity, this study has been restricted to a continuous duration function where only one execution mode for each activity will be assumed. It also operates with a version of the problem where resources are renewable, activities cannot be interrupted, and the managerial objective is to minimize project duration.

The problem is well recognized as important and difficult since Blazewicz et al. [1] have proven that the RCPSP is NP-hard in a strong sense. Numerous method algorithms have been proposed for solving the RCPSP. Two broad approaches have been used, namely (a) optimization by mathematical programming techniques and (b) heuristic techniques [2]. Solutions based on mathematical optimization seek to define the problem as a mathematical programming problem. The best solution is the one that gives the shortest project duration or the one which provides the smoothest resource profile. However, such optimization techniques remain computationally impractical for most real-life large projects because of the enormous number of variables and constraints. The alternative approach is to develop heuristics which allow a process of choosing between activities that are competing for the use of a scarce resource. The inherent variables in any project scheduling process are time and resources. The various heuristics and their algorithms assist in deciding upon the point on the continuum at which the schedule should end up. Many heuristic models have been developed and are available as computer packages. In this paper we applied the heuristics to solve the RCPSP.

1.1. Heuristics for the RCPSP

A comprehensive survey of the heuristics which have been proposed to solve the RCPSP is provided by Kolisch and Hartmann [3]. The deterministic version of the RCPSP was introduced by Möhring [4], motivated by a small bridge construction project.

* Corresponding author. Tel.: +98 21 66413034; fax: +98 21 66413025.
E-mail address: fatemi@aut.ac.ir (S.M.T. Fatemi Ghomi).

0965-9978/$ - see front matter © 2010 Elsevier Ltd. All rights reserved.
doi:10.1016/j.advengsoft.2010.05.010

Computational experiments were reported and the author showed that the problem is NP-hard. Drexl and Kimms [5] point out that the solution of the RCPSP for several deadlines provides time/cost tradeoffs. This information is valuable when negotiating the price of a project. Recent reviews on exact methods and heuristics for the RCPSP can be found in Herroelen et al. [6], Brucker et al. [7], Kolisch and Hartmann [8], and Hartmann and Kolisch [9].

The literature on solution methods for the RCPSP is scarce. Möhring [4] follows an approach based on graph theory. Specifically, he proposes an exact procedure that employs algorithms for comparability graph and interval graph recognition and orientation. Demeulemeester [10] proposes an exact cutting plane procedure that solves a sequence of RCPSPs. Drexl and Kimms [5] propose two lower bound procedures, based on Lagrangean relaxation and column generation methods. Yamashita et al. [11] develop a scatter search procedure and apply it to a large number of instances. Tormos and Lova [12] considered the RCPSP with makespan minimization as the objective. They improved each generated solution using a backward–forward pass. Kotchetov and Stolyar [13] proposed an evolutionary algorithm based on the path relinking, the tabu search and the variable neighborhood search algorithm to minimize the makespan in the RCPSP. Debels et al. [14] applied a hybrid scatter search-electro magnetism algorithm to minimize the makespan criterion in the RCPSP. Yamashita et al. [11] proposed a scatter search algorithm to minimize the resource availability cost in the RCPSP. Mobini et al. [15] developed an enhanced scatter search algorithm for the RCPSP. In the proposed algorithm, three operators were used to generate new solutions from existing solutions in the reference set.

While there is a vast literature on deterministic project scheduling problems, there is very little work that deals with data uncertainty. However, the complexity of modern projects has forced the development of procedures that deal with uncertainty in key data, such as activity duration. Uncertainty always exists in project scheduling problem, due to the vagueness of project activity duration times. Generally, the uncertainty of activity duration times in project scheduling problem was assumed to be stochastic.

1.2. Stochastic project scheduling

Freeman [16] first introduced probability theory into project scheduling problem in 1960. Charnes and Cooper [17] and Charnes et al. [18] studied project scheduling problem via chance constrained programming in early 1960s. Golenko-Ginzburg and Gonik [19] develop a heuristic procedure for the RCPSP with stochastic activity durations and an objective function that seeks to minimize the expected project duration. The procedure operates in stages where the decision to schedule the next activity is based on the precedence constraints and current resource availability. If several activities are competing for limited resources, a multiple knapsack problem is solved to select the next activities to be scheduled. The objective function of the knapsack formulation represents the maximum total contribution of the selected activities toward the expected project duration. The individual contribution is calculated as the product of the average activity duration and the probability that the activity will be in the critical path during the course of the project. Frequency values found via simulation are used to approximate the true probability values. In a follow-up work, Golenko-Ginzburg and Gonik [20] apply a similar heuristic for a project scheduling problem where the duration of an activity is a random variable that depends on the amount of resources assigned to that activity.

Tsai and Gemmill [21] propose a tabu search that can be applied to the RCPSP with either stochastic or deterministic activity durations. In the stochastic version, the durations are drawn from a probability distribution. A solution to the problem is represented by a feasible sequence, where the order of the activities is consistent with the precedence relationships in the project. Given a particular feasible sequence, the makespan is computed for a sample of activity durations. This sampling process is repeated a specified number of times and the expected makespan is the average of the makespan values. The proposed tabu search makes use of a reduced neighborhood based on feasible swaps that can be executed on the current feasible sequence. Igelmund and Radermacher [22] also address the RCPSP with stochastic activity durations. They propose a set of preselective scheduling policies. These policies, roughly speaking, define, for each possible resource conflict, a preselected activity that is postponed from a set of activities that can not be executed together due to resource conflicts. A branch-and-bound algorithm is developed in order to compute optimal preselective policies, and computational tests are reported for small instances.

Möhring and Stork [23] introduce a new class of scheduling policies, called linear preselective policies intended to minimize the makespan for the RCPSP with stochastic activity durations. This new class combines the benefits of preselective policies and priority policies and is based on both determining sets of activities that can not be executed together due to resource conflicts and choosing an activity to be postponed according to a priority list. Efficient algorithms and a necessary and sufficient criterion for preselective policies are obtained by representing scheduling policies as AND/OR precedence constraints. The approach is tested on 480 instances with 20 activities. Stork [24] compares four scheduling policies to minimize the makespan for the RCPSP with stochastic activity durations. Dominance rules and lower bounds are developed and then applied to branch-and-bound algorithms, which are tested on 1440 instances involving up to 60 activities [25].

The reader may also refer to Elmaghraby [26], Kotiah and Wallace [27], Loostma [28], MacCrimmon and Ryavec [29], and Parks and Ramsing [30] to see how different probability distributions were employed to depict stochastic activity duration times in studying project scheduling problem. Nevertheless, all papers concerning project scheduling problem with stochastic activity duration times just resolved problems concentrating on optimizing completion time under resource or cost limits [31].

