OPSEARCH (2020) 57:935–985
https://doi.org/10.1007/s12597-020-00447-8

THEORETICAL ARTICLE

Evolutionary algorithms for multi-objective stochastic resource availability cost problem

Masoud Arjmand[1] · Amir Abbas Najafi[2] · Majid Ebrahimzadeh[1]

Accepted: 7 May 2020 / Published online: 23 May 2020

Abstract

This paper investigates the resource availability cost problem in a PERT-type network, where both activities duration and resource requirement are considered as stochastic parameters. The problem has two objective functions in which the first one, namely the project's makespan, is to minimize the project's duration. However, the second one tries to minimize the total cost of resources. Since its NP-hardness is proven in a strong sense, four well-known evolutionary algorithms including strength pareto evolution algorithm II, non-dominated sorting genetic algorithm II, multi-objective particle swarm optimization, and pareto envelope-based selection algorithm II are proposed to solve the problem. Furthermore, to enhance the algorithms' performance, some efficient mutation and crossover operators, as well as two novel operators called local search and movement, are employed to solution structure for producing new generations. Also, in order to tackle uncertainty, Monte-carlo simulation is utilized. In order to tune the effective parameters, the Taguchi method is used. The performance of our proposed algorithms is evaluated by numerical test problems in different size which generated based on PSPLIB benchmark problems. Finally, to assess the relative performance of the four proposed algorithms, six well-known performance criteria are employed. Using relative percentage deviation and TOPSIS approach, the performance of algorithms is elucidated.

Keywords Scheduling · Resource availability cost problem · Multi-objective evolutionary algorithms · Monte-Carlo simulation · TOPSIS approach

✉ Masoud Arjmand
m.arjmand.ie@gmail.com

1 Department of Industrial Engineering, Faculty of Industrial and Mechanical Engineering, Qazvin Branch, Islamic Azad University, Qazvin, Iran

2 Faculty of Industrial Engineering, K. N. Toosi University of Technology, Tehran, Iran

1 Introduction

In the early 1960s, the project scheduling problem is decided by the schedule of allocating resources in order to optimize an objective function. Since Blazewicz et al. [6] proved that the NP-hardness of RCPSP, the problem has been widely studied. The decision variables for the RCPSP are the starting time of the activities while the objective is to minimize the completion time of the project. For a comprehensive survey on exact and heuristic procedures, which have been applied to solve the deterministic RCPSP refers to Icmeli et al. [26], Elmaghraby, [18], Herroelen et al. [24], Demeulemeester et al. [15] and Kolisch and Hartmann [32].

The resource availability cost problem (RACP) is an extended form of RCPSP, which introduced by Mohring as the resource investment problem [41]. solving the problem, he proposed an exact method. Besides, the author proved that the problem belongs to the NP-hard class of problems due to its complexity. The RACP consists of scheduling the activities subject to the total cost of the required resources is minimized. In RACP, both activities' start time, and the resources' capacity value are decision variables. Besides, precedence relations, as well as a fixed deadline, are imposed. It is also assumed that the resources (no matter if they are employed or not) are assigned to the project for the total project duration, and the unit cost of each resource is to be fixed independently of its period of availability.

Plenty of studies have been fulfilled in this topic. Rangaswamy [50] developed a branch-and-bound algorithm to solve the RACP. To validate the algorithms, he solved a set of problems introduced by Demeulemeester [14]. Drexl and Kimms [16] presented two lower-bound approaches for the RACP. Rodrigues and Yamashita [52] introduced an exact algorithm. To study about the heuristic and meta-heuristic methods in detail, which have been applied to solve RACP, refers to Yamashita et al. [63], Shadrokh and Kianfar [54], Ranjbar et al. [51], and Van Peteghem and Vanhoucke [61]. Nadjafi [44] defined a multi-mode RACP with recruitment and release dates for resources. To solve the problem, he proposed the simulated annealing algorithm. Finally, Arjmand and Najafi [1] proposed meta-heuristic algorithms to solve a multi-mode RACP in the determined environment.

Compare to the vast literature on deterministic project scheduling problems, there are minimal works considering uncertainty in the scheduling problems. Nonetheless, the complexity of the real project has forced scholars to consider uncertainty in the problem. A good example of which is vagueness in activity durations. Because of the ambiguity in activity durations, uncertainty exists in a project scheduling problem. Initially, Freeman [20] presented probability theory into project scheduling problem. A substantial issue in stochastic networks with non-deterministic activity duration is the total completion time of the project [23]. To deal with stochastic networks, authors employed different methods, i.e., Martin [40] applied series–parallel reductions to analyze PERT networks. Charnes et al. [7] presented a chance-constrained programming (CCT) approach to solve PERT-type problems. Fatemi Ghomi and Hashemin [19] generalized the Gaussian quadrature formula to compute F(T). Kulkarni and Adlakha [35] applied a continuous-time Markov process method to PERT-type networks considering exponentially distributed activity durations. Elmaghraby [17] calculated lower bounds for the expected project completion time. Besides, several authors have applied the Monte Carlo simulation (MCS) to estimate F(T) in PERT networks, e.g., Golenko-Ginzburg and Gonik [21] employed a heuristic method for the problem in which the duration of activities are random variables. Tsai and Gemmil [60] propose a tabu search that can be applied to the RCPSP whether it has stochastic or deterministic activity duration times. Möhring and Stork [42] presented linear preselective policies to minimize the makespan with non-deterministic activity durations. Stork [55] compares four different scheduling policies to minimize the makespan in stochastic RCPSP. Ke and Liu [28] employed a genetic algorithm to solve the RCPSP with stochastic activity durations. Baradaran and Fatemi-Ghomi [2] introduced a hybrid heuristic rule to solve the problem. Later on, they presented a hybrid algorithm based on scatter search [3]. Furthermore, they presented the multi-mode stochastic RCPSP in which each activity has several execution modes and solved it with the same method [4]. Mukherjee and Basu [43] developed a method for solving an internal PERT/CPM in AOA networks. This method involves tabular, which is more intelligible for both technical and non-technical persons. Yellapu and Penmestsa [64] presented a mathematical model for stochastic RACP where availability to resources is periodical and described by resource calendar. To solve the problem, they employed a heuristic algorithm. Goto [22] developed a max-plus-linear (MPL) representation to model and analyze discrete-event systems. Ning et al. [45] considered multi-mode cash flow balanced project scheduling problem with stochastic activity durations. To solve the problem, two meta-heuristic algorithms, namely Tabu Search (TA) and Simulated Annealing (SA) were developed. Their objective was to minimize the contractor's maximal cumulative gap between cash outflows and cash inflows. Khalilzadeh et al. [27] presented a heuristic algorithm for project scheduling with fuzzy parameters. Chen et al. [8] studied the performance of 17 priority rule heuristics and the justification technique on stochastic project scheduling problems. The outcome proved that the best priority rules performed as well as best meta-heuristic when the variance of activity duration was medium and outperformed all algorithms when this variance was high. Finally, Creemers [12] studied preemptive stochastic project scheduling problem in which activity durations are exponentially distributed. The author developed a new Markov chain to find an optimal solution.

Another emerging research area in this field considers flexible networks for project scheduling problem, in which some of the activities of the project may not be implemented. Several authors did research considering various assumptions. Kellenbrink and Helber [29] presented RCPSP with the flexible project structure, in which the activities that must be scheduled are not totally known. They employed a genetic algorithm to solve the problem. Tao et al. [58] investigated a project scheduling problem with hierarchical alternative methods regarding uncertain activity durations. A meta-heuristic combining average sample approximation with an artificial algae algorithm is developed to solve the problem. Experimental results showed that the proposed method outperformed GA. Tao and Dong [57] considered resource constraint project scheduling problem with alternative activity chain inspired form project scheduling practices. They designed an AND-OR project network representation for the problem. To solve the problem, an extended simulated annealing algorithm was proposed. Later on, They extended their research considering multi-mode activities for the project [59]. They employed hybrid meta-heuristic algorithms to resolve the issue.

Resource unavailabilities in project scheduling problem is another term in this regard. Lambrechts et al. [36] defined uncertainty as stochastic resource availability. They presented two parameters to model resources' breakdown: meantime of failure of resources, and mean time to repair resources. They aimed at generating a stable baseline schedule for the problem. Therefore, they presented a tabu search procedure operating on a surrogate, free slack-based objective function [37]. They continued their work on resource constraint project scheduling problem subject to resource unavailabilities [38]. In this paper, they determined the impact of unexpected resource breakdown on activity durations. Using this information, they developed an approach in order to insert exact idle time into the project schedule. Ma et al. [39] introduced the best surrogate measures for two types of disruptions in project scheduling, i.e., resource availability disruptions and activity duration disruptions. To deal with the above disruptions, they proposed a general framework of slack-based surrogate robustness measures.

More detailed about the differences and similarities between this paper and the mentioned paper regarding stochastic project scheduling can be found in Table 1.

To the best of our knowledge, all papers concerning project scheduling problem with stochastic activity duration times just resolved problems concentrating on optimizing completion time under resource or cost limits. In addition, there is a few research in the field of RACP, considering both stochastic activity durations and resource requirements, simultaneously. To bridge the gap, in this paper, a resource availability cost problem with two types of uncertain environments, i.e., stochastic resource availabilities and stochastic activity durations, are taken into account. Furthermore, the problem is assumed with two objective functions; the first one, namely makespan, which minimizes the project completion time, and the second one tries to reduce the total resource cost. In order to deal with the uncertainty, we used Monte Carlo simulation (MCS). To solve the problem, four well-known meta-heuristic algorithms, namely SPEA-II, NSGA-II, PESA-II, and MOPSO, are employed. To evaluate the performance of the algorithms, a set of 90 problems are generated based on PSPLIB benchmark problems. Also, six performance criteria are applied to illustrate the algorithms' performance.

The remainder of the paper is set out as follows. Section 2 is started with the problem formulation consisting of a mathematical model and notations. In Sect. 3, the solution approaches and meta-heuristic algorithms applied in the PERT-type network are defined. In Sect. 4, computational results are treated. Finally, in Sect. 5, the conclusion is explained.

Table 1 Differences and similarities between this paper and other related works

References	1st objective (2nd objective)	Uncertainty environments				Other assumptions		Algorithm(s)
		Uncertainty in Resource Availability	Activity Duration variability	Alternative Activity chain	Fuzzy parameters	Multi-mode activities	Preemptible activities	
Charnes et al. [7]	Makespan	×	✓	×	×	×	×	Chance-Constrained Programming (CCT)
Kulkarni and Adlakha [35]	Makespan	×	✓	×	×	×	×	Continuous-time Markov process
Golenko-Ginzburg and Gonik [21]	Makespan	×	✓	×	×	×	×	Heuristic Algorithm
Tsai and Gemmil [60]	Makespan	×	✓	×	×	×	×	Tabu search Algorithm
Fatemi Ghomi and Hashemin [19]	Makespan	×	✓	×	×	×	×	Analytical Algorithm
Möhring and Stork [42]	Makespan	×	✓	×	×	×	×	Linear Preselective Policies
Stork [55]	Makespan	×	✓	×	×	×	×	Branch-and-Bound Algorithm
Ke and Liu [28]	Makespan	×	✓	×	×	×	×	Genetic Algorithm
Lambrechts et al. [36]	Robustness	✓	×	×	×	×	×	Priority rule based
Lambrechts et al. [37]	Robustness	✓	×	×	×	×	×	Tabu Search Algorithm
Baradaran and Fatemi-Ghomi [2]	Makespan	×	✓	×	×	×	×	Hybrid Heuristic
Baradaran et al. [3]	Makespan	×	✓	×	×	×	×	Hybrid Scatter Search (HSS) Algorithm
Mukherjee and Basu [43]	Makespan	×	✓	×	×	×	×	Simplified Tabular Method
Lambrechts et al. [38]	Robustness	✓	×	×	×	×	×	Multiple Algorithms
Baradaran and Fatemi-Ghomi [4]	Makespan	×	✓	×	×	✓	×	Hybrid Metaheuristic Algorithm (HMA)

Table 1 (continued)

References	1st objective (2nd objective)	Uncertainty environments				Other assumptions		Algorithm(s)
		Uncertainty in Resource Availability	Activity Duration variability	Alternative Activity chain	Fuzzy parameters	Multi-mode activities	Preemptible activities	
Kellenbrink and Helber [29]	Makespan	×	×	✓	×	×	×	Genetic Algorithm
Yellapu and Penmestsa [64]	Resource cost	×	✓	×	×	×	×	Heuristic method
Goto [22]	Makespan	×	✓	×	×	×	×	Max-Plus Linear (MPL)
Ning and et al. [45]	Robustness	×	✓	×	×	✓	×	SA and TS algorithms
Khalilzadeh and et al. [27]	Makespan	×	✓	×	✓	×	×	Heuristic Algorithm
Tao and et al. [58]	Makespan	×	✓	×	×	×	×	Artificial Algae Algorithm (AAA)
Tao and Dong [57]	Makespan	×	×	✓	×	×	×	SA algorithm
Tao and Dong [59]	Makespan (Total Cost)	×	×	✓	×	✓	×	Hybrid Tabu Search NSGA2
Chen and et al. [8]	Makespan	×	✓	×	×	×	×	Priority rule-based heuristics
Creemers [12]	Makespan	×	✓	×	×	×	✓	Exact Procedure
Ma et al. [39]	Robustness	✓	✓	×	×	×	×	Tabu Search
This paper	Makespan (resource cost)	✓	✓	×	×	×	×	SPEA-II, NSGA-II, MOPSO, PESA-II

2 Mathematical model descriptions

The resource availability cost problem (RACP) in a PERT-type network can be concerned with n activities $j=1, 2, \ldots, n$, in which Nodes 1 and n, initial and terminal nodes respectively, are considered to be dummies. Consequently, the start and end nodes have zero duration and zero resource consumption. Activities are represented on activity on node (AON) network. In addition, both activities durations d_j and resource requirements r_{jk}, in which $k=1, 2, \ldots,\rho$, are independent continuous random numbers with given distribution functions. The precedence relations of activities are assumed to be finished to start with zero time lags. Moreover, each activity j has a set of predecessor P_j and can be started when all of its predecessors are terminated. Each activity has one execution mode. Remark that each resource k has a fixed resource cost of C_k for each unit of available capacity. We have two objective functions. The first one is to schedule activities such that the completion time of the project is minimized; however, the second one is to minimize the total cost of the resource capacities considering precedence and resource constraints. According to the objective functions, the problem has two decision variables. Including x_{jt} and R_k. Let $x_{jt}=1$, if activity j is finished at time t and 0 otherwise. Furthermore, activity j can be finished at a time between the earliest finish time (EF_j) and latest finish time (LF_j). The mathematical model is as follow:

$$MinZ_1 = E\left[\sum_{t=EF_n}^{T} t.x_{nt}\right] \tag{1}$$

$$MinZ_2 = E\left[\sum_{k=1}^{\rho} C_k.R_k\right] \tag{2}$$

S.T.

$$\sum_{t=1}^{T} x_{jt} = 1; \quad j = 1, \ldots, n, t = 1, \ldots, T \tag{3}$$

$$\sum_{t=EF_i}^{T} \left(t + d_i\right) \cdot x_{it} \le \sum_{t=EF_j}^{T} t \cdot x_{jt}; \quad j = 1, \ldots, n, i \in p_j \tag{4}$$

$$\sum_{j=2}^{n-1} r_{jk} \sum_{q=t}^{t+d_j-1} x_{jq} \le R_K; \quad k = 1, \ldots, \rho, t = 1, \ldots, T \tag{5}$$

$$x_{jt} \in \{0, 1\}; \quad j = 1, \ldots, n, t = 1, \ldots, T \tag{6}$$

$$R_K \ge 0; \quad K = 1, \ldots, \rho \tag{7}$$

Where the decision variables are:

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

R_K The Resource level for a resource type

The first objective function (1) represents the expected value of the project makespan. However, the second objective function (2) denotes the expected total cost of the resource capacities. Constraint (3) assures that each activity can only be finished in one time period. Constraints (4) and (5) illustrate the precedence and resource constraints, respectively. Finally, Constraints (6) and (7) determine that the decision variables are binary and positive integer variables, respectively.

3 Solution approaches

In this section, four well-known multi-objective algorithms, i.e., SPEA-II, PESA-II, NSGA-II, and MOPSO, which have been widely applied to many NP-hard problems, as well as employed operators are discussed in the ensuing sub-sections.

3.1 Common characteristics of algorithms

This section denotes common elements of our algorithms, including solution representation, generating a feasible solution, and applying our algorithms to the PERT network.

3.1.1 Solution representation

Designing a convenient solution representation is one of the key factors of the process of solving any problem. Also, for each solution in the original space, there is a unique solution in the encoded space and each encoded solution pertains to one feasible solution in the original space [47]. According to the model, the solution representation for meta-heuristic algorithms consists of two parts: the first part is an activity sequence, which has been proposed by Kolisch and Hartmann [31] as an adequate representation, and the second part represents a list of available resource capacities. The chromosome structure for a solution I is demonstrated in Fig. 1.

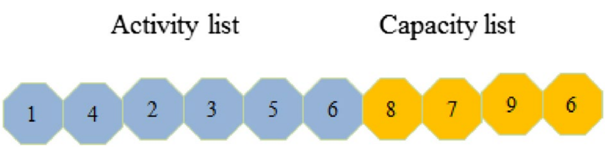


Fig. 1 Chromosome structure

3.1.2 Generating a feasible solution

Since Kelley [30] introduced a schedule generation scheme, several heuristic methods have been proposed [33]. The scheduling generation scheme constructed a feasible schedule by assigning the start times to the activities. There are two different schemes to decode the solution: the serial schedule generation scheme (*SSGS*) and the parallel schedule generation scheme (*PSGS*).

In the RCPSP cases, the set of schedules, which is generated through the *SSGS* or the *PSGS,* have different properties [34]. In this Paper, the serial-*SGS* is employed to decode a solution. *SSGS* includes several stages in which the activity with the highest priority is chosen and assigned the earliest possible starting time (*ESS*) if the activity does not violate both the precedence relation and resource level. In order to build a feasible capacity list, a number for each employed resource between a defined the lower and upper bound should be chosen. The lower and upper bound is calculated via Eqs. (8) and (9).

$$\underline{R_k} = Max\left\{ \sum_{i=1}^{n} \frac{r_{ik} \cdot d_i}{T}, \max_{i=1,\dots,n} \{r_{ik}\} \right\} \tag{8}$$

$$\overline{R_k} = \sum_{i=1}^{n} r_{ik} \tag{9}$$

Each solution or individual of the MOEAs posses a fitness value. Owing to the fact that the problem has several constraints, a randomly generated solution might be infeasible. Note that an infeasible solution may either has an activity started before its predecessors that have been finished or the resource requirements in any time periods are greater than the maximum level. In this regard, the technique called Repair, which is explained later, is employed to resolve the issue.

3.1.3 Applying meta-heuristic algorithms in *PERT*-type network

The solution techniques, which are available for resource-constrained project scheduling with stochastic activity durations, are very restricted. Owing to computational complexity in the uncertainty, optimal solution or heuristics for scheduling have been found useful for large-deterministic problems, and they are not appropriate. In this regard, various methods are developed. One of the renowned procedures, which are employed in the stochastic project scheduling environment, is Monte Carlo simulation (*MCS*). This method has become more practical when it is difficult or impossible to use mathematical methods. In this method, the random numbers are generated as the activity completion time. Then, the time of the longest path is determined as the project completion time. This procedure is repeated for the number we want the network to be simulated [19].

3.2 Strength pareto evolution algorithm (*SPEA-II*)

SPEA-II is one of the efficient algorithms in the field of multi-objective optimization (MOO). This algorithm is based on the domination concept and forming a Pareto Front. SPEA-II is found by [66]. They tried to improve the performance of SPEA and overcome the potential weakness. The overall pseudo code of the SPEA-II is explained in Fig. 2.

3.3 Non-dominated sorting genetic algorithm (*NSGA-II*)

Widely used in the literature, NSGA-II is considered as one of the well-known multi-objective evolutionary algorithms (MOEA's), developed by [13]. Moreover, NSGA-II has ensured a high resolving capacity for multi-objective combinatorial optimization problems. The structure of NSGA-II is given in Fig. 3.

3.4 Multi-objective particle swarm optimization (*MOPSO*)

MOPSO is inspired by the PSO algorithm to solve multi-objective problems. This method is motivated by the simulation of social behavior. In order to determine the movement, each individual utilizes two pieces of information. The first one is their own experience, i.e., they have tried the different alternatives and find out the best state so far. The second one is others' experiences; that is, they have utilized other individuals' information.Therefore, each individual makes his decision regarding both his own experiences and others' experiences [10]. Figure 4 shows the pseudo-code of the MOPSO algorithm.

Start
Initialization: Generate an initial population $P_0 = \emptyset$
Archive = { }
$t = 0$
While $t < T$
Fitness assignment: Calculate fitness values of individuals in P_t and $\bar{P}_t$
Determine Domination: select all non-dominated individuals in P_t and $\bar{P}_t$ to $\bar{P}_{t+1}$
If $|\bar{P}_{t+1}| > \bar{N}$
Truncate extra individuals
Else If $|\bar{P}_{t+1}| < \bar{N}$
Fill the archive with dominated individuals
End If
Apply Crossover, Mutation, Local Search and Movement operators to the mating pool
Set $\bar{P}_{t+1}$ inorder to fill the mating pool
$t = t + 1$
End While
Report A: which is the non-dominated set in the external archive
End

Fig. 2 Pseudo code of the SPEA-II

```
Begin
Initialize new Population
  Repeat
  T= fast non-dominated sorting (new Population)
  {fill the parent population}
    While | parentPop | < N
   T = crowding distance assignment (T)
   parentPopulation = parentPopulation + T
    end While
  Sort ( parentPopulation)
  ParentPopulation = first N element in parentPopulation
  {use selection, crossover, mutation, local search and movement to create a new child generation}
  ChildPopulation = generanteNewPopulation (ChildPopulation)
  NewPopulation = ParentPopulation ∪ ChildPopulation
  t = t + 1
  until (t ≥ nemberOfIterations)
```

Fig. 3 Pseudo code of the NSGA-II

```
Begin
  Initialize swarm
  External archive = { }
  Initialize leaders in an external archive
  Quality (leaders)
  iter = 0
  while iter < maxiter
      For each particle
          Select leader
          Update Position
          Mutation
          Evalution
          Update pbest
      End For
  Update leader in the external archive
  Quality (leader)
  iter = iter + 1
  End While
  Report results in the external archive
End
```

Fig. 4 Pseudo code of the MOPSO

3.5 Pareto envelope based selection (*PESA-II*)

PESA-II is one of the well-known algorithms in the multi-objective optimization area. This algorithm uses a grid-based selection strategy instead of assigning a selective fitness to an individual. Using Deb's test suite of 'T' functions with varying properties, the performance of this algorithm is proved [11]. The overall structure of the algorithm is depicted in Fig. 5.

```
Begin
Initialize newPopulation
Archive = { }
Evaluate newPopulation
Create Grid
it = 0
While it < maxit
      Archive = Archive + newPopulation
      Archive = Archive (non-dominated members)
      If |Archive| > N
          Delete extra members
      End If
      Update grid
      {use Mutation, Crossover, Local Search, Movement operators to create newPopulation}
      Evaluate newPopulation
      it = it + 1
 End While
    Report results in the external archive
End
```

Fig. 5 Pseudo code of the PESA-II

3.6 Mutation

The mutation is an operator that only applied to the activity list. In this article, we defined three different operators that change the activities sequence order, but only one of them, which is selected randomly, will be applied to the chosen chromosome. It should be mentioned that capacity list for the new chromosome will be obtained through the selected member. An example of mutation operators is illustrated in Fig. 6. Also, the employed structure of the swap, insertion, and reversion operators are described, respectively.

3.6.1 Swap operator

In this operator, we initially choose two numbers, *a* and *b*, randomly from the interval [2 *n*-1]. The numbers are selected activities. We consider the smaller number *a*. Note that the initial and terminal node cannot be selected. Let individual

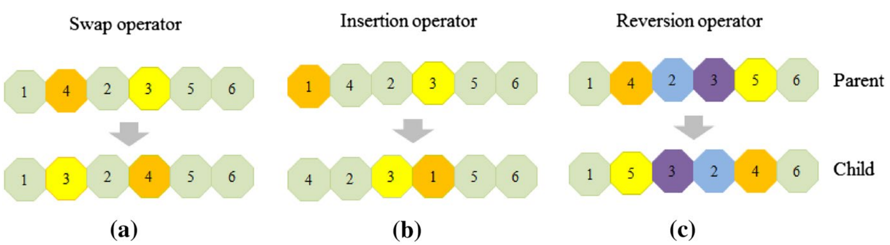


Fig. 6 Mutation operator for activity list

$I = \left(\left(j_1^I, \ldots, j_n^I \right), \left(R_1^I, \ldots, R_\rho^I \right) \right)$ be the selected chromosome for mutation. For $O_a < O_b$, *i.e.,* activity a precedes activity b, the activity list of I is replaced by $\left(j_1^I, \ldots, j_{a-1}^I, j_b^I, j_{a+1}^I, \ldots, j_{b-1}^I, j_a^I, j_{b+1}^I, \ldots, j_n^I \right)$. An example of a swap operator is shown in Fig. 6a. In this example, a and b are 4 and 3, respectively.

3.6.2 Insertion operator

Like Swap operator, *Let a* and *b* two randomly selected numbers from the interval [2 n-1]. The numbers are the selected activity and their new place, respectively. Therefore, let the activity list of the selected chromosome be $\left(j_1^I, \ldots, j_{a-1}^I, j_a^I, j_{a+1}^I, \ldots, j_{b-1}^I, j_b^I, j_{b+1}^I, \ldots, j_n^I \right)$. After applying insertion, activity list will be $\left(j_1^I, \ldots, j_{a-1}^I, j_{a+1}^I, \ldots, j_{b-1}^I, j_a^I, j_b^I, j_{b+1}^I, \ldots, j_n^I \right)$. Fig. 6b demonstrates an example in which a and b are 1 and 3, respectively.

3.6.3 Reversion operator

This operator will select two activities from the activity list and reverses the sequence of the activities between them. Let chromosome $I = \left(j_1^I, \ldots, j_{a-1}^I, j_a^I, j_{a+1}^I, \ldots, j_{b-1}^I, j_b^I, j_{b+1}^I, \ldots, j_n^I \right)$ as the selected member and a and b as the selected activities. After applying reversion, the obtained activity list will be $I_{new} = \left(j_1^I, \ldots, j_{a-1}^I, j_b^I, j_{b-1}^I, \ldots, j_{a+1}^I, j_a^I, j_{b+1}^I, \ldots, j_n^I \right)$. In Fig. 6c, an example, considering a and b are 4 and 5, respectively, are shown.

3.7 Crossover

Crossover is also applied to the activity list. We employed two permutation-based crossover operators for the activity list of the chromosome. The first operator crossover, called one-point crossover, selects an integer number r randomly from the interval [2 n-1]. Note that the initial and terminal nodes are dummy activities. Let $P_1 = \left(\left(j_1^1, \ldots, j_n^1 \right), \left(R_1^1, \ldots, R_\rho^1 \right) \right)$ and $P_2 = \left(\left(j_1^2, \ldots, j_n^2 \right), \left(R_1^2, \ldots, R_\rho^2 \right) \right)$ be selected parents. Two children C_1 and C_2 are defined through the crossover whose activity lists are $C_1 = \left(j_1^{c_1}, \ldots, j_r^{c_1}, j_{r+1}^{c_2}, \ldots, j_n^{c_2} \right)$ and $C_2 = \left(j_1^{c_2}, \ldots, j_r^{c_2}, j_{r+1}^{c_1}, \ldots, j_n^{c_1} \right)$ respectively. $\left(j_1^{c_1}, \ldots, j_r^{c_1} \right) = \left(j_1^1, \ldots, j_r^1 \right)$ and $\left(j_{r+1}^{c_1}, \ldots, j_n^{c_1} \right) = \left(j_{r+1}^2, \ldots, j_n^2 \right)$ where $j_b^2 \notin \left\{ j_1^{c_2}, \ldots, j_r^{c_2} \right\}$ Moreover, $\left(j_1^{c_2}, \ldots, j_r^{c_2} \right) = \left(j_1^2, \ldots, j_r^2 \right)$ and $\left(j_{r+1}^{c_2}, \ldots, j_n^{c_2} \right) = \left(j_{r+1}^1, \ldots, j_n^1 \right)$ where $j_b^1 \notin \left\{ j_1^{c_1}, \ldots, j_r^{c_1} \right\}$. Figure 7a shows an example of this operator.

The second crossover operator is a two-point crossover, in which two integer numbers, r_1 and r_2, $r_1 < r_2$ are generated from the interval [2 n−1], which is called cutting point. Two children called C_1 and C_2 are defined by this crossover. Their activity lists are $C_1 = \left(j_1^{c_1}, \ldots, j_{r_1}^{c_1}, j_{r+1}^{c_1}, \ldots, j_{r_2}^{c_1}, j_{r_2+1}^{c_1}, \ldots, j_n^{c_1} \right)$ and $C_2 = \left(j_1^{c_2}, \ldots, j_{r_1}^{c_2}, j_{r+1}^{c_2}, \ldots, j_{r_2}^{c_2}, j_{r_2+1}^{c_2}, \ldots, j_n^{c_2} \right)$ respectively. Thus, $\left(j_1^{c_1}, \ldots, j_r^{c_1} \right) = \left(j_1^1, \ldots, j_r^1 \right)$ and $j_a^{c_1}, a = r_1 + 1, \ldots, r_2$ is j_2^b where b is the lowest index such that $j_b^2 \notin \left\{ j_1^{c_1}, \ldots, j_{a-1}^{c_1} \right\}$ and $j_a^{c_1}, a = r_2 + 1, \ldots, n$ is j_1^b where b is the

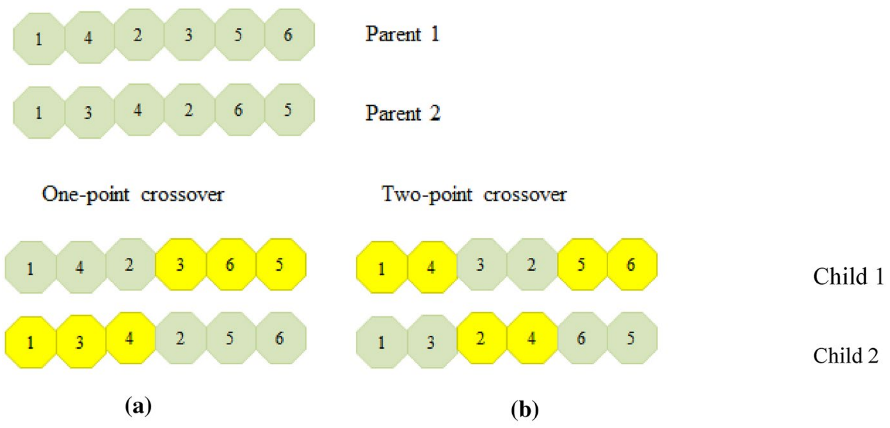


Fig. 7 Crossover operator for activity list

lowest index such that $j_b^1 \notin \left\{ j_1^{c_1}, \ldots, j_{a-1}^{c_1} \right\}$ The definition of C_2 is similar to C_1. An example is illustrated in Fig. 7b.

3.8 Local search

This operator is exerted to the second part, namely the capacity list, of a chromosome. This operator consists of one-point and multi-point operators. Remark that one of the operators is selected randomly and employed. Figure 8 illustrates an example of both a one-point and multi-point local search.