This paper presents a new hybrid metaheuristic algorithm based on scatter search for the RCPSP with PERT-type network. Scatter search is an evolutionary or population-based method in which solutions are combined to yield better solutions using convex or non-convex linear combinations. We applied the path relinking algorithm and crossover operator to diversify the search directions and a local search approach to improve the best found solutions. In proposed algorithm we use path relinking concepts to generate children from parent solutions, in the form of a new combination method.

The paper is organized as follows. We start in Section 2 with the problem formulation and solving methods consisting of mathematical model, solution approach and applying scatter search (SS) algorithm in PERT-type network. Section 3 develops the new hybrid metaheuristic algorithm based on scatter search. Experimental design based on benchmark problem sets and computational results are treated in Section 4 and we end with the conclusions in Section 5.

2. Problem definition and solving methods

The resource-constrained project scheduling problem (RCPSP) with network project of PERT-type can be concerned with n non-preemptive activities $j = 1, 2, \ldots, n$ and m renewable resources. The availability of each resource is R_k, $k = 1, 2, \ldots, m$. Activities durations d_j are independent continuous randoms with given distribution function. The problem's model is presented by an activity-on-node (AON) network. Thus there are two dummy activities, first activity and the last activity (0 and $n+1$) which denote start and end of the project, respectively. The dummy start and end activities have zero duration and zero resource consumption. The objective is to minimize the expected project duration, namely project's makespan.

2.1. Mathematical model

Based on the above definitions, RCPSP can be represented formally as the following binary integer program [12]:

$$\min \sum_{t=EF_n}^{T} tx_{nt} \tag{1}$$

s.t.

$$\sum_{t=EF_j}^{T} x_{it} = 1 \quad \forall j, \tag{2}$$

$$\sum_{t=EF_j}^{T} x_{it}(t-d_j) \geqslant \sum_{t=EF_i}^{T} tx_{it} \quad \forall j, \quad i \in P_j, \tag{3}$$

$$\sum_{j=1}^{n} \left(r_{jk} \sum_{q=t}^{t+d_j-1} x_{jq} \right) \leqslant R_k \quad \forall k, \quad t = 1, \ldots, T, \tag{4}$$

$$x_{it} \in \{0,1\} \quad \forall j, \quad t; \tag{5}$$

where the decision variable

$$x_{jt} = \begin{cases} 1 & \text{if activity } j\text{is completed in period } t \\ 0 & \text{otherwise,} \end{cases}$$

and the additional parameters are as follows: *EFj* is the earliest finish time of activity *j*, computed by a forward critical path pass on the project network; *T* is the upper bound on horizon and P_j is the set of total predecessors of activity *j*. Constraints (2) ensure that activity *j* can be finished in one and only one time period, while constraints (3) ensure that precedence relationships between any two activities are not violated and constraints (4) ensure that no resource is used beyond its capacity.

2.2. Solution approach

2.2.1. Solution representation

A solution representation should be computationally fast in the transformation between solutions. Also for each solution in the original space, there is a solution in the encoded space and each encoded solution corresponds to one feasible solution in the original space [32]. Based on the RCPSP literature, the activity-list (AL) representation has been the most widely used representation [3] and has performed better than the other representations. Hence, the AL representation approach is used in the proposed hybrid scatter search (HSS). In the activity-list representation, a precedence feasible activity list is given, in which each activity is posed in the list after all of its predecessors. The position of an activity in an AL determines its relative priority comparing with the other activities.

2.2.2. Schedule generation scheme (SGS)

Since 1963 when Kelley [33] introduced a schedule generation scheme, a large number of different heuristic algorithms have been suggested in the literature [34]. In spite of this drawback, heuristic procedures are used very widely in practice [35]. The scheduling scheme determines the way in which a feasible schedule is constructed by assigning starting times to the different activities. Two different schemes can be distinguished: the Serial Schedule Generation Scheme (S-SGS) and the Parallel Schedule Generation Scheme (P-SGS). Both build feasible project schedules by stepwise extension of a partial schedule (a schedule where only a subset of activities has been scheduled). Activities that, in each step, compete for resources are ordered by a priority rule. S-SGS is an activity oriented scheme and consists of stages, in each of which one activity from the schedulable or decision set is selected and scheduled at the earliest precedence and resource feasible completion time. The P-SGS is a time oriented scheme and in each step of the scheduling process a set of activities (which might be empty) is scheduled. For the resource-unconstrained scheduling problem, both SGS generate feasible schedules that are optimal. In the resource-constrained case, the set of schedules obtained with the S-SGS or the P-SGS have different properties [36].

For the AL representation, the serial-SGS can be used to decode a solution while the parallel SGS cannot be used without modification [12]. In order to use the both decoding approaches in HSS, we use a modification scheme for the AL to change it to a random-key representation (the random-key representation uses an array that assigns a number to each activity such that denotes its priority). In this approach, a priority value which equals to its rank in the AL is assigned to each activity and the obtained list is used as a random-key representation. In HSS, the both serial and parallel schemes are used to decode an AL to a schedule.

3. Development of a new hybrid metaheuristic algorithm based on scatter search

3.1. General overview on scatter search (SS) algorithm

Scatter search is an evolutionary method that has been successfully applied to hard optimization problems. The fundamental concepts and principles of the method were first proposed in the 1970s, based on formulations dating back to the 1960s to combine decision rules and problem constraints. In contrast to other evolutionary methods like genetic algorithms, scatter search is founded on the premise that systematic designs and methods for creating new solutions afford significant benefits beyond those derived from recourse to randomization. It uses strategies for search diversification and intensification that have proved effective in a variety of optimization problems [37].

Scatter search operates on a set of solutions, the reference set, by combining these solutions to create new ones. When the main mechanism for combining solutions is such that a new solution is created from the linear combination of two other solutions, the reference set may evolve as illustrated in Fig. 1. This figure assumes that the original reference set of solutions consists of the circles la-

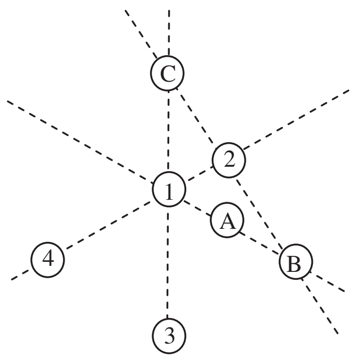


Fig. 1. Two-dimensional reference set.

beled A–C. After a non-convex combination of reference solutions A and B, solution 1 is created. More precisely, a number of solutions in the line segment defined by A and B is created; however, only solution 1 is introduced in the reference set. (The criteria used to select solutions for membership in the reference set are discussed later.) In a similar way, convex and non-convex combinations of original and newly created reference solutions create points 2–4. The resulting complete reference set shown in Fig. 1 consists of seven solutions (or elements).