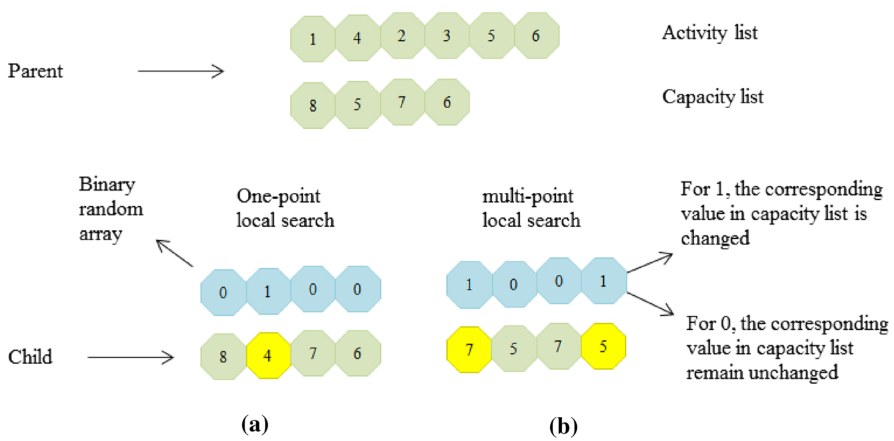


Fig. 8 Local Search operator for a capacity list

Begin

it = 0

While *it< Maximum number of running*
Determine solution I
Specify resource type k: $R_k = \{1, \ldots, \rho\}$
Determine P_k^I, a set of all intervals at which resource type K is at maximum level, MR_k^I,
Set all significant activities (ASA_k) which is executed during time intervals contained in P_k^I
Choose an activity j from ASA_k randomly
Establish a set S^j including all activities which started with or before the activity j
Select activities from S^j which is not the immediate successor of activity j and as right shift them as possible
Put the activity j right after them
Let $R_r^I = R_r^I$, $r = 1, \ldots, \rho; r \neq k, \textit{and set } R_r^I = R_r^I - 1$
Calculate the start time of activities using the serial scheduling scheme
it=it+1
End while

Sort the solutions via domination criteria (Rank, Crowding distance)
Select the best solution in terms of domination concepts

If |number of best solution|>1
Select one randomly
End if

End

Fig. 9 Pseudo code of the movement

Table 2 Precedence relationship of example

Nodes	Prerequisite activities
1	[]
2	1
3	1
4	1
5	2
6	[3,4]
7	[4,6]
8	[4,5]
9	[7,8]

3.9 Movement

This operator is designed to minimize both objective functions simultaneously (Fig. 9). To do so, this operator alters both parts of the solution: activity list, and capacity list. Initially, a solution and resource type *K* are chosen randomly.

It is noticeable that by applying all the mentioned operators, to the chromosome, the solutions might be infeasible in terms of precedence constraints. Therefore, a function called repair function is used to make the chromosome feasible. To elucidate the issue, an example is provided to show how this method works. Table 2 illustrates the activities and their related prerequisite activities. Since the relation between activities in this paper is FS(0), an activity can only be started if all of its

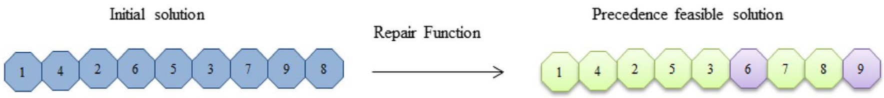


Fig. 10 An example of a repair function

Table 3 Test problem classification

Problem groups	File name at PSPLIB	# problems	Size of the problems	# non-dummy activities	# renewable resources
1	J1059.m	10	Small	10	2
2	J1062.m	10	Small	10	3
3	J1064.m	10	Small	10	4
4	J2059.m	10	Medium	20	2
5	J2060.m	10	Medium	20	3
6	J2064.m	10	Medium	20	4
7	J3045.m	10	Large	30	2
8	J3047.m	10	Large	30	3
9	J3048.m	10	Large	30	4

prerequisite activities have been fulfilled. Figure 10 clarifies the repair function by which a solution changed to a feasible one. Accordingly, the precedence relationships of all the activities are monitored. If its prerequisite activities are not finished, the activity shifted to place forward and started at the earliest possible time. Considering Fig. 10, two activities that are not in the right place have been moved after their precedence activities. This function is applied to all newly generated solution before their evaluating its fitness. As a result, all the solutions generated will be feasible.

4 Computational experiments

In this section, the performance of the proposed four multi-objective algorithms, namely PESA-II, NSGA-II, and MOPSO, are compared. Note that the algorithms studied in this paper are coded using MATLAB 2014a.

4.1 The test problems

Since the presented mathematical model is a newly defined problem in some aspects, we redefine a set of 90 standard problems categorized into three different groups, small, medium, and hard, from PSPLIB. More details about the problems are provided in Table 3.

These standard test problems containing the activities and predecessor relations between the activities are chosen as our fundamental test problems. Also, some new data are required for our problems, according to its mathematical model, which are produced and described as below:

- The number of (non-dummy) activities in different groups is 10, 20, or 30.
- The activity duration is stochastic and is randomly produced by uniform U(1,10).
- Resource requirement r_{ik} is also considered as a stochastic parameter randomly produced by uniform U(1,10).
- The maximal number of predecessors and successors for each activity is equal to three.
- The network complexity (NC) coefficient is assumed to be three.
- Resource Factor (RF) is considered to be 1.5.
- The number of renewable resources considering different groups varies from 2 to 4.
- The average cost of each resource level C_k is supposed to be equal.
- The number of initial and terminal activities is equal to three.

4.2 Comparison criteria for algorithms evaluation

In this paper, six comparison criteria are applied to evaluate the performance of the algorithms. For more information about criteria, refer to Table 4.

4.3 Parameters tuning

It is obvious that the various levels of the parameters affect the quality of the solutions obtained by the hybrid algorithms. Thereby, selecting the best combination of parameters can augment the search process to find more suitable solution, and prevent being trapped in a local optimum. There are many techniques for designing an

Table 4 Performance criteria

Metris	Criteria calculation	Brief description
CPU-time (↓)	–	This criterion shows elapsed time
NPS (↑)	–	This criterion illustrates the number of solutions in pareto fronts
MID [49] (↓)	$MID = \frac{\sum_{i=1}^{n} c_i}{n}$	This criterion calculates total nearness of solutions from the ideal solution
Spacing [53] (↓)	$S = \sqrt{\frac{1}{n-1} \times \sum_{i=1}^{n} (d_i - \bar{d})^2}$	is defined to measure the closeness of solution within Pareto Front
Diversity [65] (↑)	$D = \sqrt{\sum_{m=1}^{M} \left(\max_{i=1:\lvert Q\rvert} f_m^i - \min_{i=1:\lvert Q\rvert} f_m^i\right)^2}$	This measure defines the extension of solutions
Simultaneous Metrics (SM) [48] (↓)	$SM = \frac{MID}{D}$	This measure two well-known criteria at the same time

experimental investigation. Although a full factorial experiment is most appropriate used method, the investigation becomes more complicated when the number of factors and their decided levels is significantly increased. To overcome this defect, fractional factorial experiments are used to diminish the number of required tests [9].

In this regard, the Taguchi method is utilized to set the parameters of the presented algorithms. This method that is designed based on orthogonal arrays can be used efficiently as an alternative for the full factorial experimental design to investigate a group of factors. These factors are divided into two groups: controllable noise factors and noise factors. The method initial goal is to select the best level of the factors such that the effect of controllable factors is maximized and the effect of noise factors is minimized [56]. Hence, a measure called signal to noise ratio (S/N) is employed to evaluate the algorithms' performance. The value is calculated through Eq. (10):

$$S/N\,Ratio = -10\log\frac{1}{n}\left(S\left(Y^2\right)\right) \tag{10}$$

where n and Y are the number of orthogonal arrays and the response value, respectively. SM criterion is the most crucial criterion among the mentioned criteria due to the fact that it considers two critical criteria, MID and D, simultaneously, SM is applied for tuning the parameters. Consequently, the response factor is calculated through the Eq. (11);

$$SM = MID/D \tag{11}$$

where MID and D are considered to assess convergence and diversity, respectively. For each algorithm, three levels of parameters are shown in Table 5. Using the Minitab software, the orthogonal arrays are obtained.

As we mentioned before, we divide the test problems into small, medium, and large size problems. In this paper, the Taguchi method is applied to all scales for parameter tuning. To do so, for each category of problem, one problem is randomly selected. To yield more reliable results, each problem is tackled five times. The best result among the 5-time runs of each problem is considered the result of that problem.

Parameter tuning by the Taguchi method is explained in detail by representing the step by step results for small-size problems. The result for each level is represented in Figs. 11 and 12. Accordingly, the optimal levels of factors are represented in Table 6. The orthogonal arrays of these designs along with the all experimental results are represented in ("Appendix 1"). Furthermore, the delta value represented in Table 7, the Archive size has the most influence on the SPEA-II. P-movement and P-local search operators are the other practical factors on SPEA-II, respectively. Therefore, movement and local search operators have an impact on SPEA-II.

Table 5 Algorithm parameter ranges along with their levels

Alg.	Parameters	Symbol	Parameter level		
			Level 1	Level 2	Level 3
SPEA-II	Pop size	A	40	45	50
	Archive size	B	25	30	35
	P-crossover	C	0.7	0.8	0.9
	P-mutation	D	0.1	0.2	0.3
	P-local search	E	0.4	0.5	0.6
	P-movement	F	0.4	0.5	0.6
	Max iteration	G	100	200	300
PESA-II	Pop size	A	40	45	50
	Archive size	B	25	30	35
	P-crossover	C	0.7	0.8	0.9
	P-mutation	D	0.1	0.2	0.3
	Max iteration	E	100	200	300
	N-Grid	F	5	8	10
NSGA-II	Pop size	A	25	30	35
	P-crossover	B	0.7	0.8	0.9
	P-mutation	C	0.1	0.2	0.3
	Max iteration	D	100	200	300
MOPSO	C1	A	1	1.5	2
	C2	B	1	1.5	2
	W	C	0.7	0.8	0.9
	Pop size	D	40	45	50
	Rep size	E	25	30	35
	N-Grid	F	5	8	10
	Max iteration	G	100	200	300

4.4 The computational results

In this section, the performance of the proposed algorithm is evaluated. Remark that the computational results are presented in ("Appendix 2"). Figure 13 illustrates Box-Plots of each criterion, of the four presented algorithms. Furthermore, Fig. 14 shows the results of the problems for different criteria graphically. According to Figs. 13, and 14, in terms of some criteria, it is denoted that the SPEA-II has the best performance; e.g., the obtained result of NPS criterion shows that SPEA-II outperforms other algorithms. Afterward, MOPSO, PESA-II, and NSGA-II are placed, respectively. Moreover, in terms of CPU-time, the SPEA-II, and MOPSO has also obtained the best performance, respectively. However, the results of the PESA-II and NSGA-II are slightly close. However, in terms of other criteria, the outcomes are very close.

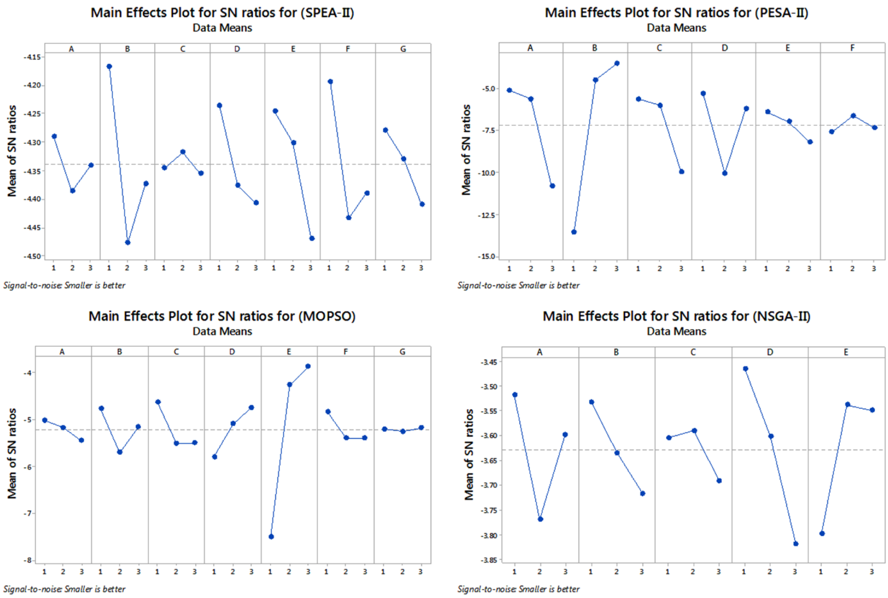


Fig. 11 The S/N ratio plots for each level of the factors (small-size problem)

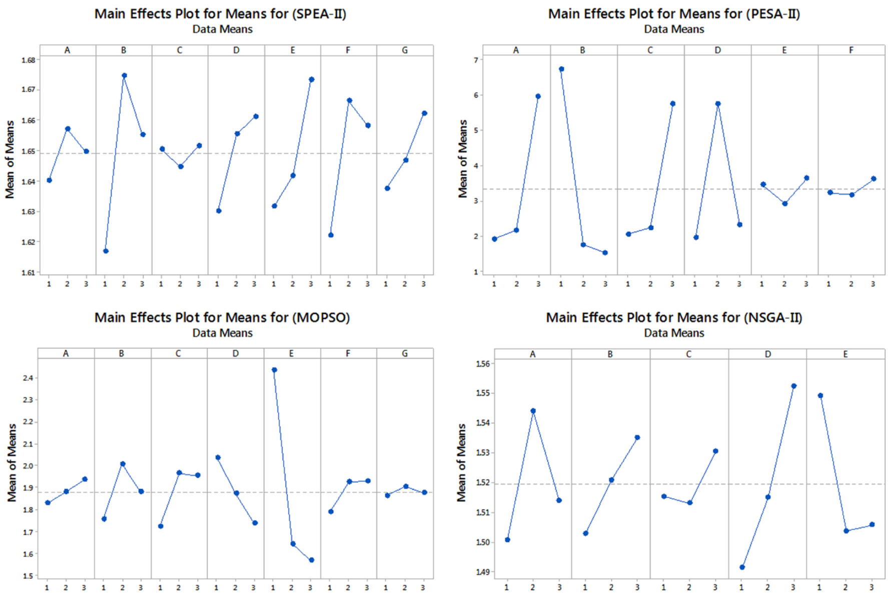


Fig. 12 The mean plots for each level of the factors (small-size problem)

Table 6 Parameters setting values

Alg.	Parameters	Symbol	Selected level		
			Small-size problems	Medium-size problems	Large-size problems
SPEA-II	Pop size	A	1	2	2
	Archive size	B	1	2	3
	P-crossover	C	2	2	3
	P-mutation	D	1	3	3
	P-local search	E	1	3	2
	P-movement	F	1	1	1
	Max iteration	G	1	1	1
PESA-II	Pop size	A	1	1	2
	Archive size	B	3	3	3
	P-crossover	C	1	1	2
	P-mutation	D	1	1	3
	Max iteration	E	1	2	2
	N- Grid	F	2	1	1
NSGA-II	Pop size	A	1	1	1
	P-crossover	B	1	3	2
	P-mutation	C	2	2	2
	Max iteration	D	1	1	1
	P-local search	E	2	1	1
MOPSO	C1	A	1	1	1
	C2	B	1	3	2
	W	C	1	2	1
	Pop size	D	3	2	2
	Rep size	E	3	3	3
	N- Grid	F	1	1	1
	Max iteration	G	1	1	1

Table 7 S/N ratio value for SPEA-II

Response Table for the signal to noise ratio (smaller is better)							
Level	A	B	C	D	E	F	G
1	1.640	1.617	1.650	1.630	1.632	1.622	1.638
2	1.657	1.675	1.645	1.655	1.642	1.666	1.647
3	1.650	1.655	1.652	1.661	1.673	1.658	1.662
Delta	0.017	0.058	0.007	0.031	0.042	0.044	0.025
Rank	6	1	7	4	3	2	5

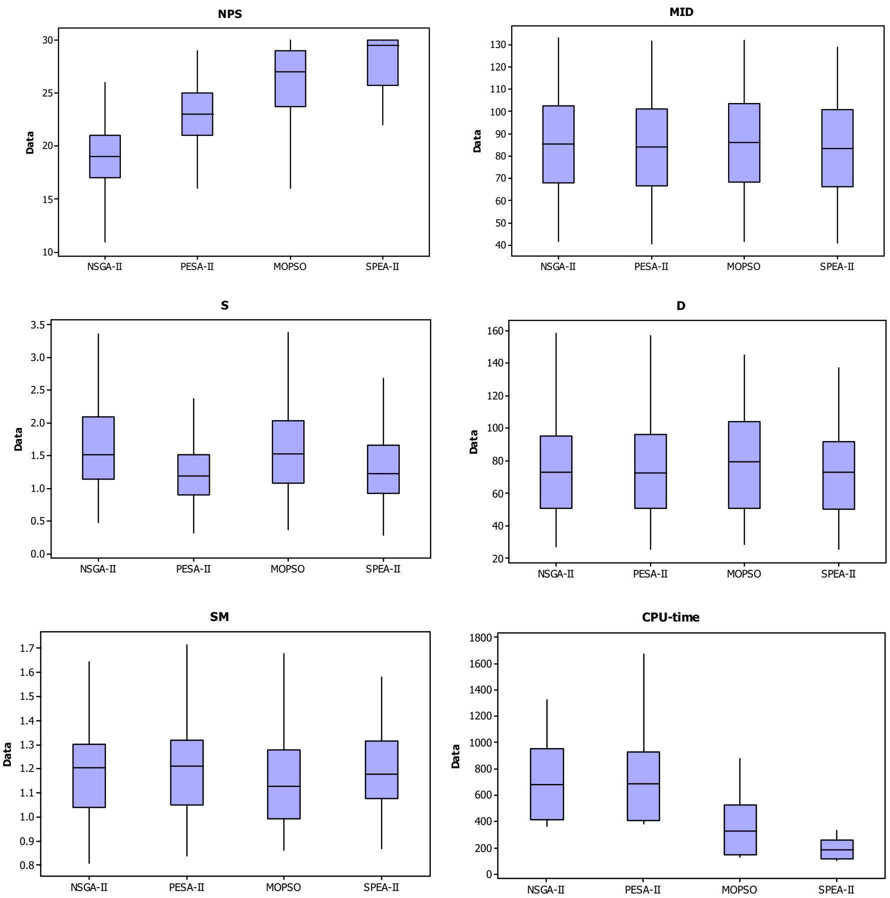


Fig. 13 Box-plot results for statistical comparison

4.5 Sensitivity analysis

Since the results in some of the criteria are very close, and we cannot compare them, in this section, we use the Relative Percentage Deviation (*RPD*). In this method, the obtained results of these performance criteria for each problem are transformed to a Relative Percentage Deviation (*RPD*) that is calculated by Eq. (12):

$$RPD = \left| \frac{Algorithm_{solution} - Best_{solution}}{Best_{solution}} \right| \times 100 \tag{12}$$

where $Algorithm_{solution}$ is the obtained value for each experiment by each performance criteria, $Best_{solution}$ is the best value between the obtained values of four algorithms. Then, the average of the RPD's obtained for problems are calculated. The

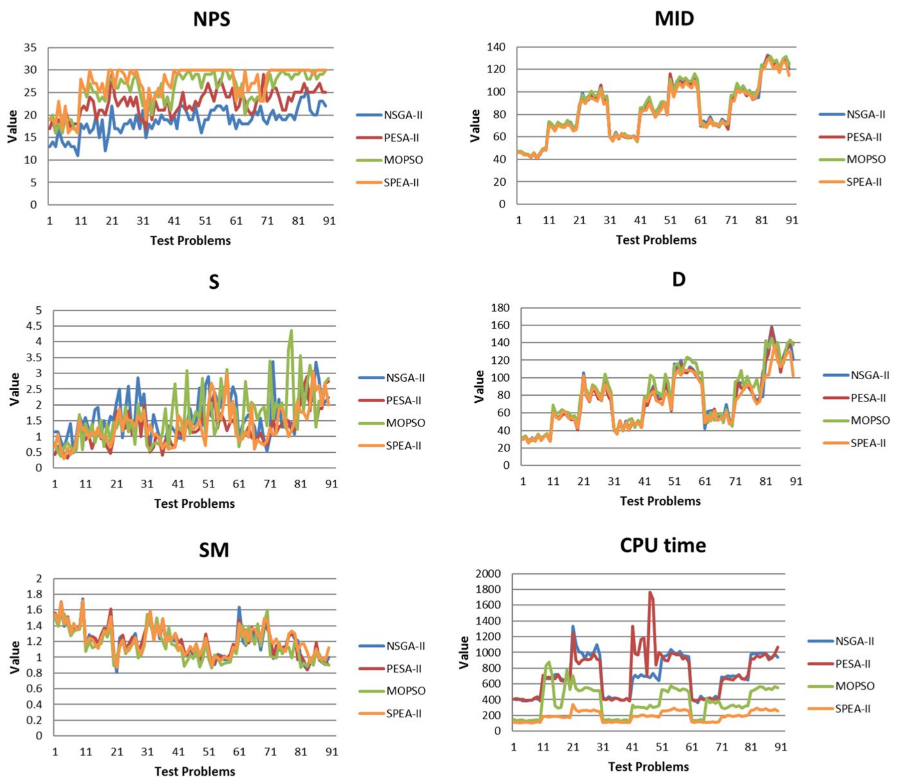


Fig. 14 A detailed comparison of criteria on different test problems

Table 8 Average RPD for criteria on test problems

Alg.	NPS (%)	MID (%)	S (%)	D (%)	SM (%)	CPU-time (%)
NSGA-II	31.92	2.87	66.92	6.44	6.41	266.94
PESA-II	17.39	1.85	21.54	8.27	7.42	285.68
MOPSO	6.94	3.73	61.25	2.20	2.46	91.00
SPEA-II	1.45	0.62	31.66	9.73	8.06	0.00

results are shown in Table 8. Remark that the less value shows the higher performance. Also, the best result in each metric is bolded. Accordingly, SPEA-II has the best performance in NPS, MID, and CPU-time criteria. Furthermore, MOPSO has gain better results in D and SM criteria, and PESA-II is the best in terms of S criterion. Remark that NSGA-II has obtained the worst results.

4.6 TOPSIS approach

Since the algorithms' results are close in some aspects, and each of the defined algorithms has some advantages in some criteria rather than others, we cannot certainly determine which algorithm has the best performance. Hence, in order to investigate the performance more comprehensively, a Multi-Attribute Decision Making (MADM) technique is employed. We apply a renown multi-attribute decision-making method called TOPSIS (a technique for order performance by similarity to ideal solution), which was introduced by Hwang and Yoon [25]. This method can also be integrated with other approaches, e.g., AHP and Fuzzy techniques, to deal with various decision-making problems [5, 46].

TOPSIS is a practical and useful technique for ranking alternatives. This method is derived from the Euclidean distance of each quality performance of the distance between the positive ideal solution and the negative ideal one. TOPSIS considers both positive and negative simultaneously to chose the most suitable alternative: the most preferred alternative should not only have the shortest distance from the positive ideal solution but also have the longest distance from the negative ideal solution. The final score is calculated according to the distance between the positive and negative ideal [62]. The overall process of the TOPSIS method to find the best possible solution is described in Fig. 15.

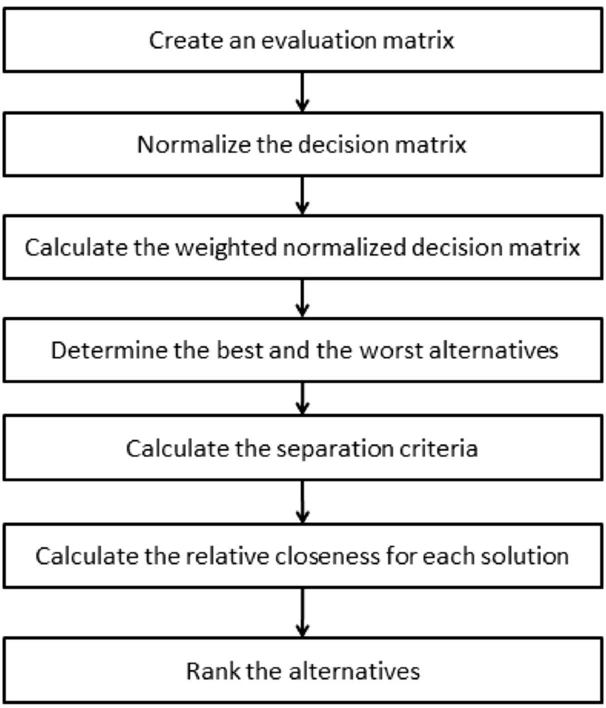
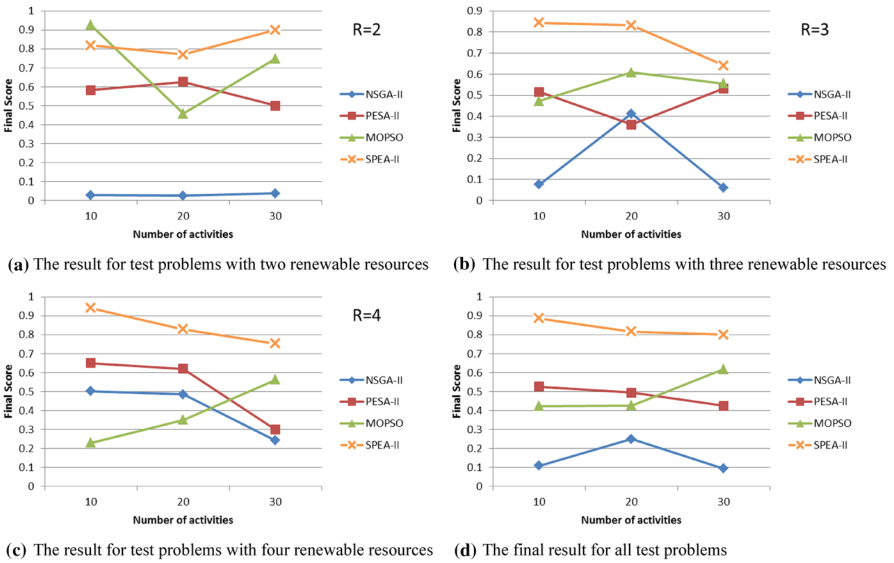


Fig. 15 Flow chart for the TOPSIS method



(a) The result for test problems with two renewable resources **(b)** The result for test problems with three renewable resources

(c) The result for test problems with four renewable resources **(d)** The final result for all test problems

Fig. 16 TOPSIS results

To evaluate the algorithms' performance more precisely, the final score for algorithms are calculated. As it is obvious, the more (less) final score shows a better result. Figure 16 demonstrates the results graphically. Remark that, in Fig. 16, the problems are distinguished by their number of renewable resources and activities. Furthermore, Fig. 16a–c demonstrate the results from all problems with two, three, and four renewable resources, respectively, and Fig. 16d illustrates the final result, considering all test problems.

Accordingly, it is implied that the number of activities has no impact on the efficiency of SPEA-II, and SPEA-II has gained the best result in almost all modes. Meanwhile, MOPSO is sensitive to the number of activities and has obtained the most relevant result in all problems with 30 activities. Moreover, NSGA-II is sensitive to the number of resources. As a result, by increasing the number of resources, NSGA-II has a better performance. However, it is clear that by increasing the number of activities, MOPSO has better performance. As a result, it has the best performance in problems with 30 activities.

In addition, the final result is demonstrated in Table 9. Accordingly, NSGA-II with 0.8975 values score has the worst result. In contrast, SPEA-II has gained a 0.8157 value score and be in the first place. Moreover, after SPEA-II, MOPSO, and PESA-II with 0.4766 and 0.4353 value score are placed, respectively.

Table 9 TOPSIS final result

Alg.	Final score	Rank
NSGA-II	0.0829	4
PESA-II	0.4353	3
MOPSO	0.4766	2
SPEA-II	0.8157	1

5 Conclusions and future research

In this paper, a bi-objective resource availability cost problem with stochastic activity durations and resource requirement are considered. Furthermore, in order to consider uncertainty in the model, a PERT-type network, where activities require a random amount of resources of various types with random duration, is considered. The problem has two objectives, in which the first one is to minimize the regular criterion namely project's makespan, and the second one is to minimize the total resource cost. Since the problem is NP-hard in the strong sense, meta-heuristic algorithms are presented. To do so, four meta-heuristic algorithms, namely SPEA-II, PESA-II, MOPSO, and NSGA-II, are employed to solve the problem. The parameters of these algorithms are tuned by the Taguchi method, and finally, six performance criteria are used to analyze the diversity and convergence of proposed algorithms. Results for project completion time are provided from Monte Carlo simulation (MCS) runs. The performance of the algorithms is tested on the redefined problem from PSPLIB, including different sizes. Moreover, to investigate the performance of the algorithms more comprehensively, a MADM technique called TOPSIS and RPD method are applied.

According to the obtained results, in terms of NPS and CPU-time criteria, SPEA-II has acquired the best performance. Furthermore, Average RPD for criteria on all test problems has shown that PESA-II has relatively best performance considering S criterion with an average 21.54 percent deviation. Regarding D and SM criteria, MOPSO with average less deviation has represented the best performance. Considering Fig. 16d, it is noteworthy that MOPSO, in contrast to PESA-II, has shown better performance by increasing the complexity of the problem. It is also proved that SPEA-II has the best performance in all types of problems. According to Table 7, it has been determined that movement and local search, after the archive size, have the most impact on the performance of the SPEA-II, respectively.

Some extensions of this research as a future study might be of interest. We can consider multiple execution modes for each activity, considering the required resource and activity duration. We can also consider preemption in the model. Finally, applying other solution approaches to this model would be proper research as a future study.

Appendix 1: Tuning the algorithms' parameters

See Tables 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20 and 21.

Table 10 Computational results to tune SPEA-II for small-size problem

	A	B	C	D	E	F	G	R1	R2	R3	R4	R5
1	1	1	1	1	1	1	1	1.507	1.624	1.492	1.518	1.605
2	1	1	1	1	2	2	2	1.774	1.584	1.750	1.693	1.661
3	1	1	1	1	3	3	3	1.868	1.696	1.938	1.975	1.773
4	1	2	2	2	1	1	1	1.645	1.714	1.700	1.684	1.798
5	1	2	2	2	2	2	2	1.679	1.682	1.729	1.698	1.709
6	1	2	2	2	3	3	3	1.698	1.699	1.680	1.702	1.739
7	1	3	3	3	1	1	1	1.617	1.700	1.701	1.699	1.712
8	1	3	3	3	2	2	2	1.698	1.720	1.700	1.688	1.682
9	1	3	3	3	3	3	3	1.698	1.698	1.685	1.698	1.698
10	2	1	2	3	1	2	3	1.946	1.659	1.727	1.796	1.966
11	2	1	2	3	2	3	1	1.632	1.677	1.812	1.946	1.746
12	2	1	2	3	3	1	2	1.959	1.613	1.925	1.677	1.608
13	2	2	3	1	1	2	3	1.700	1.698	1.694	1.698	1.832
14	2	2	3	1	2	3	1	1.743	1.710	1.635	1.700	1.708
15	2	2	3	1	3	1	2	1.709	1.681	1.699	1.698	1.671
16	2	3	1	2	1	2	3	1.659	1.670	1.700	1.698	1.700
17	2	3	1	2	2	3	1	1.700	1.701	1.702	1.677	1.702
18	2	3	1	2	3	1	2	1.679	1.699	1.682	1.698	1.699
19	3	1	3	2	1	3	2	1.777	1.838	1.845	2.009	1.550
20	3	1	3	2	2	1	3	1.776	1.804	2.099	2.054	1.669
21	3	1	3	2	3	2	1	2.030	1.811	1.735	1.661	1.800
22	3	2	1	3	1	3	2	1.671	1.676	1.701	1.688	1.693
23	3	2	1	3	2	1	3	1.698	1.713	1.697	1.707	1.706
24	3	2	1	3	3	2	1	1.699	1.702	1.705	1.705	1.700
25	3	3	2	1	1	3	2	1.698	1.698	1.764	1.703	1.701
26	3	3	2	1	2	1	3	1.681	1.521	1.700	1.689	1.702
27	3	3	2	1	3	2	1	1.712	1.680	1.703	1.702	1.681

Table 11 Computational results to tune PESA-II for small-size problem

	A	B	C	D	E	F	R1	R2	R3	R4	R5
1	1	1	1	1	1	1	4.028	4.105	5.094	3.441	3.460
2	1	1	1	1	2	2	2.592	3.794	3.678	2.444	2.946
3	1	1	1	1	3	3	2.879	2.294	1.932	1.939	2.100
4	1	2	2	2	1	1	1.958	1.470	1.520	1.409	1.554
5	1	2	2	2	2	2	1.360	1.416	2.227	1.447	1.693
6	1	2	2	2	3	3	2.754	2.120	3.071	2.141	2.194
7	1	3	3	3	1	1	1.674	1.317	1.472	1.231	1.285
8	1	3	3	3	2	2	2.289	2.125	1.872	1.936	3.103
9	1	3	3	3	3	3	1.775	1.306	1.350	2.074	1.489
10	2	1	2	3	1	2	3.140	1.829	1.871	2.134	2.680
11	2	1	2	3	2	3	3.238	4.989	5.569	3.315	3.176
12	2	1	2	3	3	1	7.892	5.783	7.789	7.451	5.253
13	2	2	3	1	1	2	1.449	1.592	1.609	1.574	1.270
14	2	2	3	1	2	3	1.478	1.529	1.991	1.881	1.690
15	2	2	3	1	3	1	3.482	1.989	2.041	2.656	2.331
16	2	3	1	2	1	2	1.760	1.297	1.162	1.370	1.234
17	2	3	1	2	2	3	2.486	1.714	1.706	2.693	2.593
18	2	3	1	2	3	1	1.538	1.593	1.818	2.086	1.610
19	3	1	3	2	1	3	6.391	7.429	8.457	9.508	7.399
20	3	1	3	2	2	1	4.054	3.424	5.780	4.225	6.876
21	3	1	3	2	3	2	9.166	8.504	7.164	8.750	4.087
22	3	2	1	3	1	3	1.722	1.721	1.576	2.871	1.535
23	3	2	1	3	2	1	1.949	1.774	1.569	1.407	1.615
24	3	2	1	3	3	2	4.358	5.225	3.197	3.652	5.614
25	3	3	2	1	1	3	2.438	1.802	2.323	2.263	1.787
26	3	3	2	1	2	1	1.805	1.812	2.089	2.679	1.824
27	3	3	2	1	3	2	1.634	1.281	1.594	1.966	1.772