More precisely, Fig. 1 shows a precursor form of the resulting reference set. Scatter search does not leave solutions in a raw form produced by its combination mechanism, but subjects candidates for entry into the reference set to heuristic improvement, as we elaborate subsequently. Unlike a "population" in genetic algorithms, the reference set of solutions in scatter search is relatively small. In genetic algorithms, two solutions are randomly chosen from the population and a "crossover" or combination mechanism is applied to generate one or more offspring. A typical population size in a genetic algorithm consists of 100 elements, which are randomly sampled to create combinations. In contrast, scatter search chooses two or more elements of the reference set in a systematic way with the purpose of creating new solutions. Since the number of two-element to five-element subsets of a reference set, for example, can be quite large, even a highly selective process for isolating preferred instances of these subsets as a basis for generating combined solutions can produce a significant number of combinations, and so there is a practical need for keeping the cardinality of the set small [38].

A scatter search implementation consists of the following five methods [37]:

(A) *A diversification generation method* to generate a collection of diverse trial solutions, using an arbitrary trial solution (or seed solution) as an input.
(B) *An improvement method* to transform a trial solution into one or more enhanced trial solutions. (Neither the input nor the output solutions are required to be feasible, though the output solutions will more usually be expected to be so. If no improvement of the input trial solution results, the "enhanced" solution is considered to be the same as the input solution.)
(C) *Reference set update method* to build and maintain a reference set consisting of the b "best" solutions found (where the value of b is typically small, e.g., no more than 20), organized to provide efficient accessing by other parts of the method. Solutions gain membership to the reference set according to their quality or diversity.
(D) *A subset generation method* to operate on the reference set, to produce a subset of its solutions as a basis for creating combined solutions.
(E) *Solution combination method* to transform a given subset of solutions produced by the Subset Generation Method into one or more combined solution vectors.

In SS, a set of preferred solutions are preserved in the reference set (refset). Operating on the refset and combining included solutions, new solutions are generated. The refset consists of b solutions and is usually divided to the two subsets with $b1$ and $b2$ solutions. First subset (refset1) is a collection of $b1$ high quality solutions and second one (refset2) is a collection of $b2$ diverse solutions.

3.2. The proposed scatter search algorithm

This section describes steps of the proposed scatter search algorithm and designed methods:

3.2.1. The diversification generation method

This method is applied to generate diverse solutions. In the proposed HSS, the initial population (pool) is generated using the diversification generation method. Two schemes in HSS are devised to generate diverse solutions; a random scheme and a tabu list based approach. The diversification generation method in HSS applies these two schemes to generate the precedence feasible activity lists and subsequently by using SGSs, converts the generated activity lists to feasible schedules. In order to build precedence feasible activity list, the diversification generation method starts from the dummy start activity and adds it into the activity list. Afterwards, the eligible set is formed from those activities which do not exist in the activity list. The eligible set consists of the activities whose predecessors have been added to the list.

According to the literature [15] the solutions generated by the tabu list based scheme are generally more diverse in comparison with the solutions generated by the random scheme. However, the tabu list based scheme needs slightly more attempt in term of CPU time. In the tabu list based scheme, a tabu list of the previously assigned activities in each permutation of the activity list is preserved and the activity which does not exist in the tabu list is selected from elements of the eligible set and the eligible set is subsequently updated. This process is repeated until all the activities would be added in the activity list. The constructed activity list can be converted to a solution using the SGS. In the proposed HSS, the diversification generation method uses both serial and parallel SGS to decode an activity list of a schedule. The best solution found by serial and parallel SGS with the lowest makespan value is accepted as the generated solution. However, each generated activity list corresponds to two solutions.

3.2.1.1. Tabu list based diversification generation method. In this approach, a strategic design is devised in which activities are added from the eligible set to the activity list considering previously generated activity lists. For each permutation of the activity list, a list of the assigned activities in the previously generated solutions is preserved. The length of this list for each permutation is randomly selected between the predetermined lower bound and the upper bound which are calculated with respect to the length of the activity list. The values of the lower bound and the upper bound of the tabu list's length are set to one tenth and one fifth of the activity list's length, respectively. Whenever the eligible set is composed for a permutation, the activities which are the member of the tabu list are omitted and one activity is selected from the remaining activities. It is possible that, in some occasions, the length of the eligible set equals to the length of the tabu list for a permutation. In these cases, as a tie breaking rule, the first activity added to the tabu list is selected to add to the activity list. In other words, a first-in first-out (FIFO) strategy is used.

3.2.2. The improvement method

The improvement method is generally used to transform a solution to one or more enhanced solutions in the scatter search algorithms. In HSS, a forward–backward heuristic [39] is used to improve the previously generated schedules. At first, this method converts the given schedule to a right-justified one by backward-scheduling activities according to non-increasing order of their finish times. In other words, in the forward pass, the activities are sorted according to non-increasing order of their finish times. Afterwards, the activities are considered from right to left and they are scheduled at the latest feasible time. After the forward pass, the produced schedule is converted to a left-justified one in the backward pass. That is, the activities are sorted in non-decreasing order of their start times and they are scheduled from the project's start time (time zero in the project horizon) at the earliest feasible time. Debels et al. [14] showed that this improvement method can only lead to better or equal schedule in comparison with the current schedule.

3.2.3. The reference set update method

The scatter search algorithm constructs and updates the refset using this method. The reference set update method stores the solutions in the refset according to their quality to form the subset refset1 and according to their diversity, to form the subset refset2. In order to maintain appropriate diversity in refset1 a new simple heuristic called dynamic diversification generator (DDG) is used in this study. In DDG, the difference between two solutions is defined as the number of dissimilar elements in their activity lists. For example in Fig. 2, the difference between two activity lists is equal to 7.

DDG examines all the solutions in the refset1 and calculates their difference in comparison with the first solution. Those solutions with the difference lower than φ are replaced with a generated solution using the tabu list based diversification generation method. Thus, using DDG, the desired diversity in the refset1 is preserved. DDG impedes the algorithm from trapping on the local optima. Parameter υ is the initial value of φ. In the first iteration of the algorithm, υ is set to an arbitrary value and during the algorithm's iterations, it is iteratively decreased using Relation (6), where $[(\eta \cdot \upsilon)/N]$ indicates the greatest integer smaller than or equal to $(\eta \cdot \upsilon)/N$. The number of the generated solutions is shown by η and N is the maximum number of the generated solutions. By increasing the number of generated solutions (η), φ decreases and the algorithm's search space will be converged to around the best solution previously found.

$$\varphi = \upsilon - [(\eta \cdot \upsilon)/N] \tag{6}$$

After updating refset, if there is no difference between new refset and the previous one, the refset rebuilding approach is used. In this approach, refset2 is cleared and filled with generated solutions using the diversification generation method.

3.2.4. The subset generation method

Before the combination method is used to generate the new activity lists, the subset generation method forms subsets which are used in the combination method. The proposed subset generation method divides refset solutions into three kinds of desired subsets. Each of these subsets consists of one or two solutions. Table 1 represents these subsets and the executed operators on these subsets. The designed operators are described in the following subsection.

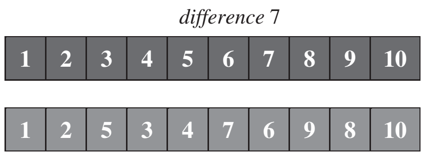


Fig. 2. The difference between two activity lists.

Table 1
Subsets and their corresponding operators.

First solution (subset)	Second solution (subset)	Operator
First solution in refsetj	Second solution in refset1	The path relinking
refset1	–	The permutation-based operator
refset1	refset2	The two-point crossover

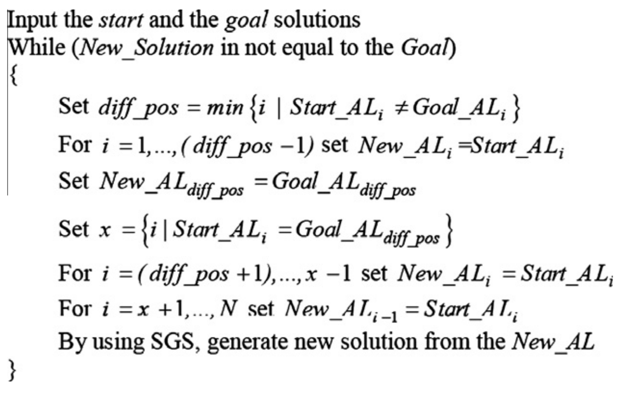
3.2.5. The combination method

In order to generate new solutions, three operators are used in HSS, i.e., the well known two-point crossover, the path relinking, and the permutation-based operator. The proposed operators are described in the three following subsections.

3.2.5.1. Path relinking. The path relinking operator is devised to perform search concentrating on the best founded schedules and the region between them. In each iteration, a path between two activity lists, which have provided the best objective functions, is generated and moves are selected in such a way that common attributes between starting and goal activity lists are preserved and the other parts of the generated solutions are selected in direction of achievement to goal activity list. The newly obtained activity lists are scheduled using the serial-SGS. Fig. 3 presents the designed structure of the proposed path relinking.

The first and the second schedules in the refset are considered as the Goal and the start schedule, respectively. The path relinking operator initially finds the first difference in the activity lists (diff-pos) and copies the part of the activity lists (Goal-AL and Start-AL) located before the diff-pos to a new AL (New-AL). From the Goal-AL, next activity is selected, added to the New-AL and deleted from the Start-AL. The remaining part of the Start-AL is copied to the New-AL. Now, the New AL is a complete AL and can be decoded. At the next iteration, the previously obtained AL (the New-AL in the previous iteration) serves as the Start-AL. This procedure is iteratively repeated until the Start-AL and the Goal-AL become the same. By means of path relinking operator, an activity from Start-AL is shifted to its position in the Goal-AL. Since both the Goal-AL and the Start-AL are precedence feasible activity lists, the New-AL is precedence feasible.

3.2.5.2. Two-point crossover. The crossover operator constructs the offspring from the parents. Two parent activity lists, $F = (f_0, f_1, \ldots, f_{n+1})$ and $M = (m_0, m_1, \ldots, m_{n+1})$, are used to produce one offspring. First, two crossover points, $c1$ and $c2$, are randomly drawn where $0 < c1 < c2 < n + 1$. The first c1 elements are taken from one of the parent lists, added activities from the first parent are eliminated from the second one and next c2–c1 elements are taken from start of the second parent, and the final $n + 1$–$c2$ elements are taken again from the first parent in which the added activities to the generating activity list are eliminated. The two-point crossover operator is used by HSS to elicit offspring from the refset1 as the first parent and the refset2 as the second parent.



```
Input the start and the goal solutions
While (New_Solution in not equal to the Goal)
{
    Set $diff\_pos = min\{i \mid Start\_AL_i \neq Goal\_AL_i\}$
    For $i = 1, \ldots, (diff\_pos - 1)$ set $New\_AL_i = Start\_AL_i$
    Set $New\_AL_{diff\_pos} = Goal\_AL_{diff\_pos}$
    Set $x = \{i \mid Start\_AL_i = Goal\_AL_{diff\_pos}\}$
    For $i = (diff\_pos + 1), \ldots, x - 1$ set $New\_AL_i = Start\_AL_i$
    For $i = x + 1, \ldots, N$ set $New\_AL_{i-1} = Start\_AL_i$
    By using SGS, generate new solution from the New_AL
}
```

Fig. 3. The procedure of the path relinking operator.

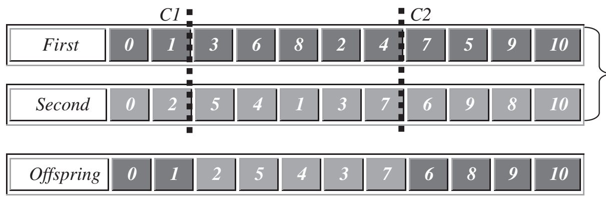


Fig. 4. The two-point crossover operator.

```
For(i=1,2,...n)
{
    Randomly generate $p$ where 0<$p$<1.
    If ($p$<$P_{per}$)
        {
            If ($AL_i$ is not predecessor for $AL_{i+1}$)
                {
                    Replace $AL_i$ with $AL_{i+1}$.
                }
        }
}
```

Fig. 5. The permutation-based operator pseudocode.

Fig. 4 illustrates the implementation of two-point crossover operator.

3.2.5.3. The permutation-based operator. Using this operator, the proposed algorithm transforms the current schedule's activity list to a new one by a controlled manner. In the initial iterations of HSS, this operator changes the current activity list to a new one with a large amount of difference. This prevents the algorithm to converge to local minimum in the initial iterations. Afterwards, in the next iterations, the algorithm's search direction is conducted toward the best existing schedules. Fig. 5 represents the pseudocode of this operator.