Table 12 Computational results to tune NSGA-II for small-size problem

	A	B	C	D	E	R1	R2	R3	R4	R5
1	1	1	1	1	1	1.495	1.988	1.956	2.211	2.090
2	1	1	1	1	2	2.045	1.732	1.687	1.423	1.433
3	1	1	1	1	3	1.846	1.436	2.028	1.638	1.954
4	1	2	2	2	1	1.709	1.578	2.643	1.880	1.651
5	1	2	2	2	2	1.565	2.656	1.772	1.974	1.482
6	1	2	2	2	3	1.440	1.899	1.833	1.413	1.972
7	1	3	3	3	1	1.579	1.650	2.518	2.660	1.641
8	1	3	3	3	2	1.786	1.803	2.232	2.404	1.508
9	1	3	3	3	3	1.620	1.722	1.646	1.592	2.378
10	2	1	2	3	1	1.570	1.984	1.652	2.326	1.913
11	2	1	2	3	2	2.745	2.101	2.675	1.752	1.577
12	2	1	2	3	3	2.530	1.658	1.529	1.514	1.520
13	2	2	3	1	1	1.685	1.653	2.672	1.571	2.391
14	2	2	3	1	2	1.453	1.713	1.683	1.531	1.794
15	2	2	3	1	3	1.622	2.244	1.560	1.800	1.804
16	2	3	1	2	1	1.575	1.644	1.786	2.035	1.585
17	2	3	1	2	2	1.580	1.950	1.573	1.724	1.635
18	2	3	1	2	3	1.504	1.521	1.652	1.936	1.782
19	3	1	3	2	1	1.893	1.979	1.442	1.478	1.628
20	3	1	3	2	2	1.559	1.809	2.379	1.825	1.541
21	3	1	3	2	3	1.601	2.878	1.528	2.329	2.278
22	3	2	1	3	1	1.784	1.571	1.838	1.985	1.691
23	3	2	1	3	2	1.942	1.828	1.536	1.569	2.042
24	3	2	1	3	3	1.967	1.688	1.860	1.736	1.524
25	3	3	2	1	1	2.123	1.684	1.916	1.561	1.867
26	3	3	2	1	2	1.515	1.741	1.460	1.441	1.550
27	3	3	2	1	3	1.983	2.454	1.482	1.889	1.703

Table 13 Computational results to tune MOPSO for small-size problem

	A	B	C	D	E	F	G	R1	R2	R3	R4	R5
1	1	1	1	1	1	1	1	2.816	3.078	1.947	1.985	2.185
2	1	1	1	1	2	2	2	1.837	1.538	1.917	2.188	1.809
3	1	1	1	1	3	3	3	1.758	1.889	1.617	1.619	1.673
4	1	2	2	2	1	1	1	4.763	4.148	3.349	2.988	4.049
5	1	2	2	2	2	2	2	1.945	1.519	1.486	1.889	1.454
6	1	2	2	2	3	3	3	1.644	1.827	1.675	1.650	2.903
7	1	3	3	3	1	1	1	1.882	2.899	3.154	3.568	2.555
8	1	3	3	3	2	2	2	2.816	3.297	1.983	1.968	2.608
9	1	3	3	3	3	3	3	1.809	1.410	1.809	1.469	1.488
10	2	1	2	3	1	2	3	1.847	2.070	3.139	2.273	1.910
11	2	1	2	3	2	3	1	2.112	2.703	2.203	1.682	1.836
12	2	1	2	3	3	1	2	1.602	2.326	1.627	1.557	1.855
13	2	2	3	1	1	2	3	4.601	3.401	3.256	3.233	3.908
14	2	2	3	1	2	3	1	1.933	1.899	2.707	2.625	2.313
15	2	2	3	1	3	1	2	2.276	1.809	1.565	1.664	1.785
16	2	3	1	2	1	2	3	2.336	3.103	3.431	3.589	2.724
17	2	3	1	2	2	3	1	1.455	1.433	1.554	1.452	1.703
18	2	3	1	2	3	1	2	1.918	2.102	1.776	1.420	1.367
19	3	1	3	2	1	3	2	2.582	4.450	2.724	2.690	2.913
20	3	1	3	2	2	1	3	1.798	1.810	1.748	1.590	1.820
21	3	1	3	2	3	2	1	1.742	2.594	1.454	1.722	2.044
22	3	2	1	3	1	3	2	2.804	2.219	2.153	1.910	2.499
23	3	2	1	3	2	1	3	1.678	1.893	2.081	1.917	1.713
24	3	2	1	3	3	2	1	1.991	1.926	1.723	1.687	1.752
25	3	3	2	1	1	3	2	3.459	3.232	3.883	3.192	3.450
26	3	3	2	1	2	1	3	2.530	1.531	1.616	2.253	1.535
27	3	3	2	1	3	2	1	2.242	1.922	1.864	1.793	1.966

Table 14 Computational results to tune SPEA-II for medium-size problem

	A	B	C	D	E	F	G	R1	R2	R3	R4	R5
1	1	1	1	1	1	1	1	2.114	1.408	1.461	1.840	1.861
2	1	1	1	1	2	2	2	2.344	2.140	3.556	2.481	2.135
3	1	1	1	1	3	3	3	2.563	2.909	1.688	1.561	1.541
4	1	2	2	2	1	1	1	1.862	1.197	1.218	1.343	1.501
5	1	2	2	2	2	2	2	1.288	1.658	2.265	1.640	1.742
6	1	2	2	2	3	3	3	1.815	1.294	1.304	1.349	1.685
7	1	3	3	3	1	1	1	1.508	1.228	1.462	1.621	1.686
8	1	3	3	3	2	2	2	1.298	1.738	1.575	1.379	1.244
9	1	3	3	3	3	3	3	1.582	1.338	2.039	1.248	1.637
10	2	1	2	3	1	2	3	2.293	2.117	2.340	2.149	1.474
11	2	1	2	3	2	3	1	1.270	1.295	1.753	1.214	1.338
12	2	1	2	3	3	1	2	1.509	1.663	1.366	1.307	1.425
13	2	2	3	1	1	2	3	1.373	1.222	1.963	1.466	1.388
14	2	2	3	1	2	3	1	1.245	1.156	1.247	1.504	1.361
15	2	2	3	1	3	1	2	1.631	1.206	1.223	1.486	1.801
16	2	3	1	2	1	2	3	1.939	1.594	1.607	1.264	1.245
17	2	3	1	2	2	3	1	1.586	1.285	1.224	1.422	1.659
18	2	3	1	2	3	1	2	1.239	1.491	2.015	1.328	1.719
19	3	1	3	2	1	3	2	1.514	2.633	1.861	2.708	2.163
20	3	1	3	2	2	1	3	1.524	1.995	1.644	1.527	1.819
21	3	1	3	2	3	2	1	1.350	1.553	1.557	2.007	1.648
22	3	2	1	3	1	3	2	1.669	1.489	1.517	1.938	1.333
23	3	2	1	3	2	1	3	1.484	1.249	1.293	1.592	1.767
24	3	2	1	3	3	2	1	1.209	1.287	1.257	1.212	1.219
25	3	3	2	1	1	3	2	1.519	1.403	1.832	1.301	1.441
26	3	3	2	1	2	1	3	1.365	2.009	1.322	1.263	1.265
27	3	3	2	1	3	2	1	1.867	1.426	1.192	1.260	1.633

Table 15 Computational results to tune PESA-II for medium-size problem

	A	B	C	D	E	F	R1	R2	R3	R4	R5
1	1	1	1	1	1	1	1.759	1.7192	1.735	1.721	1.801
2	1	1	1	1	2	2	1.539	1.4239	1.891	1.441	2.642
3	1	1	1	1	3	3	2.176	1.8677	1.922	2.624	3.032
4	1	2	2	2	1	1	1.296	1.414	1.298	1.717	1.996
5	1	2	2	2	2	2	1.587	2.390	1.696	1.982	1.775
6	1	2	2	2	3	3	1.766	1.212	1.659	1.612	1.938
7	1	3	3	3	1	1	1.713	2.483	1.347	1.373	1.551
8	1	3	3	3	2	2	1.406	1.436	1.520	2.635	1.758
9	1	3	3	3	3	3	1.622	1.456	1.644	1.243	1.240
10	2	1	2	3	1	2	2.899	4.124	2.796	3.029	2.662
11	2	1	2	3	2	3	2.429	2.917	1.917	2.539	2.483
12	2	1	2	3	3	1	1.962	1.602	2.588	1.639	1.620
13	2	2	3	1	1	2	1.481	1.159	1.280	1.121	1.748
14	2	2	3	1	2	3	1.200	1.156	1.326	1.348	1.742
15	2	2	3	1	3	1	1.718	1.929	1.677	1.695	2.590
16	2	3	1	2	1	2	1.453	1.444	1.352	1.728	1.375
17	2	3	1	2	2	3	1.390	1.248	1.129	2.023	1.201
18	2	3	1	2	3	1	1.434	1.320	1.137	1.278	1.131
19	3	1	3	2	1	3	10.12	10.494	10.60	7.760	6.769
20	3	1	3	2	2	1	2.112	2.257	2.303	2.514	2.318
21	3	1	3	2	3	2	4.912	4.510	5.884	5.380	5.722
22	3	2	1	3	1	3	1.679	1.654	1.560	2.799	1.576
23	3	2	1	3	2	1	1.669	2.020	1.643	2.153	2.393
24	3	2	1	3	3	2	1.359	1.861	1.450	1.627	1.788
25	3	3	2	1	1	3	1.287	1.312	1.197	1.172	1.294
26	3	3	2	1	2	1	2.232	1.499	1.956	1.479	1.480
27	3	3	2	1	3	2	1.472	1.523	1.474	1.394	1.805

Table 16 Computational results to tune NSGA-II for medium-size problem

	A	B	C	D	E	R1	R2	R3	R4	R5
1	1	1	1	1	1	1.110	1.055	1.071	1.223	1.069
2	1	1	1	1	2	1.222	1.448	1.102	1.615	1.103
3	1	1	1	1	3	1.122	1.124	1.488	1.343	1.535
4	1	2	2	2	1	1.525	2.088	1.472	1.172	1.122
5	1	2	2	2	2	1.912	1.216	1.179	1.078	1.179
6	1	2	2	2	3	1.151	1.811	1.271	1.209	1.690
7	1	3	3	3	1	1.983	1.134	1.246	1.472	1.406
8	1	3	3	3	2	1.721	1.995	1.500	1.699	1.116
9	1	3	3	3	3	1.106	1.559	1.106	1.735	1.953
10	2	1	2	3	1	1.798	1.762	1.108	1.370	1.322
11	2	1	2	3	2	1.134	1.535	1.235	1.351	1.188
12	2	1	2	3	3	1.768	1.129	1.891	1.866	1.128
13	2	2	3	1	1	1.232	1.613	1.128	1.174	1.460
14	2	2	3	1	2	1.301	1.374	1.126	1.527	1.490
15	2	2	3	1	3	2.013	1.075	1.792	1.119	1.225
16	2	3	1	2	1	1.127	1.692	1.686	1.197	1.182
17	2	3	1	2	2	1.106	1.491	1.241	1.165	1.109
18	2	3	1	2	3	1.245	1.687	1.167	1.118	1.759
19	3	1	3	2	1	1.165	1.835	1.193	2.139	1.540
20	3	1	3	2	2	1.444	1.202	1.772	1.180	1.212
21	3	1	3	2	3	1.405	1.271	1.144	1.144	1.744
22	3	2	1	3	1	1.661	1.231	1.556	1.136	1.781
23	3	2	1	3	2	1.313	1.404	1.218	1.140	1.581
24	3	2	1	3	3	1.985	1.257	1.104	1.171	1.453
25	3	3	2	1	1	1.257	1.042	1.076	1.305	1.183
26	3	3	2	1	2	1.365	1.252	2.000	1.269	1.113
27	3	3	2	1	3	1.126	1.168	1.401	1.101	1.104

Table 17 Computational results to tune MOPSO for medium-size problem

	A	B	C	D	E	F	G	R1	R2	R3	R4	R5
1	1	1	1	1	1	1	1	1.674	1.334	1.228	1.723	1.215
2	1	1	1	1	2	2	2	1.950	1.227	1.223	1.272	1.215
3	1	1	1	1	3	3	3	1.165	2.058	1.147	1.473	1.384
4	1	2	2	2	1	1	1	1.528	1.480	1.648	1.952	1.418
5	1	2	2	2	2	2	2	1.293	1.081	1.165	1.574	1.082
6	1	2	2	2	3	3	3	1.191	1.661	1.198	1.446	1.234
7	1	3	3	3	1	1	1	1.648	1.246	1.252	1.787	1.414
8	1	3	3	3	2	2	2	1.302	1.593	1.642	1.382	1.846
9	1	3	3	3	3	3	3	1.732	1.577	1.345	1.362	1.126
10	2	1	2	3	1	2	3	2.114	1.250	1.468	1.334	1.730
11	2	1	2	3	2	3	1	1.240	1.316	1.581	1.607	1.233
12	2	1	2	3	3	1	2	1.250	1.279	1.154	1.194	1.160
13	2	2	3	1	1	2	3	1.839	1.906	1.532	2.419	1.541
14	2	2	3	1	2	3	1	1.387	1.179	1.142	1.606	1.308
15	2	2	3	1	3	1	2	1.072	1.429	1.176	1.053	1.059
16	2	3	1	2	1	2	3	1.466	1.743	2.416	1.445	2.037
17	2	3	1	2	2	3	1	1.786	1.374	1.738	1.171	1.217
18	2	3	1	2	3	1	2	1.172	1.108	1.199	1.968	1.214
19	3	1	3	2	1	3	2	2.865	3.020	3.347	2.660	3.798
20	3	1	3	2	2	1	3	1.114	1.659	1.216	1.121	1.227
21	3	1	3	2	3	2	1	1.100	1.155	1.243	1.585	1.285
22	3	2	1	3	1	3	2	1.946	1.638	1.514	1.994	1.489
23	3	2	1	3	2	1	3	1.333	1.374	1.190	1.597	1.398
24	3	2	1	3	3	2	1	1.103	1.187	1.498	1.864	1.202
25	3	3	2	1	1	3	2	1.503	1.889	2.455	2.840	1.523
26	3	3	2	1	2	1	3	1.469	1.451	1.190	2.077	1.188
27	3	3	2	1	3	2	1	1.670	1.064	1.142	1.272	1.315

Table 18 Computational results to tune SPEA-II for large-size problem

	A	B	C	D	E	F	G	R1	R2	R3	R4	R5
1	1	1	1	1	1	1	1	1.674	1.445	1.425	1.765	1.939
2	1	1	1	1	2	2	2	1.356	1.413	1.519	1.708	1.603
3	1	1	1	1	3	3	3	1.580	1.939	2.058	1.867	1.353
4	1	2	2	2	1	1	1	1.626	1.473	1.600	1.584	1.169
5	1	2	2	2	2	2	2	1.637	1.2703	1.856	1.880	1.521
6	1	2	2	2	3	3	3	2.011	1.4122	1.496	1.527	1.938
7	1	3	3	3	1	1	1	1.050	1.099	1.691	1.126	1.175
8	1	3	3	3	2	2	2	1.187	1.009	1.692	1.969	1.188
9	1	3	3	3	3	3	3	1.108	1.598	1.034	1.520	1.206
10	2	1	2	3	1	2	3	1.669	1.229	1.428	1.227	1.312
11	2	1	2	3	2	3	1	1.094	1.271	1.613	1.092	1.249
12	2	1	2	3	3	1	2	1.295	1.097	1.256	1.030	1.514
13	2	2	3	1	1	2	3	1.214	1.236	1.345	1.346	1.160
14	2	2	3	1	2	3	1	1.072	1.154	1.255	1.137	1.070
15	2	2	3	1	3	1	2	1.471	1.482	1.579	1.529	1.163
16	2	3	1	2	1	2	3	1.395	1.144	1.526	1.062	0.988
17	2	3	1	2	2	3	1	1.273	0.980	0.964	1.380	1.169
18	2	3	1	2	3	1	2	1.691	0.917	1.209	1.567	1.016
19	3	1	3	2	1	3	2	1.790	2.705	1.458	1.507	1.425
20	3	1	3	2	2	1	3	1.089	1.288	1.754	1.522	1.678
21	3	1	3	2	3	2	1	1.304	1.294	1.267	1.286	1.311
22	3	2	1	3	1	3	2	2.144	1.378	1.304	2.034	1.448
23	3	2	1	3	2	1	3	1.658	1.378	1.445	1.054	1.276
24	3	2	1	3	3	2	1	1.472	1.1552	1.388	1.423	1.165
25	3	3	2	1	1	3	2	1.126	1.250	1.235	1.060	1.328
26	3	3	2	1	2	1	3	1.169	1.370	1.342	1.280	1.620
27	3	3	2	1	3	2	1	0.958	1.058	1.380	1.313	1.191