In a given AL with a predetermined probability (P_{per}), each element is replaced by the next one if it is not a predecessor for the next activity. Thus, the critical path in input solution could be changed to a new one and a new solution could be generated. A dynamic scheme is used to determine the P_{per}. During the initial iterations of the algorithm, the higher amounts for the P_{per} are used and consequently the P_{per} is diminished. In the next iterations of the algorithm, the P_{per} subsequently decreases on a scale of the generated solutions of algorithm (η) and the maximum amount of the generated solution (N). Relation (7) shows the P_{per} function, where χ and γ are the control parameters.

$$P_{per} = 1 - [(\eta \cdot \chi/N)/\gamma] \tag{7}$$

According to Relation (7), P_{per} decreases in some steps. Each step's length equals to N/χ schedules, i.e., after generating each N/χ schedule, the P_{per} will decrease. The higher amounts for χ lead to generate more solutions in steps of the P_{per} reduction. Furthermore, the higher values for γ lead to the lower difference between P_{per} in steps of reduction. It is notable that the proportion in Relation (8) presents the last amount of the P_{per}. Using Relation (8), another control scheme can be exerted to the P_{per} reduction process. With greater values of χ in comparison with γ, some negative values of the P_{per} will be defined. These values are not acceptable and will be replaced with 0.

$$\min\{P_{per}\} = 1 - \chi/\gamma \tag{8}$$

3.2.5.4. The combination of operators. The combination of the operators outperforms their individual usage, because it collects specifications of the given solutions. Path relinking and the two-point crossover operators are completely dependent on starting solutions, while the permutation-based operator works like a local search and is more promising to reach the higher quality solutions starting from local optima. Therefore when these three operators are working together, the path relinking and the two-point crossover lead the algorithm towards high quality regions of the feasible region and the permutation-based operator causes more precise search in these areas. Combination of operators causes the synergy of individual operators can be utilized.

3.3. Applying scatter search (SS) algorithm in PERT-type network

Limited solution techniques are available for resource constrained single project scheduling with stochastic task durations. Due to computational complexity, optimal solution or heuristics for scheduling have been found useful for large deterministic problems. A welcomed approach, which was utilized in the stochastic project scheduling environment, is simulation. This method is summarized as follows [40]:

(a) utilizing the Monte-Carlo simulation to randomly generate a sample duration for each activity from its estimated duration distribution;
(b) performing a critical path analysis on the network model using these sample durations, and recording the results;
(c) repeating this procedure until acceptable estimate for the time-related variables of interest are obtained.

However, this welcomed approach is extremely time-consuming to be used as a project management tool. The Monte-Carlo simulation is used to investigate behavior of the completion-time random variable, as a function of the number of resources assigned to work on each activity. Also estimated is the distribution of the random variable "resource in use" at each resource level. Estimation of completion time distribution function of stochastic networks with Monte-Carlo simulation is performed as follows [41]:

In this method the random numbers are generated from completion time distribution functions of all activities. Then the lengths of all paths are computed. Time of longest path is the network completion time. This operation is repeated for the number of times we want the network to be simulated. Suppose this number is M, the resource is non-constraint resource and the network consists of l paths. Let U_i^q the completion time of ith path in qth run be represented by

$$U_i^q = \sum_{j=1}^{n} a_{ij} t_j^q, \quad i = 1,2,\ldots,l; \;\; q = 1,2,\ldots,M \tag{9}$$

where

$$a_{ij} = \begin{cases} 1 & \text{if the } j\text{th arc is on the } i\text{th path;} \\ 0 & \text{otherwise} \end{cases}$$

Then we have

$$MKSP^q = \mathrm{Max}\{u_i^q\} \tag{10}$$

Relations (9) and (10) apply in non-constraint resource scheduling. In constrained resource allocation condition, we must use the following algorithm.

An algorithm for completion time distribution	
1.	Clock is initially set to time 0
2.	Simulate the selected network *M* times and compute mean of *ES*, *LS*, *EF*, *LF*, *SLACK* and activities duration times, where *M* is the number of simulation runs
3.	Determine one priority rule (PR), *i.* e., PR1, and continue the procedure for "selected rule"
4.	Set $q = 1$
5.	Form the activity waiting list (AWL). This list contains all activities that can be scheduled at the current clock time. An activity is included on the waiting list if both of the following conditions are met: 10.1. All preceding activities have been scheduled and completed by current clock time, AND 10.2. There are enough resources of each necessary kind available to schedule the activity
6.	Determine the Scheduling Activity Set (SAS) of activities, i.e., activities are ordered according to selected rule, i.e., PR1, with non increase value of activities first
7.	Determine the next activity to be scheduled by selecting an activity, from the Scheduling Activity Set (SAS)
8.	Calculate the scheduled completion time for the activity selected
9.	Reduce the number of units of resource available by the number of units needed for the selected activity
10.	Develop a new activity waiting list (AWL)
11.	Compute the new resource available. In this paper we have renewable resource, therefore the resources allocated to selected activities are freed and added to remaining resources and available resources are considered for $T = T + 1$
12.	If activity waiting list (AWL) is an empty set, stop; record the makespan for this simulation run, i.e., $T^{(q)}$ and go to step 11. Otherwise, go to step 10
13.	$T = T + 1$ and return to step 4
14.	If $q = M$, go to step 12. Otherwise, set $q = q + 1$, return to step 4
15.	Calculate mean of makespan (denoted by $\overline{MKSP}$) from the following relation:

$$\overline{MKSP}_j = \left(\sum_{q=1}^{M} MKSP_j^q\right)/M \text{ where } j \in A \text{ and } q = 1, 2, \ldots, M \qquad (11)$$

where $MKSP^q$ is network completion time in *q*th simulation. The mean of makespan (denoted by $\overline{MKSP}$) is defined in the Relation (11).

4. Experimental design

This section presents the results of computational design and related descriptions in two subsections. The first subsection presents computational experiments for optimal solution in small size networks and compares results against the heuristic solution in order to validate the new algorithm. In the second subsection we apply the new algorithm for heuristic solution in networks of real world sizes.

4.1. Validation of the new metaheuristic algorithm with optimal solution

It is of interest to evaluate the capabilities and efficiency of the new metaheuristic algorithm in the key area of the resource allocation in single project scheduling. The new metaheuristic algorithm has been analyzed with the optimal solution. In this section we analyze the proposed algorithm from the perspective of 'validation results' which assumes that it is possible to apply the results in small network for large networks. It is important to say that the deterministic status of this algorithm has been validated for large networks [15]. This perspective requires the definition of the 'example network' for experimental design.

We assume the activity-on-node (AON) of example network representation, yielding the graph $G(N, A)$; in which the set of nodes *N* represent the *n* "activities" (numbered from 1 to *n*, i.e. $|N| = n$) and the set of arcs A represent the precedence relations between activities, which are assumed to be finish-start relations with a minimal time-lag of zero. There are two dummy activities 0 and $n + 1$ which denote start and end of the project, respectively. The dummy start and end activities have zero duration and zero resource consumption. We also assume that the nodes are topologically ordered so that an arrow leads from a lower number to a larger one; with node 0 as the start node and node $n + 1$ as the end node. We use a regular performance measure, in particular the makespan. That is to say, the objective is to minimize the project makespan.

The new metaheuristic algorithm uses a schedule representation to encode a project schedule and a schedule generation scheme (SGS) to translate the schedule representation to a schedule. The SGS determines how a feasible schedule is constructed by assigning starting times to the activities, whereby the relative priorities are determined by the schedule representation. We represent a schedule *S* of the project by a list $X = (\lambda)$. The list is an activity list $\lambda = (j_0, \ldots, j_{n+1})$, a sequence of activities where activity j_i is the activity number with ith priority for being scheduled using the SGS. We call a "Precedence Feasible Activity List" (PFAL) if every activity is positioned after all of its predecessors in the list [42]. We translate a representation of the project to a schedule *S* using a SGS and denote its project makespan by $\overline{MKSP}$.

The example project is stochastic network and duration of the activities (d_j) are independent continuous random variables. As mentioned previously in Section 1 a fundamental problem in stochastic networks is the completion time of the project [41]. Several authors have used Monte-Carlo simulation to estimate project makespan ($\overline{MKSP}$) in PERT networks.

In this paper the network makespan has been computed by Monte-Carlo (MC) simulation. We computed the mean of makespan ($\overline{MKSP}$) for all of problems through applying each activity list. To be tractable in analysis, assume that the duration for each activity (d_j) is uniform distributed in the interval $[a_j, b_j]$ as follows:

$$d_j \sim Uni(a_j, b_j), \quad \forall j \in N \qquad (12)$$

A single renewable resource, r_j, is assumed to be limited by *R* as follows:

$$r_j \leqslant R, \quad \forall j \in N \qquad (13)$$

4.1.1. Numerical example

Fig. 6 illustrates a project with five non-dummy activities and one resource on an AON network. Dummy activities are at the first and the end of the network, dummy start (S) and finish activities (F). Each node represents an activity.

The upper limit on resource availability is $R = 6$. Table 2 gives the parameters for the uniform distribution ($d_j \sim Uni(a_j, b_j)$) and r_j for each individual activity.

To find optimal solution, we must compute all of factorial combinations (5!) which is 120 activity lists. In this paper, first of all we investigate all of factorial combinations for type I constraint (precedence feasibility). This method helps us to eliminate unfeasible activity lists and decreases number of instances for scheduling. In other words we compute only PFALs.

As we said, the efficiency of the new metaheuristic algorithm in small projects is measured against of "optimal solution" which is the best solution of all PFALs. Thus at first for each PFAL, we simulate 1000 runs and calculate the total project makspan ($\overline{MKSP}$, averaged over 1000 runs). Note that the best solution of the output from the simulation runs provides an "optimal resource allocation solution".

The results showed that the optimal solution using Monte-Carlo sampling for all of precedence feasible activity lists (PFALs) is 12.583. Also we calculate the total project makespan for considered example network with applying the new hybrid metaheuristic algorithm. If T and t are assumed to be random variable and expected completion time (makespan), respectively, we have:

$$\operatorname{Max} P(T \leqslant t|R) \tag{14}$$

where R is the total constrained resource which can be allocated. Relation (14) implies the maximization of the probability of completion time within the desired time if the total allocated resource is R. In this paper we assume t as optimal solution.

As obviously the $P(T \leqslant t)$ must be maximized within some specified period of time. If this time is the best solution, in fact we want to minimize the probability of differences between the optimal solution and the solution of new hybrid metaheuristic algorithm. Fig. 7 shows the cumulative distribution function and suitably illustrates that the probability of project completion time at $t = 12.583$ and with $R = 6$ in maximum is equal to 0.954. Thus we can write

$$\operatorname{Max} P(T \leqslant 12.583|R = 6) = 0.954 \tag{15}$$

The results implies which we have less than 5% differences between the optimal solution and the solution of new hybrid metaheuristic algorithm and indicates the capabilities and efficiency of the new metaheuristic algorithm in the key area of the resource allocation in single project scheduling.

For considered problem the critical path based lower bound is employed. Table 3 represents the results obtained by the proposed algorithm. The number of schedules as the algorithm's stopping criterion is 1000. The first and the second columns represent the average and the maximum percentage deviation from the lower bounds, respectively. The probability that HSS presents the near-optimal solution in the considered problem is illustrated in the third column. The fourth column consists of the average deviation percentage from the best solution in considered example. The fifth and the sixth columns indicate the average and the maximum CPU time, respectively.

Table 3 shows the percent differences between the values ($\overline{MKSP}$) of the best possible solution which is the minimum of $\overline{MKSP}$ and critical path as the lower bound (LB). The probability that HSS presents the near-optimal solution in the considered problem is illustrated in 17.4% and 30% differences are observed for average and maximum deviation of critical path, respectively.

The average and maximum computational (CPU) time are 0.009 and 0.056 for generating 1000 solutions, respectively. With generation of 1000 solutions as the stopping criterion, HSS is capable to give the near-optimal solutions for 95.4% problems and this fact is an evidence of appropriate performance of HSS.

4.2. Benchmark problem sets

Previous section showed appropriate performance of HSS. This section applies the proposed algorithm for large size problems. In order to evaluate the efficiency of the heuristic developed in this paper in real world situations, we will use the standard test problems. The standard test problems available on a World Wide Web called PSPLIB are employed to evaluate the algorithm's performance. These test problems consist of three sets of J30, J60 and J120 instances generated by the problem generator ProGen designed by Kolisch and Sprecher [43].

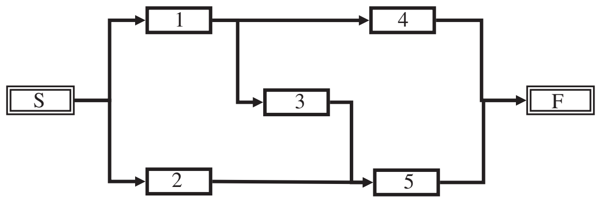


Fig. 6. Example network.

Table 2
Parameters of the uniform distribution (d_j) and r_j for each individual activity of example network.

Activity j	Uniform distribution interval $[a_j, b_j]$	r_j
1	[2,4]	4
2	[3,8]	3
3	[1,3]	3
4	[4,8]	3
5	[2,5]	3

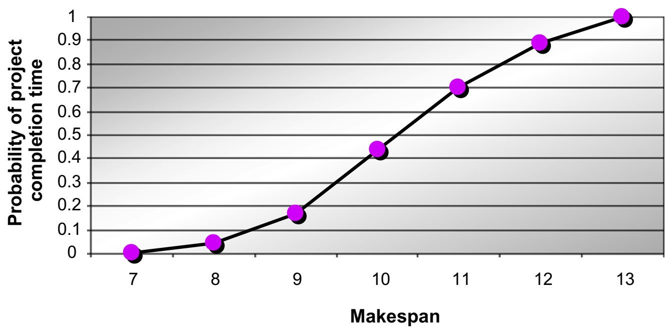


Fig. 7. The cumulative distribution function curve for example network using new hybrid metaheuristic algorithm.