Table 19 Computational results to tune PESA-II for large-size problem

	A	B	C	D	E	F	R1	R2	R3	R4	R5
1	1	1	1	1	1	1	2.163	1.872	1.812	2.661	1.762
2	1	1	1	1	2	2	1.213	1.776	1.113	1.344	1.109
3	1	1	1	1	3	3	1.548	1.309	1.425	1.948	1.217
4	1	2	2	2	1	1	0.996	1.073	1.753	1.622	0.983
5	1	2	2	2	2	2	0.909	1.195	0.912	1.262	1.230
6	1	2	2	2	3	3	1.231	1.133	1.356	1.325	1.290
7	1	3	3	3	1	1	0.918	1.184	1.776	1.034	0.933
8	1	3	3	3	2	2	0.889	0.965	0.854	0.857	0.947
9	1	3	3	3	3	3	1.115	1.192	1.114	1.411	0.921
10	2	1	2	3	1	2	1.108	1.829	1.240	1.168	1.080
11	2	1	2	3	2	3	1.596	1.718	1.599	1.373	2.036
12	2	1	2	3	3	1	0.925	0.913	0.913	1.249	1.242
13	2	2	3	1	1	2	1.137	0.993	1.288	1.290	1.701
14	2	2	3	1	2	3	1.293	1.238	1.031	1.021	1.142
15	2	2	3	1	3	1	1.246	1.744	0.959	1.261	1.546
16	2	3	1	2	1	2	0.913	1.018	1.133	1.069	0.931
17	2	3	1	2	2	3	1.011	1.117	1.378	1.148	0.989
18	2	3	1	2	3	1	0.986	1.166	1.283	0.997	0.951
19	3	1	3	2	1	3	2.235	2.826	2.721	2.193	3.375
20	3	1	3	2	2	1	1.516	1.933	1.229	1.371	1.170
21	3	1	3	2	3	2	4.222	5.191	4.533	4.597	3.764
22	3	2	1	3	1	3	1.705	1.020	1.079	1.110	1.066
23	3	2	1	3	2	1	1.205	1.712	1.577	1.641	1.649
24	3	2	1	3	3	2	0.996	1.244	0.960	1.164	1.087
25	3	3	2	1	1	3	1.023	1.310	1.737	1.003	1.151
26	3	3	2	1	2	1	0.972	1.296	1.064	1.732	0.928
27	3	3	2	1	3	2	1.427	0.996	1.067	0.897	1.108

Table 20 Computational results to tune NSGA-II for large-size problem

	A	B	C	D	E	R1	R2	R3	R4	R5
1	1	1	1	1	1	0.950	0.834	0.918	0.817	0.857
2	1	1	1	1	2	0.878	0.816	0.891	1.343	0.881
3	1	1	1	1	3	1.115	1.209	0.910	0.880	0.843
4	1	2	2	2	1	0.787	1.356	0.806	0.972	0.988
5	1	2	2	2	2	1.300	0.832	1.180	1.366	0.988
6	1	2	2	2	3	1.128	0.916	0.997	0.810	1.122
7	1	3	3	3	1	0.813	1.364	0.938	0.818	1.089
8	1	3	3	3	2	0.985	0.941	1.554	0.851	1.053
9	1	3	3	3	3	1.025	0.906	0.858	1.078	0.893
10	2	1	2	3	1	0.827	1.451	1.246	1.062	0.814
11	2	1	2	3	2	0.859	0.881	0.911	0.804	1.214
12	2	1	2	3	3	0.870	1.014	0.864	1.491	1.243
13	2	2	3	1	1	0.965	0.994	1.250	1.423	0.840
14	2	2	3	1	2	1.178	1.265	0.855	0.816	1.071
15	2	2	3	1	3	0.881	1.072	0.841	1.119	1.026
16	2	3	1	2	1	0.840	1.035	1.167	0.967	1.106
17	2	3	1	2	2	0.956	0.817	1.169	1.358	0.983
18	2	3	1	2	3	1.033	0.857	0.848	0.832	1.157
19	3	1	3	2	1	0.839	1.287	0.879	0.920	0.874
20	3	1	3	2	2	1.099	1.229	0.856	0.977	1.159
21	3	1	3	2	3	0.830	0.840	1.211	0.859	0.825
22	3	2	1	3	1	0.909	0.837	1.000	0.924	0.845
23	3	2	1	3	2	0.841	0.912	1.333	1.018	0.841
24	3	2	1	3	3	1.064	1.385	1.000	0.832	1.268
25	3	3	2	1	1	1.021	1.125	0.880	0.852	0.909
26	3	3	2	1	2	0.930	0.912	0.927	0.927	0.809
27	3	3	2	1	3	1.116	0.797	1.100	1.288	1.251

Table 21 Computational results to tune MOPSO for large-size problem

	A	B	C	D	E	F	G	R1	R2	R3	R4	R5
1	1	1	1	1	1	1	1	1.464	1.896	1.386	1.306	1.395
2	1	1	1	1	2	2	2	1.681	1.503	1.357	1.210	1.236
3	1	1	1	1	3	3	3	2.275	1.567	1.249	2.008	1.226
4	1	2	2	2	1	1	1	1.369	1.280	1.613	1.937	1.497
5	1	2	2	2	2	2	2	1.224	1.229	1.239	1.749	1.283
6	1	2	2	2	3	3	3	1.338	1.214	1.744	1.180	1.297
7	1	3	3	3	1	1	1	1.273	1.393	1.449	1.573	1.850
8	1	3	3	3	2	2	2	1.421	2.169	1.316	1.239	2.046
9	1	3	3	3	3	3	3	1.354	1.290	1.227	1.931	1.821
10	2	1	2	3	1	2	3	2.382	2.279	2.230	2.019	1.617
11	2	1	2	3	2	3	1	1.813	1.259	1.928	1.756	1.218
12	2	1	2	3	3	1	2	2.004	1.576	1.132	1.657	1.343
13	2	2	3	1	1	2	3	1.606	2.039	1.648	2.538	1.789
14	2	2	3	1	2	3	1	1.214	1.111	1.351	1.419	1.190
15	2	2	3	1	3	1	2	1.957	1.158	1.758	1.335	1.477
16	2	3	1	2	1	2	3	1.358	1.397	1.567	1.321	1.386
17	2	3	1	2	2	3	1	1.756	1.580	1.648	1.483	1.280
18	2	3	1	2	3	1	2	1.965	1.219	1.261	1.529	1.529
19	3	1	3	2	1	3	2	2.828	1.587	1.733	1.963	2.487
20	3	1	3	2	2	1	3	1.526	1.218	1.531	1.372	1.867
21	3	1	3	2	3	2	1	1.095	1.754	1.194	1.164	1.177
22	3	2	1	3	1	3	2	1.977	1.312	1.336	1.677	1.318
23	3	2	1	3	2	1	3	1.233	1.266	1.663	1.763	1.253
24	3	2	1	3	3	2	1	1.331	1.353	1.400	1.205	1.343
25	3	3	2	1	1	3	2	2.678	2.238	1.633	1.850	2.234
26	3	3	2	1	2	1	3	2.264	1.295	1.258	1.250	1.276
27	3	3	2	1	3	2	1	1.360	1.326	1.360	1.187	1.165

Appendix 2: Computational results for meta-heuristic algorithms

See Tables 22, 23, 24 and 25.

Table 22 Computational result for NSGA-II

	NPS	MID	S	D	SM	CPU-time
1	13	47.1397726	1.15514457	30.4138127	1.54994618	404.3489
2	14	46.5531889	1.14757269	32.4228315	1.43581503	397.0284
3	13	44.6858388	0.87398293	27.1543735	1.64562216	409.4317
4	17	44.3317657	0.60863879	31.7672788	1.39551663	381.7427
5	14	41.9738496	1.04049861	28.4822752	1.47368317	381.5333
6	13	45.5707014	1.41738293	34.5114474	1.32045176	389.9926
7	14	41.8368186	0.47717115	31.4006369	1.33235573	394.3176
8	13	45.4065354	1.03923048	32.9660431	1.37737293	411.0373

Table 22 (continued)

	NPS	MID	S	D	SM	CPU-time
9	13	49.1203646	1.66101606	35.3553391	1.38933372	406.3892
10	11	48.7703652	1.33811265	27.9885691	1.74251013	408.6667
11	18	72.8973812	1.5999183	64.899923	1.12322755	696.1833
12	17	70.3960096	1.19311752	54.7507078	1.28575524	689.1406
13	18	67.2382325	1.28276662	55.6366785	1.20852348	661.4166
14	16	71.8669353	1.8461672	58.5795186	1.226827	675.1025
15	17	70.9549977	1.92686121	60.5775536	1.17130841	720.5656
16	20	70.071016	0.9399888	54.9796326	1.27449044	717.5923
17	15	73.0580059	1.51695183	53.5350353	1.36467653	664.5466
18	19	73.0448098	1.18153434	53.5350353	1.36443003	651.9568
19	12	67.657309	1.63762578	42.9343685	1.575831	656.0386
20	16	68.0715497	1.53046834	65.4794624	1.03958626	650.2064
21	22	85.5844727	2.06378797	105.78204	0.8090643	1330.8438
22	17	99.261788	2.48876889	79.294136	1.25181751	1108.4926
23	17	92.8723042	0.96885317	72.7189109	1.27714102	1005.9143
24	16	92.6021098	1.83693585	87.5646047	1.05752901	999.5877
25	18	95.491341	2.58325427	85.2997069	1.11948029	922.791
26	18	96.4027623	1.04999222	77.5695817	1.2427908	1001.0809
27	19	96.0988283	1.55089109	73.7750635	1.30259228	958.3336
28	17	103.598463	2.8660282	97.0286556	1.06770997	1004.7133
29	20	93.6121928	1.9801648	80.7727677	1.15895735	1093.9078
30	17	92.320798	2.33990322	73.2273173	1.26074259	967.8371
31	21	60.4677456	1.20964379	42.2018957	1.4328206	435.3822
32	15	58.1854484	0.63079693	38.901928	1.49569575	401.2917
33	18	64.0299562	1.69951935	51.1601407	1.25155942	438.0152
34	17	58.5739585	1.34508911	40.174121	1.45800224	415.2276
35	19	62.7519031	0.95194476	50.7290844	1.23700051	422.2327
36	18	60.5144835	0.61484894	41.1052308	1.4721845	416.8058
37	20	60.6945383	0.89195232	50.8806447	1.19288069	396.8396
38	19	60.0498626	1.33456084	47.6629835	1.25988468	395.0797
39	19	61.0855837	1.31424939	49.6423207	1.23051427	414.7991
40	18	57.781136	1.14674874	47.8539445	1.20744772	384.4491
41	20	82.7788412	0.95509713	69.0344841	1.19909408	662.4643
42	17	85.3435522	0.93234366	71.1376131	1.19969659	707.0633
43	22	86.1738884	1.1483095	82.3660124	1.04623116	678.6003
44	22	93.7030719	1.4105355	90.2804519	1.03791098	719.3565
45	21	84.8868424	1.89784339	75.1664819	1.12931775	697.8747
46	19	81.7312804	1.45983256	82.3465846	0.99252787	695.0214
47	19	85.3236462	1.34015797	76.5404468	1.11475239	682.0479
48	22	87.5135909	2.52807611	93.8400767	0.93258226	733.8909
49	19	90.7119066	1.35370404	85.7055424	1.05841354	682.6297
50	16	79.5131544	2.62202212	62.5856214	1.27047	638.7042

Table 22 (continued)

	NPS	MID	S	D	SM	CPU-time
51	19	115.819691	2.90297287	115.725365	1.00081509	933.6823
52	19	101.486267	1.6393195	109.726205	0.92490456	953.0442
53	21	109.115458	2.40158678	120.104788	0.90850215	967.9273
54	21	111.224499	2.13398799	107.389757	1.03570863	1034.5938
55	22	107.612987	1.92805669	105.321603	1.02175607	999.0513
56	22	109.436572	2.51946535	112.738458	0.97071198	968.0552
57	21	106.23281	2.67366344	110.897129	0.96575281	1010.9819
58	22	105.504872	2.07174352	104.847508	1.00626972	963.4064
59	18	113.882639	2.56711226	99.4693923	1.14490133	950.4535
60	20	108.271654	2.27160968	96.0866276	1.12681292	946.9389
61	17	69.2478202	0.9677141	42.3254061	1.63608165	405.0061
62	19	74.3631908	1.36094223	61.6574408	1.20607002	388.7741
63	18	71.4652213	1.5555929	62.2337529	1.1483354	363.6207
64	18	77.8979603	1.70864775	62.5542964	1.24528553	448.1217
65	18	72.2655991	1.51747985	54.0281408	1.3375548	406.0335
66	19	70.9955457	1.32329555	56.8369598	1.24910878	414.6287
67	21	70.3966444	0.83694456	53.9559079	1.30470688	421.5453
68	20	75.5779339	1.58539386	69.5402042	1.08682358	405.0503
69	22	72.8661413	0.91149466	53.0298029	1.37406019	441.3907
70	19	72.8539632	0.5467608	49.2913786	1.47802649	407.9489
71	18	93.7656905	1.00547521	77.5613306	1.20892318	688.5269
72	20	93.1837068	3.37005388	94.9092198	0.98181933	680.50007
73	21	103.345572	1.56372571	93.7360123	1.10251727	702.9857
74	19	102.589858	2.18115265	93.9795722	1.0916187	695.2888
75	20	102.673426	1.3105242	91.7014722	1.11964861	702.8019
76	19	98.0873783	1.31851414	85.5628424	1.14637821	693.5988
77	19	101.514088	1.3268703	79.6075373	1.27518187	714.0684
78	20	94.0399773	1.43152477	72.2775207	1.30109578	662.4655
79	20	95.6773009	1.07869317	77.498129	1.23457562	663.8926
80	19	94.6860621	2.05298243	78.249345	1.21005565	653.0656
81	22	120.66524	2.49761791	119.787979	1.00732344	975.9642
82	24	125.118734	2.58949942	138.657852	0.90235592	980.3482
83	24	132.832551	2.26491834	158.250561	0.83938123	986.1662
84	26	131.234808	2.06127668	142.330461	0.92204303	990.0802
85	21	125.603573	2.1627804	132.313869	0.94928501	993.3865
86	20	127.975749	3.35157465	110.272571	1.16054017	980.9684
87	20	120.66464	2.82812938	124.52309	0.96901418	948.9089
88	23	127.827031	2.0052106	137.148241	0.93203551	932.1596
89	23	129.445911	2.07975371	142.043514	0.91131166	975.2262
90	22	120.215464	2.22189031	120.531158	0.99738081	934.8399
Average	18.678	84.755	1.645	75.316	1.194	695.156