Table 3
The obtained results for the considered example.

Ave. LB dev. (%)	Max. LB gap (%)	Prob. of best	Ave. best dev. (%)	Ave. CPU time	Max. CPU time
0.174	0.30	0.954	0.03	0.009	0.056

The sets J30 and J60 consist of 480 instances with 30 and 60 non-dummy activities, respectively and the set J120 consists of 600 instances with 120 non-dummy activities. For each problem size, instances have been systematically generated by varying three parameters: the network complexity, the resource factor, and the resource strength.

The network complexity reflects the average number of the immediate successors of an activity. The resource factor is a measure of the average number of resources requested per job. The resource strength describes the scarceness of the resource capacities [44]. In order to evaluate the performance of the HSS, we use these test problems and investigate the efficiency of new hybrid metaheuristic algorithm in larger projects which can be measured with regard to critical path as the lower bounds [45]. Regarding the duration of activities, we consider a stochastic and continuous time, we calculate the makespan for all of these test problems by M times network simulation with non-constraint resources and compare the results with constraint resources condition. Experiments were performed on a personal computer with Intel Pentium IV 3 GHz CPU and 1 GB RAM under windows XP professional. The algorithm was coded in Visual C.

4.2.1. Parameters setting

As mentioned before, the different levels of the parameters of the permutation-based operator (γ and χ) and the different level of their proportion affect the convergence of this operator and consequently, affect the convergence of the algorithm. Besides, while DDG heuristic parameter (v) serves as a threshold for replacing an assessed schedule from the refset2, very high values of this parameter can lead to interruption in the search process and also very low values can lead to inconvenient convergence during the process. Moreover, the effects of the reference set parameters (refset size and $b1$) are undeniable.

In order to tune parameters, a factorial analysis is conducted on the "test set". Table 4 indicates values of the effective parameters. According to this table, the factorial analysis includes 486 different combinations according to Table 4. Using each combination, 1000 schedules are generated for each of the 200 instances in the "test set". Among the 486 combinations, the combinations with the best obtained results, the fastest combination, and the worst combination are determined. The results indicate that the best performance of the algorithm in term of solution quality is achieved by the combination (100, 100, 6, 10, 7, and 100) for the parameters.

4.2.2. Computational results

Table 5 summarizes the results of proposed algorithm for the standard test problems. We report the average and maximum deviation from the lower bound for each obtained solution. Also the average and median values for the required CPU time are reported. The number of schedules as the algorithm's stopping criterion is 1000. In Table 5 the first column indicates the instance set.

Table 4
Values of the effective parameters.

Parameter	Values
Parameters of the permutation	γ = (50, 100, 150) χ = (50, 100, 150)
DDG heuristic parameter	v = (2, 6, 8)
The reference set parameters	Poolsize = (50, 100, 200) refset size = (10, 15) $b1$ = (3, 5, 7)

Table 5
The results for the standard test problems.

Problem set	Ave. LB dev. (%)	Max. LB gap (%)	Ave. CPU time	Max. CPU time
J30	0.090	0.490	0.561	1.402
J60	0.064	0.423	1.145	3.233
J120	0.145	0.575	2.394	5.420

The second and the third columns represent the average and the maximum percentage deviation from the lower bounds, respectively. For sets, the critical path based lower bound is employed. The fourth and the fifth columns indicate the average and the maximum CPU time, respectively.

We indicated the validation of HSS algorithm in previous section to compare the optimal solution in small network. In this section we analyzed the HSS results for large networks. The obtained results for the standard test problems indicate that HSS is capable for the problem sets. For J30 problem set the average deviation from the LB solution equals 9%. It is also worth noting that these results are obtained within the average computational time of 0.561 s for generating 1000 solutions which is considerably rational and promising to solve real world problems. Moreover, the generation of 1000 solutions using HSS algorithm leads to 6.4% for J60 and 14.5% for J120 of the problems set for the average deviation from the LB solution. Also the maximum percentage gap from the LB solution for J30, J60 and J120 are 49, 42 and 57, respectively. Columns "Ave. CPU time" and "Max. CPU time" contain the results for each problem set, that is, J30, J60, and J120. All of these results also support the efficiency of HSS. It is important to say among three problem sets used in this paper for real world problems, the proposed HSS has first, second and third ranks of efficiency to provide solution for J60, J30, and J120 problem sets, respectively.

5. Conclusions

This paper presents a metaheuristic algorithm to solve the stochastic RCPSP based on scatter search algorithm. A PERT-type network, where activities require resources of various types with random duration, is considered. The problem is to minimize the regular criterion namely project's makespan. Results for project completion time are provided from Monte-Carlo (MC) simulation runs. The experiments are conducted to validate the new algorithm through the comparisons of results between optimal solution and solution of developed algorithm in small size networks. Also the efficiency of the proposed algorithm, for real world problems, in terms of solution quality, is compared with well-reported benchmark test problems available on the PSPLIB. In developed metaheuristic algorithm based on scatter search, the activity-list representation method was used as the encoding scheme. In the initial population, activity lists were decoded to the solutions using both serial and parallel SGS and serial-SGS was used during the iterations of the algorithm. Three operators are applied to generate new solutions, path relinking, two-point crossover, and permutation-based operator. Using these operators and a scatter search framework, an efficient algorithm is proposed for the stochastic RCPSP. The performance of the algorithm is tested on small networks which indicate that the new algorithm outperforms the near-optimal solution in most cases. Moreover, the new algorithm gives good results in the standard test problems, compared to the critical path based lower bound. All of these results illustrate the capability of new algorithm in small and large problems of real world situation.