Table 23 Computational result for PESA-II

	NPS	MID	S	D	SM	CPU-time
1	17	46.2698057	0.43639567	29.6984848	1.5579854	406.0273
2	19	46.1834017	0.69727959	31.8276609	1.45104605	412.659
3	16	43.9355437	0.50924781	25.7945731	1.70328633	398.7296
4	19	43.7818727	0.47041104	30.9864487	1.41293613	409.1372
5	16	41.9716088	0.32145503	27.6810404	1.51625835	387.2319
6	21	45.0858437	0.52517571	34.5114474	1.30640257	388.2439
7	19	40.7798261	0.50355462	29.8328678	1.36694288	391.9545
8	17	44.368793	0.90805221	32.9660431	1.3458938	421.2683
9	18	48.599952	1.4087495	35.3553391	1.37461422	435.3863
10	18	47.9740198	1.21008831	27.9885691	1.71405761	382.3779
11	21	72.6601356	0.90811159	62.4256358	1.16394707	713.7333
12	22	69.1902886	1.02615155	56.6127194	1.22216861	663.8644
13	21	66.764215	0.61690472	52.8393793	1.2635314	684.9124
14	24	71.5541353	0.90505825	57.6451212	1.24128693	664.2288
15	23	69.7407554	1.26453603	59.7414429	1.16737648	667.5721
16	19	70.2049691	0.94677044	54.5046787	1.28805399	718.8464
17	21	72.1141803	0.87998918	52.4785671	1.37416443	639.0639
18	21	70.2138496	0.66590433	52.6501662	1.33359217	634.8961
19	20	65.6409439	0.46724275	40.6649726	1.61418881	670.2005
20	23	67.0746197	0.96896506	64.3506022	1.04233088	639.9411
21	29	88.669748	1.80535602	102.342562	0.86640149	1246.1992
22	24	93.4927504	1.46851004	79.2933793	1.17907385	901.9445
23	22	93.0473478	1.45620036	72.5407472	1.28269078	853.4562
24	23	96.1832789	1.26628522	86.7640479	1.10856145	901.8319
25	24	97.1595563	1.81647054	83.5693724	1.16262159	918.5218
26	22	94.759046	1.10284952	75.6066135	1.25331689	908.1244
27	24	94.6240866	0.96413963	76.4222481	1.2381746	916.5098
28	22	105.970194	1.95831008	97.0286556	1.09215358	985.2157
29	24	92.778661	0.9895102	79.8764045	1.16152776	916.3508
30	19	95.5031521	2.00189968	73.5923909	1.29773134	900.379
31	19	60.0937133	1.19868349	42.2018957	1.42395768	407.7626
32	17	56.4009118	0.50278635	35.571899	1.58554683	396.3088

Table 23 (continued)

	NPS	MID	S	D	SM	CPU-time
33	25	62.2233139	0.66206747	51.1601407	1.21624595	414.7023
34	19	58.0389241	1.08773062	40.174121	1.44468436	391.5689
35	23	62.1139035	0.90081211	49.1674689	1.26331302	412.679
36	21	59.9946615	0.41069048	40.3316253	1.48753394	410.5243
37	21	59.3438614	0.85940179	48.259714	1.22967702	393.7719
38	21	59.3968807	0.98927583	47.6629835	1.2461847	390.8066
39	24	60.0932839	0.74152213	49.5241355	1.21341409	411.5321
40	19	56.6401288	0.6680687	43.4165867	1.30457351	385.5541
41	20	81.2239629	1.16429333	68.8967343	1.17892326	1333.3063
42	21	84.5021358	1.05270626	68.8726361	1.22693337	985.6504
43	22	85.4659355	1.49845519	79.7568806	1.07158072	971.6923
44	25	90.7169138	1.47673062	88.391176	1.02631188	1166.7499
45	21	83.7930893	1.19427204	73.8704271	1.13432523	1179.3059
46	22	79.2681778	1.12638563	75.3721434	1.05169064	697.7265
47	21	83.4234039	1.14953407	75.9041501	1.09906249	1763.2838
48	23	87.8347139	1.23803363	93.5694395	0.93871155	1677.4041
49	22	91.2229532	1.71582903	83.738641	1.08937704	841.3523
50	24	80.9203292	1.25230223	62.4374887	1.29602152	999.2546
51	25	115.96777	1.7218401	104.618545	1.10848196	958.3031
52	27	100.284664	1.9533593	109.935618	0.91221267	899.2519
53	25	108.864057	2.38103619	116.720007	0.93269406	894.9629
54	23	108.967093	1.2657857	106.001132	1.02798047	987.6576
55	26	105.331474	1.72849423	105.304511	1.00025605	983.5607
56	28	111.577453	1.52880629	110.171503	1.01276147	967.1907
57	26	107.101322	1.18513875	108.064055	0.99109109	973.3899
58	24	106.467204	1.427829	102.727796	1.03640114	915.1254
59	24	113.931788	1.51958232	98.0051019	1.16250874	939.2668
60	26	104.655286	1.31976688	96.0866276	1.08917639	880.7009
61	21	71.1871774	0.93990881	48.335908	1.4727597	398.1095

Table 23 (continued)

	NPS	MID	S	D	SM	CPU-time
62	22	69.4314409	1.16760012	55.3772516	1.25378994	384.9892
63	25	69.8640444	1.06580173	51.9234051	1.34552124	391.4617
64	23	76.3089709	1.29748921	64.4127317	1.1846877	415.3885
65	21	70.8058727	0.97414187	51.3813196	1.37804699	401.7340
66	20	70.5037648	0.90565475	53.4269595	1.31962899	408.9691
67	21	69.0724372	0.93253367	52.2612667	1.32167553	418.9994
68	24	74.0236898	1.11335763	61.358292	1.20641705	399.6253
69	23	71.6670996	0.82125904	51.3824873	1.39477677	406.5263
70	29	66.7932027	0.7338407	49.1011202	1.36031933	409.9435
71	23	91.7796345	1.06752271	72.427619	1.2671911	638.0336
72	24	95.6457181	1.58831786	95.131488	1.00540547	668.6071
73	26	101.008215	1.09505181	87.6926451	1.15184363	664.0481
74	26	99.8748489	1.30263638	88.0009091	1.13492974	657.4868
75	24	101.872887	1.37829973	91.2175422	1.11681245	649.1887
76	21	96.6816313	1.5655822	86.0009302	1.12419285	688.4834
77	21	99.2324103	1.48311127	78.4948406	1.26419023	708.8898
78	24	93.3707034	1.21094047	71.5614421	1.30476274	662.6285
79	23	94.2379752	1.20171156	73.830617	1.2764078	657.4473
80	25	102.368686	1.8521339	105.550936	0.96985105	711.6483
81	25	121.320452	2.01988449	118.249905	1.02596659	912.8097
82	25	123.200852	2.56872212	128.29279	0.96031002	916.9095
83	27	131.576697	2.9035002	156.984076	0.83815315	959.221
84	25	131.282616	1.81901072	136.080711	0.96474081	937.7472
85	25	121.518148	2.99435024	128.088095	0.94870759	959.8625
86	25	126.361489	1.36154324	106.566224	1.18575553	980.3902
87	26	119.611587	1.88607855	123.239604	0.97056127	907.5629
88	27	127.347931	1.88017093	133.206606	0.95601813	929.685
89	25	129.186669	2.66852019	137.153345	0.94191409	987.1512
90	25	125.347579	2.75095135	139.517024	0.89843931	1063.8213
Average	22.589	84.052	1.266	73.997	1.205	727.850

Table 24 Computational result for MOPSO

	NPS	MID	S	D	SM	CPU-time
1	19	47.2107897	0.79970755	31.689746	1.48978126	145.66547
2	20	47.1284923	0.51524138	33.8354843	1.39287181	136.8452
3	16	44.6985034	0.3732738	28.5895086	1.56345826	139.8835
4	18	44.3615237	0.65698857	28.5895086	1.55167143	142.2329
5	16	42.686323	0.80145701	30.120425	1.4171886	127.9942
6	19	45.9763095	0.67121257	36.0693776	1.27466324	136.0378
7	19	41.7532943	0.50215908	31.4006369	1.32969578	131.5177
8	19	45.5088274	0.57236556	33.7010386	1.35036869	142.6247
9	17	49.4571082	1.70051895	36.4878062	1.35544209	141.3721
10	17	48.4487608	0.58158	28.8506499	1.6792953	131.0916
11	27	73.5991631	1.41962236	68.6002915	1.07286954	482.5437
12	26	71.7070074	1.02719933	59.4763819	1.20563836	834.5827
13	24	67.820667	1.40072445	60.4470016	1.12198563	878.4093
14	27	73.1875309	0.81243891	64.2435989	1.13921904	715.8227
15	25	70.8194416	1.15836091	60.9133811	1.16262536	320.2755
16	25	70.5733818	1.52385476	60.3075451	1.17022475	289.4074
17	23	74.6473029	0.67601483	55.7584074	1.33876318	303.734
18	24	71.9219682	1.40227558	57.2646488	1.25595755	508.1873
19	23	66.9349379	0.87010063	47.848093	1.39890503	778.8503
20	28	70.8099069	1.45605832	79.3987405	0.89182658	526.6253
21	28	88.1762886	1.27208532	86.3733755	1.02087348	705.6707
22	26	97.6535316	1.59163196	86.5390085	1.12843368	546.3543
23	26	93.0589351	1.17444586	77.0633506	1.20756409	515.0402
24	28	95.6772963	1.94954886	91.7877988	1.04237489	521.0693
25	27	100.525614	1.1469767	88.8686671	1.1311705	549.4704
26	27	96.6220411	1.5358211	83.3088231	1.15980562	551.9013
27	25	97.0208513	0.75643903	79.2487224	1.22425761	537.6956
28	28	103.590094	2.0187219	104.115513	0.9949535	510.6354
29	29	94.9201121	1.55629977	92.4400346	1.02682904	514.8878
30	25	96.6080744	1.85098172	83.8250559	1.15249639	513.7072
31	22	60.354482	0.7066781	39.0824769	1.54428498	143.6008
32	23	57.525495	0.56791693	40.511233	1.41998875	140.1819
33	23	62.9250832	0.77209228	48.9608006	1.28521353	147.469

Table 24 (continued)

	NPS	MID	S	D	SM	CPU-time
34	20	58.5542033	0.81602374	39.0082043	1.50107405	129.691
35	25	62.031124	0.73084426	48.2700735	1.28508451	141.4495
36	24	62.9362608	1.13169105	52.1724065	1.20631316	148.1504
37	27	58.9823174	1.88955745	53.5421329	1.10160567	129.1143
38	22	60.2504445	0.63135942	43.6022935	1.38181824	141.425
39	24	60.1387306	1.5582413	51.7304552	1.16254014	140.0592
40	21	55.5816264	2.65923369	48.2103723	1.15289768	128.146
41	26	85.6219223	1.32046612	79.1760065	1.08141249	330.6834
42	29	85.8713147	1.19357888	76.5665723	1.12152486	301.2688
43	29	91.2944675	1.27677241	103.070073	0.88575146	309.702
44	28	95.386611	3.08745542	100.396016	0.95010355	302.9874
45	29	86.3417094	1.10011195	84.0061902	1.02780175	303.9911
46	30	80.768561	1.51470949	82.3150047	0.98121310	281.5435
47	28	85.4029025	1.7907168	77.776346	1.09805753	329.1288
48	29	91.2742067	1.58651957	104.478897	0.87361380	302.8387
49	29	91.0936728	2.84761589	93.1652296	0.97776470	310.932
50	26	82.3971688	1.09684863	68.270345	1.20692475	322.4328
51	29	111.212506	1.80915842	113.856225	0.97678025	529.3281
52	29	100.411811	2.06872084	116.578557	0.86132316	522.6306
53	29	108.74034	1.06118254	106.001132	1.02584137	494.9708
54	30	113.418832	1.33817512	114.049288	0.99447208	570.5058
55	26	110.590869	2.83173228	123.223374	0.89748288	552.6276
56	27	112.623416	2.06719045	121.928504	0.92368407	539.061
57	30	109.133306	3.11879729	117.443603	0.92924019	510.8029
58	30	109.784166	1.649298	116.928867	0.93889703	536.2339
59	28	116.025448	2.03144591	105.13401	1.10359576	530.875
60	29	110.975835	1.87326274	106.396689	1.04694184	498.7081
61	27	75.9742604	0.6704168	59.9286242	1.26774578	135.7040
62	24	68.73086	1.18143857	49.0505861	1.40122403	130.6302
63	27	70.4639347	2.74252156	60.2756999	1.16902723	129.4805
64	20	71.7953221	1.81357455	51.3813196	1.39730397	147.7210
65	23	73.1117685	2.07984873	58.3177503	1.25367951	134.0006

Table 24 (continued)

	NPS	MID	S	D	SM	CPU-time
66	24	69.9830486	1.96944044	60.1295269	1.16387162	410.7584
67	23	69.2060001	1.52991255	48.5732437	1.42477617	359.4070
68	24	72.885556	1.81287665	63.8087768	1.14224970	379.9615
69	23	71.047282	1.94112962	49.6080639	1.43217204	355.2310
70	24	71.390226	1.33953269	44.7320914	1.59595101	385.4683
71	27	96.8352539	3.38645702	95.4670624	1.01433156	297.7737
72	29	97.325186	1.56507448	97.3120753	1.00013473	282.2374
73	26	107.546795	2.06607767	108.226614	0.99371856	301.1156
74	28	102.015209	1.90212913	86.2786184	1.18239271	322.71373
75	29	105.444654	2.06605214	101.212647	1.04181302	331.2807
76	30	100.03617	1.26868563	90.1441069	1.10973611	306.2976
77	29	100.794657	3.74699223	89.3756119	1.12776466	323.9188
78	29	97.3267348	4.34843837	98.2637268	0.99046452	291.6533
79	28	98.7343301	1.51671463	82.6145266	1.19512069	315.9374
80	29	103.892955	1.22556913	92.9191046	1.11810112	324.9288
81	29	123.926468	3.55172892	142.249645	0.87119003	506.7652
82	30	123.69209	1.81173696	134.380058	0.92046463	518.1973
83	30	128.14392	2.65454717	145.041925	0.88349572	545.4812
84	28	131.352243	3.25997046	132.280006	0.99298637	568.7789
85	28	126.927491	2.71008307	140.563011	0.90299354	560.0773
86	30	128.435177	1.2945154	119.200839	1.07746873	530.7032
87	28	121.702129	2.12291603	116.283963	1.04659427	542.7187
88	29	129.460824	2.12746525	139.248555	0.92971036	529.8913
89	29	131.725787	2.68198698	143.401534	0.91858004	565.0641
90	30	124.151312	2.84746308	137.887635	0.90038031	554.0897
Average	25.733	85.609	1.635	79.124	1.152	370.430

Table 25 Computational result for SPEA-II

	NPS	MID	S	D	SM	CPU-time
1	19	45.8196092	0.65257027	29.6984848	1.54282649	111.686
2	19	46.1038027	1.07066138	31.8276609	1.44854511	107.9909
3	18	43.8531578	0.54616907	25.7945731	1.70009241	105.9649
4	23	43.8539851	0.28727973	30.9864487	1.41526335	112.9407
5	17	41.8713699	0.64431907	27.9256155	1.49938933	110.9128
6	22	45.3297431	0.43444745	34.884322	1.30257882	110.3131
7	16	41.0320875	0.54680892	28.6803068	1.43067115	104.573
8	17	44.7477963	0.9087807	32.9660431	1.35739058	110.7285
9	17	48.7841204	1.52850369	35.3553391	1.37982329	114.4998
10	16	48.0838218	1.23382873	27.9885691	1.71798071	110.001
11	28	69.4455447	1.21267556	59.5116795	1.16692295	185.9419
12	26	69.087195	0.95743246	55.7584074	1.23904534	183.1662
13	25	65.1640361	1.14376571	53.0135832	1.22919509	175.8171
14	30	70.3001797	1.04399762	60.7170487	1.15783262	182.6557
15	27	68.7360636	1.28146995	61.7562952	1.11302116	185.7314
16	27	68.2450069	0.97361195	54.5046787	1.25209447	184.5096
17	25	71.15649	1.02368615	53.6809091	1.32554555	177.1059
18	27	69.143808	0.92785326	52.6501662	1.31326856	172.1769
19	24	65.0515052	0.67801928	42.8886931	1.51675186	177.4057
20	30	66.6622409	0.97443175	65.3587026	1.01994437	171.038
21	30	88.6049831	1.49078395	102.083299	0.86796747	337.3966
22	26	92.0240535	1.89486756	79.8521133	1.15243103	268.697
23	30	89.459707	1.21060448	73.6114122	1.21529671	247.376
24	30	94.1057577	1.66486339	87.7437177	1.07250707	259.3033
25	29	94.3153218	1.53834407	87.2900911	1.08048142	263.7475
26	27	93.270563	1.46262068	79.7481034	1.16956465	262.8124
27	30	91.6872375	1.31197841	73.2273173	1.25209062	253.3518
28	29	101.09627	1.65859457	91.9193124	1.0998371	268.857
29	30	89.2111615	1.48240252	85.3531487	1.04520059	255.6408
30	27	91.1942278	1.80066464	77.6309217	1.17471525	249.7666
31	26	59.621649	1.02660006	42.2018957	1.41277182	112.0819
32	19	56.2065457	1.38412047	35.571899	1.5800828	110.9581
33	26	62.3879428	1.04029582	51.1601407	1.21946386	117.6835
34	23	58.4904442	1.06959414	41.0238955	1.42576524	110.9026
35	24	63.0999088	0.94508645	49.1674689	1.28336703	115.2032
36	26	60.3006636	0.62264943	40.3316253	1.49512109	115.2988
37	24	60.1259726	0.79052358	50.8806447	1.18170619	110.5635
38	29	60.0917305	0.606695	47.6629835	1.26076309	109.2707
39	27	60.0469936	0.64162365	48.8401474	1.22945971	113.3477
40	26	56.6188544	0.64925993	43.4165867	1.3040835	114.1766
41	30	81.5640106	1.68689336	74.8227238	1.09009678	192.9555
42	29	83.1205462	1.55305198	72.4985517	1.1465132	184.7751

Table 25 (continued)

	NPS	MID	S	D	SM	CPU-time
43	30	83.9103998	0.88932365	81.3181407	1.03187799	186.3201
44	30	89.3322797	1.51561982	84.2911621	1.059806	199.6176
45	30	83.6032312	1.45662182	73.4019073	1.138979	202.0862
46	30	77.4973461	1.21972298	71.8913068	1.07797938	186.4877
47	30	81.1963993	1.31547815	69.3423392	1.17094982	190.6368
48	30	84.0538942	1.51206641	82.4545936	1.01939614	193.1974
49	30	90.3641727	1.08362949	80.1560977	1.12735244	189.6752
50	30	79.6169145	0.71554175	64.0312424	1.2434073	178.5168
51	30	108.437173	1.995397	101.360545	1.0698164	256.3998
52	30	95.4507922	2.68297313	108.00537	0.88375969	251.9227
53	30	103.39612	2.40495657	103.128851	1.0025916	258.8875
54	30	107.780678	1.59360792	108.475988	0.9935902	266.713
55	30	103.915476	1.56554864	108.216265	0.96025746	293.6362
56	30	108.804908	2.10667249	108.216265	1.00543951	268.7855
57	30	105.347279	2.99581317	106.850363	0.98593281	266.0543
58	30	103.446839	1.4507707	104.635749	0.98863763	266.6034
59	30	109.985012	2.12388713	97.4790234	1.12829415	271.9166
60	30	101.681347	1.08942672	92.1366377	1.10359298	257.2033
61	23	70.2693016	0.88362351	48.9799959	1.43465307	115.975
62	26	72.1023135	0.91765588	57.2450871	1.25953714	108.5034
63	24	68.314609	1.16885922	50.8464355	1.34354765	118.6758
64	30	75.8700466	0.97899785	62.6498204	1.21101778	118.4195
65	25	69.8810733	2.01745715	51.3813196	1.36004824	114.1784
66	27	72.2973523	0.60355075	53.4269595	1.35319983	114.0728
67	27	69.8136062	0.90358577	53.9559079	1.29390105	110.2614
68	29	74.3510139	0.81705949	61.1820235	1.21524281	115.9344
69	23	71.3463822	0.73258436	47.8016736	1.49254988	109.3489
70	23	72.5739761	0.86878226	49.2913786	1.47234624	116.597
71	29	91.4517049	1.28461913	73.830617	1.23866911	181.5128
72	30	91.6995699	1.06034044	77.8973684	1.17718444	181.8349
73	30	101.488575	1.0995924	82.3507134	1.2323946	200.6126
74	30	100.139989	1.87881107	88.0009091	1.13794267	190.7096
75	30	100.904473	1.9137494	91.4846435	1.10296623	185.4717
76	30	97.837655	1.51725705	91.0140648	1.07497292	193.0036
77	30	98.2993649	0.99917207	76.0392004	1.2927459	199.6152
78	30	93.8447176	1.19115514	70.2933852	1.33504337	184.8566
79	30	95.1474691	1.42709415	73.1160721	1.30132085	195.2044
80	30	101.600279	2.11257864	93.6726214	1.08463153	198.8545
81	30	119.489763	1.78241732	102.64989	1.16405154	259.0925
82	30	120.69847	1.77230287	102.566271	1.1767852	266.4988
83	30	128.860154	1.54199453	120.910049	1.06575222	290.6421
84	30	127.808574	1.97578444	136.880094	0.93372652	269.4998

Table 25 (continued)

	NPS	MID	S	D	SM	CPU-time
85	30	120.374167	3.07176987	118.816834	1.013107	271.8504
86	29	123.946899	2.45679915	112.105129	1.10563094	282.7771
87	30	117.45041	2.60618186	120.054821	0.97830649	259.5936
88	30	127.75615	1.98388913	128.223867	0.99635235	263.7688
89	30	125.080555	2.69390116	130.320221	0.95979391	276.2119
90	30	114.565692	1.99843617	102.137946	1.12167609	256.4482
Average	27.167	82.802	1.351	71.961	1.209	188.933

References

1. Arjmand, M., Najafi, A.A.: Solving a multi-mode bi-objective resource investment problem using meta-heuristic algorithms. Adv. Comput. Tech. Electromagn. **1**(2015), 1–18 (2015)
2. Baradaran, S., Fatemi-Ghomi, S.M.T.: A hybrid heuristic rule for constrained resource allocation in PERT type networks. World Appl. Sci. J. **7**(10), 1324–1330 (2009)
3. Baradaran, S., Fatemi-Ghomi, S.M.T., Mobini, M., Hashemin, S.S.: A hybrid scatter search approach for resource-constrained project scheduling problem in PERT-type networks. Adv. Eng. Softw. **41**, 966–975 (2010)
4. Baradaran, S., Fatemi-Ghomi, S.M.T.: Multi-mode renewable resource-constrained allocation in PERT networks. Appl. Soft Comput. **12**(2012), 82–90 (2012)
5. Beg, I., Rashid, T.: Multi-criteria trapezoidal valued intuitionistic fuzzy decision making with Choquet integral based TOPSIS. Opsearch **51**(1), 98–129 (2013)
6. Blazewicz, J., Lenstra, J.K., Rinnooy Kan, A.H.H.: Scheduling subject to resource constraints. Discrete Appl Math **5**, 11–24 (1983)
7. Charnes, A., Cooper, W., Thompson, G.: Critical path analysis via chance constrained and stochastic programming. Oper. Res. **12**, 460–470 (1964)
8. Chen, Z., Demeulemeester, E., Bai, S., Guo, Y.: Efficient priority rules for the stochastic resource-constrained project scheduling problem. Eur. J. Oper. Res. **270**(3), 957–967 (2018)
9. Cochran, W.G., Cox, G.M.: Experimental Designs, 2nd edn. Wiley, New York (1992)
10. Coello-Coello, C.A., Van Veldhuizen, D.A., Lamont, G.B.: Evolutionary Algorithms for Solving Multi-Objective Problems. Kluwer Academic Publishers, Berlin (2002)
11. Corne, D.W., Joshua, N.J., Knowles, D., Oates, M. J.: Region-based selection in evolutionary multi-objective optimization (2003)
12. Creemers, S.: The preemptive stochastic resource-constrained project scheduling problem. Eur. J. Oper. Res. **277**(1), 238–247 (2019)
13. Deb, K.: Multi-objective Optimization Using Evolutionary Algorithms. Wiley, Chichester (2001)
14. Demeulemeester, E.: Minimizing resource availability costs in time-limited project networks. Manag. Sci. **41**, 1590–1598 (1995)
15. Demeulemeester, E.L., Herroelen, W.: Project Scheduling: A Research Handbook. Kluwer Academic Publishers, Boston (2002)
16. Drexl, A., Kimms, A.: Optimization guided lower and upper bounds for the resource investment problem, J. Oper. Res. Soc **52**, 340–351 (2001)
17. Elmaghraby, S.: On the expected duration of PERT type networks. Manag. Sci. **13**, 299–306 (1967)
18. Elmaghraby, S.E.: Activity nets: a guided tour through some recent developments. Eur. J. Oper. Res. **13**(64), 199–215 (1995)
19. Fatemi Ghomi, S.M.T., Hashemin, S.S.: A New Analytical Algorithm and Generation of Gaussian Quadrature Formula for Stochastic Network. Eur. J. Oper. Res. **114**, 610–625 (1999)
20. Freeman, R.J.: A generalized PERT. Oper. Res. **8**, 281–293 (1960)
21. Golenko-Ginzburg, D., Gonik, A.: Stochastic Network Project Scheduling with Non-consumable Limited Resources. Int. J. Prod. Econ. **48**, 29–37 (1997)

22. Goto, H.: Forward-compatible framework with critical-chain project management using a max-plus linear representation. OPSEARCH **54**(1), 201–216 (2016)
23. Hartmann, S., Briskorn, D.: A survey of variants and extensions of the resource constrained project scheduling problem. Eur. J. Oper. Res. **207**, 1–14 (2010)
24. Herroelen, W., De Reyck, B., Demeulemeester, E.L.: Resource-constrained project scheduling: a survey of recent developments. Comput. Oper. Res. **25**, 279–302 (1999)
25. Hwang, C.L., Yoon, K.: Multiple Attribute Decision Making. Springer, Berlin (1981)
26. Icmeli, O., Erenguç, S.S., Zappe, C.J.: Project scheduling problems: a survey. Int. J. Oper. Prod. Manag. **13**, 80–91 (1993)
27. Khalilzadeh, M., Shakeri, H., Gholami, H., Amini, L.: A heuristic algorithm for project scheduling with fuzzyparameters. Proc. Comput. Sci. **121**(2017), 63–71 (2017)
28. Ke, H., Liu, B.: Project scheduling problem with stochastic activity duration times. Appl. Math. Comput. **168**(1), 342–353 (2005)
29. Kellenbrink, C., Helber, S.: Scheduling resource-constrained projects with a flexible project structure. Eur. J. Oper. Res. **246**(2), 379–391 (2015).
30. Kelley, J.E.: The critical-path method: resources planning and scheduling. In: Muth, J., Thompson, G. (eds.) Industrial Scheduling, pp. 347–365. Prentice-Hall, Upper Saddle River, NJ (1963)
31. Kolisch, R., Hartmann, S.: Experimental investigation of heuristics for resource constrained project scheduling: an update. Eur. J. Oper. Res. **174**, 23–37 (2006)
32. Kolisch, R., Hartmann, S.: Heuristic algorithms for solving the resource constrained project scheduling problem: classification and computational analysis. In: Weglarz, J. (ed.) Project Scheduling: Recent Models, Algorithms and Applications, pp. 147–178. Kluwer Academic Publishers, Boston (1998)
33. Kolisch, R., Padman, R.: An integrated survey of project scheduling. Technical Report 463, Manuskripte aus den Instituten für Betriebswirtschaftslehre der Universitat Kiel (1997)
34. Kolisch, R.: Serial and parallel resource-constrained project scheduling methods revisited: theory and computation. Eur. J. Oper. Res. **90**, 320–333 (1996)
35. Kulkarni, V., Adlakha, V.: Markov and Markov-regenerative PERT networks. Oper. Res. **34**, 769–781 (1986)
36. Lambrechts, O., Demeulemeester, E., Herroelen, W.: Proactive and reactive strategies for resource-constrained project scheduling with uncertain resource availabilities. J. Sched. **11**(2), 121–136 (2008)
37. Lambrechts, O., Demeulemeester, E., Herroelen, W.: A tabu search procedure for developing robust predictive project schedules. Int. J. Prod. Econ. **111**(2), 493–508 (2008)
38. Lambrechts, O., Demeulemeester, E., Herroelen, W.: Time slack-based techniques for robust project scheduling subject to resource uncertainty. Ann. Oper. Res. **186**(1), 443–464 (2011)
39. Ma, Z., Demeulemeester, E., He, Z., Wang, N.: A computational experiment to explore better robustness measures for project scheduling under two types of uncertain environments. Comput. Ind. Eng. **131**, 382–390 (2019)
40. Martin, J.: Distribution of the time through a directed acyclic network. Oper. Res. **13**, 46–66 (1980)
41. Möhring, R.F.: Minimizing costs of resource requirements in project networks subject to a fixed completion time. Oper. Res. **32**, 89–120 (1984)
42. Möhring, R.H., Stork, F.: Linear preselective policies for stochastic project scheduling. Math. Methods Oper. Res. **52**, 501–515 (2000)
43. Mukherjee, S., Basu, K.: Solution of interval PERT/CPM network problems by a simplified tabular method. Opsearch **48**(4), 355–370 (2011)
44. Nadjafi, B.: Multi-mode resource availability cost problem with recruitment and release dates for resources. Appl. Math. Model. **38**, 5347–5355 (2014)
45. Ning, M., He, Z., Wang, N.: Metaheuristics for multi-mode cash flow balanced project scheduling with stochastic duration of activities. Autom. Constr. **81**, 224–233 (2017)
46. Nooramin, A.S., Sayareh, J., Moghadam, M.K., Alizmini, H.R.: TOPSIS and AHP techniques for selecting the most efficient marine container yard gantry crane. Opsearch **49**(2), 116–132 (2012)
47. Palmer, C., Kershenbaum, A.: Representing trees in genetic algorithms. In: Proceedings of the first IEEE International Conference on Evolutionary Computation, New York, pp. 376–84 (1994)
48. Rahmati, S.H.A., Hajipour, V., Niaki, S.T.A.: A softcomputing Pareto-based meta-heuristic algorithm for a multi-objective multi-server facility location problem. Appl. Soft Comput. **13**, 1728–1740 (2013)

49. Rahmati, S.H.A., Zandieh, M., Yazdani, M.: Developing two multi-objective evolutionary algorithms for the multi-objective flexible job shop scheduling problem. Int. J. Adv. Manuf. Technol. **64**, 915–932 (2012)
50. Rangaswamy, M.: Multiple Resource Planning and Allocation in Resource-Constrained Project Networks, Ph.D. Thesis, Graduate School of Business, University of Colorado (1998)
51. Ranjbar, M., Kianfar, F., Shadrokh, S.: Solving the resource availability cost problem in project scheduling by path relinking and genetic algorithm. Appl. Math. Comput. **196**, 879–888 (2008)
52. Rodrigues, S., Yamashita, D.: An exact algorithm for minimizing resource availability costs in project scheduling. Eur. J. Oper. Res. **206**, 562–568 (2010)
53. Schott, J.R.: Fault tolerant design using single and multi-criteria genetic algorithms optimization. Master's thesis, Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, Cambridge, MA (1995)
54. Shadrokh, S., Kianfar, F.: A genetic algorithm for resource investment project scheduling problem, tardiness permitted with penalty. Eur. J. Oper. Res. **181**, 86–101 (2007)
55. 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)
56. Taguchi, G.: Introduction to Quality Engineering: Designing Quality into Products and Processes. Asian Productivity Organization, Tokyo (1986)
57. Tao, S., Dong, Z.S.: Scheduling resource-constrained project problem with alternative activity chains. Comput. Ind. Eng. **114**(2017), 288–296 (2017)
58. Tao, S., Wu, C., Sheng, Z., Wang, X.: Stochastic project scheduling with hierarchical alternatives. Appl. Math. Model. **58**, 181–202 (2017)
59. Tao, S., Dong, Z.S.: Multi-mode resource-constrained project scheduling problem with alternative project structures. Comput. Ind. Eng. **125**(2018), 333–347 (2018)
60. Tsai, Y.W., Gemmil, D.D.: Using tabu search to schedule activities of stochastic resource-constrained projects. Eur. J. Oper. Res. **111**, 129–141 (1998)
61. Van Peteghem, V., Vanhoucke, M.: An artificial immune system algorithm for the resource availability cost problem. Flexible. Serv. Manuf. J. **1**, 1 (2011). https://doi.org/10.1007/s10696-011-9117-0
62. Xuebin, L.: Study of multi-objective optimization and multi-attribute decision making for economic and environmental power dispatch. Electr. Power Syst. Res. **79**, 789–795 (2009)
63. Yamashita, D., Armentano, V., Laguna, M.: Scatter search for project scheduling with resource availability cost. Eur. J. Oper. Res. **169**, 623–637 (2006)
64. Yellapu, G., Penmestsa, S.K.: Modeling od a scheduling problem with expected availability of resources. Opsearch **52**(4), 771–781 (2015)
65. Zitzler, E.: Evolutionary Algorithms for Multi-objective Optimization: Methods and Applications. PhD. Thesis, Dissertation ETH No. 13398, Swiss Federal Institute of Technology (ETH), Zürich, Switzerland (1999)
66. Zitzler, E., Laumanns, M., Thiele, L.: Improving the Strength Pareto Evolutionary Algorithm, Computer Engineering and Network Laboratory (2001)