References

[1] Blazewicz J, Lenstra JK, Rinnooy Kan AHG. Scheduling subject to resource constraints. Discrete Appl Math 1983;5:11–24.
[2] Abeyasinghe M, Chelaka L, Greenwood David J, Eric Johansen D. An efficient method for scheduling construction projects with resource constraints. Int J Project Manage 2001;19:29–45.
[3] Kolisch R, Hartmann S. Experimental investigation of heuristics for resource-constrained project scheduling: an update. Eur J Oper Res 2006;174:23–37.
[4] Möhring RH. Minimizing costs of resource requirements in project networks subject to a fixed completion time. Oper Res 1984;32:89–120.
[5] Drexl A, Kimms A. Optimization guided lower and upper bounds for the resource investment problem. J Oper Res Soc 2001;52:340–51.
[6] Herroelen W, De Reyck B, Demeulemeester E. Resource constrained project scheduling: a survey of recent developments. Comput Oper Res 1998;25:279–302.
[7] Brucker P, Drexl A, Mohring R, Neumann K, Pesch E. Resource-constrained project scheduling: notation, classification, models and methods. Eur J Oper Res 1999;112:3–41.
[8] Kolisch R, Hartmann S. Heuristic algorithms for the resource constrained project scheduling problem: classification and computational analysis. In: Weglarz J, editor. Project scheduling: recent models, algorithms, and applications. Boston: Kluwer Academic; 1998. p. 147–78.
[9] Hartmann S, Kolisch R. Experimental evaluation of state-of the- art heuristics for the resource-constrained project scheduling problem. Eur J Oper Res 2000;127:394–407.
[10] Demeulemeester E. Minimizing resource availability costs in time limited project networks. Manage Sci 1995;41:1590–8.
[11] Yamashita DS, Armentano VA, Laguna M. Scatter search for project scheduling with resource availability cost, special issue on scatter search. Eur J Oper Res 2006;169:623–37.
[12] Tormos P, Lova A. A competitive heuristic solution technique for resource-constrained project scheduling. Ann Oper Res 2001;102:65–81.
[13] Kochetov Y, Stolyar A. Evolutionary local search with variable neighborhood for the resource constrained project scheduling problem. In: Proceedings of the 3rd international workshop of computer science and information technologies, Russia; 2003.
[14] Debles D, Reyck BD, Leus R, Vanhoucke M. A hybrid scatter search/ electromagnetism meta-heuristic for project scheduling. Eur J Oper Res 2006;169:638–53.
[15] Mobini DMM, Rabbani M, Amalnik S, Razmi J, Rahimi-Vahed A. Using an enhanced scatter search algorithm for a resource-constrained project scheduling. Soft Comput doi: 10.1007/s00500-008-0337-5 [Published online: 28.07.08].
[16] Freeman RJ. A generalized PERT. Oper Res 1960;8:281–93.
[17] Charnes A, Cooper WW. Deterministic equivalents for optimizing and satisficing under chance constraints. Oper Res Int J 1963;11:18–39.
[18] Charnes A, Cooper WW, Thompson GL. Critical path analysis via chance constrained and stochastic programming. Oper Res Int J 1964;12:460–70.
[19] Golenko-Ginzburg D, Gonik A. Stochastic network project scheduling with non-consumable limited resources. Int J Prod Econ 1997;48:29–37.
[20] Golenko-Ginzburg D, Gonik A. A heuristic for network project scheduling with random activity durations depending on the resource allocation. Int J Prod Econ 1998;55:149–62.
[21] Tsai YW, Gemmill DD. Using tabu search to schedule activities of stochastic resource-constrained projects. Eur J Oper Res 1998;111:129–41.
[22] Igelmund G, Radermacher FJ. Algorithmic approaches to preselective strategies for stochastic scheduling problems. Networks 1983;13:29–48.
[23] Möhring RH, Stork F. Linear preselective policies for stochastic project scheduling. Math Methods Oper Res 2000;52:501–15.
[24] Stork F. Branch-and-bound algorithms for stochastic resource-constrained project scheduling. Technical rep. 702-2000. Combinatorial optimization & graph algorithms group, Technische Universität Berlin; 2000.
[25] Yamashita DS, Armentano VA, Laguna M. Robust optimization models for project scheduling with resource availability cost. J Sched 2007;10:67–76.
[26] Elmaghraby SE. On the expected duration of PERT type network. Manage Sci 1967;13:299–306.
[27] Kotiah TCT, Wallace ND. Another look at the PERT assumptions. Manage Sci 1973;20:44–9.
[28] Loostma FA. Network planning with stochastic activity durations, an evaluation of PERT. Stat Neerl 1966;20:43–69.
[29] MacCrimmon KR, Ryavec CA. An analytical study of PERT assumptions. Oper Res 1964;12:16–37.
[30] Parks WH, Ramsing KD. The use of the compound poisson in PERT. Manage Sci 1969;15:397–402.
[31] Ke Hua, Liu Baoding. Project scheduling problem with stochastic activity duration times. Appl Math Comput 2005;168:342–53.
[32] Palmer C, Kershenbaum A. Representing trees in genetic algorithms. In: Proceedings of the first IEEE international conference on evolutionary computation, New York; 1994. p. 376–84.
[33] Kelley JE. The critical-path method: resources planning and scheduling. In: Muth J, Thompson G, editors. Industrial scheduling. New Jersey: Prentice-Hall; 1963.
[34] Kolisch R, Padman R. An integrated survey of project scheduling. Technical report 463, Manuskripte aus den Instituten für Betriebswirtschaftslehre der Universitat Kiel; 1997.
[35] Moder JJ, Phillips CR, Davis EW. Project management with CPM, PERT and precedence diagramming. 3rd ed. New York: Van Nostrand Reinhold International Company Limited; 1983.
[36] Kolisch R. Serial and parallel resource-constrained project scheduling methods revisited: theory and computation. Eur J Oper Res 1996;90:320–33.
[37] Marti R, Laguna M, Glover F. Principles of scatter search. Eur J Oper Res 2006;169:359–72.
[38] Laguna M, Marti R. Scatter search: methodology and implementations in C. Operations research/computer science interfaces series. Boston: Kluwer; 2003.
[39] Li KY, Willis RJ. An iterative scheduling technique for resource-constrained project scheduling. Eur J Oper Res 1992;56:370–9.
[40] Kurihara K, Nishiuchi N. Efficient Monte-Carlo simulation method of GERT-type network for project management. Comput Ind Eng 2002;42.
[41] Fatemi Ghomi SMT, Hashemin SS. A new analytical algorithm and generation of Gaussian quadrature formula for stochastic network. Eur J Oper Res 1999;114:610–25.
[42] Ranjbar M, De Reyck B, Kianfar F. A hybrid scatter search for the discrete time/ resource trade-off problem in project scheduling. Eur J Oper Res 2009;193(1):35–48.
[43] Kolisch R, Sprecher A. PSPLIB-a project scheduling problem library. Eur J Oper Res 1997;96:205–16.
[44] Hartmann S. A competitive genetic algorithm for resource-constrained project scheduling. Naval Res Logist 1998;45:733–50.
[45] Maroto C, Tormos P, Lova A. The evolution of software quality in project scheduling. In: Weglarz J, editor. Handbook on project scheduling: recent models. Algorithms and applications. Dordrecht: Kluwer; 1998 [chapter 11].