An effective heuristic for project scheduling with resource availability cost

Xia Zhu[a,b], Rubén Ruiz[c], Shiyu Li[a,b], Xiaoping Li[a,b,*]

[a] *School of Computer Science and Engineering, Southeast University, Nanjing 211189, China*
[b] *Key Laboratory of Computer Network and Information Integration, Ministry of Education, Nanjing, 211189, China*
[c] *Grupo de Sistemas de Optimización Aplicada, Instituto Tecnológico de Informática, Ciudad Politécnica de la Innovación, Edifico 8G, Acc. B. Universitat Politècnica de València, Camino de Vera s/n, 46021, València, Spain*

Abstract

The resource constrained project scheduling problem (RCPSP) is widely studied in the literature and has a host of applications in practice. As a variant of the RCPSP, the resource availability cost problem (RACP), which has the aim of minimizing the availability costs of renewable resources in order to complete a project subject to a given deadline, is considered in this paper. We divide the RACP into two sub-problems: the sequencing problem and the resource decision problem, and propose a multi-start iterative search heuristic (MSIS) to solve it. For the sequencing problem, an iterative search framework is constructed to effectively search the activity sequences. A two stage resource adjustment procedure and a backward peak elimination procedure is developed for solving the resource decision problem. MSIS is compared with three existing algorithms on both PSPLib and RanGen data sets involving 1380 instances. A complete calibration of the different parameters and operators of MSIS by means of a design of experiments approach is given. Experimental and statistical results show that MSIS outperforms the other three algorithms in both effectiveness and efficiency by a significant margin.

Keywords: Project Scheduling, Heuristic, Resource Availability Cost

1. Introduction

The resource constrained project scheduling problem (RCPSP) is one of the classical project scheduling problems which receives extensive attention in both academia and industry. The resource availability cost problem (RACP) is a variant of the RCPSP, which has the aim of minimizing renewable resource costs in order to complete a project subject to a

*Corresponding author: Xiaoping Li, Professor of the School of Computer Science and Engineering, Southeast University, Nanjing 211189, China. Tel.& Fax: 86-25-52090916.

Email addresses: zhuxia@seu.edu.cn (Xia Zhu), rruiz@eio.upv.es (Rubén Ruiz), xpli@seu.edu.cn (Xiaoping Li)

given project deadline. RACP was first introduced by Möhring [17], motivated by a small bridge construction project. The author referred to it as the resource investment problem (RIP) and proved that it is NP-hard.

The existing research on the RCPSP is abundant [24], [14], [15]. However, to the best of our knowledge, the research on RACP is rather scarce. The existing algorithms for RACP can be classified into exact algorithms, heuristics and meta-heuristics. Möhring [17] proposed an exact algorithm based on comparability graphs and the recognition and orientation of interval graphs. He also proved the duality relation between the RACP and the Single Mode Resource Constrained Project Scheduling Problem (SMRCPSP). Demeulemeester [2] transformed the RACP into multiple SMRCPSPs, so that it could be solved based on the branch and bound algorithm by Demeulemeester and Herroelen [4]. An exact cutting plane algorithm (MBA) was developed. This made use of efficient points that delimited the solution space of all possible combinations of resource availability. If the maximal completion time (makespan) exceeds the deadline, the availability of one resource is increased by one unit. The procedure above is repeated until the makespan is not greater than the deadline. Rodrigues and Yamashita [20] presented an improved MBA algorithm (MMBA). The initial solutions were constructed by a heuristic and new lower bounds were calculated for reducing the solution space so as to improve the effectiveness of the branching scheme. Experiments were carried out on the PSPLib instances by Kolisch and Sprecher [13], and results showed that MMBA improves MBA by 24%. Drexl and Kimms [7] proposed two lower bounds by Lagrangean relaxation and column generation methods. They pointed out that the solution of the RACP with different deadlines provides time-cost tradeoffs, which were valuable for negotiating the price for the project. Yamashita et al. [25] developed a meta-heuristic based on scatter search, named SS. They also transformed the RACP into muliple SMRCPSPs so that the heuristic by Tormos and Lova [22] could be adopted. Comparison was made on the solutions obtained by SS with those by Demeulemeester [2] and upper and lower bounds by Drexl and Kimms [7] on small instances. SS was also compared with the regular scatter search on large instances. The results showed that SS outperforms the other algorithms in all cases. Shadrokh and Kianfar [21] presented a genetic algorithm for the RACP with tardiness permit. The chromosome was represented by an activity list and a capacity list of resources. Van Peteghem and Vanhoucke [23] proposed an artificial immune algorithm for the RACP. Differing from all the algorithms mentioned above, Ranjbar et al. [19] addressed the RACP without transforming it into SMRCPSPs. The problem was represented via a priority list and converted into a real schedule using an available schedule generation scheme. Two meta-heuristics on the basis of path-relinking (PR) and genetic algorithm (GA) were developed.

In most of the existing work, the RACP is transformed into multiple SMRCPSPs by decreasing the resource availability so that heuristics for the SMRCPSP can be adopted to solve the RACP. Since the SMRCPSP is NP-hard it is time consuming to find a feasible solution under predetermined resources. This usually leads to inefficiencies in the algorithms. Ranjbar et al. [19] first solved the RACP without transforming it into SMRCPSPs. However, in their algorithms, precedence unfeasible activity lists may be generated during the search which makes the algorithm inefficient. Furthermore, a simple resource decision strategy also

makes it difficult to find better solutions. To overcome this disadvantage, in this paper, the RACP is solved by dividing it into two sub-problems: the sequencing problem and the resource decision problem. These two sub-problems are sequentially addressed. A multi-start iterative search method (MSIS) is proposed. In the MSIS for the sequencing problem, a feasible neighbourhood and a path-relinking based method are constructed. For the resource decision problem, two heuristics: backward peak elimination procedure (BPEP) and two stage resource adjustment procedure (TSRAP) are developed. Experiments are carried out and compared with the three best existing algorithms (SS [25], PR [19], GA [19]). The results show that MSIS is superior to the others in both effectiveness and efficiency in a significant way.

The rest of the paper is organized as follows: Section 2 describes the RACP in detail. Section 3 presents the solution representation and feasible solution definition. The multi-start iterative search method is introduced in Section 4. Experimental results are shown in Section 5 and conclusions and future research are given in Section 6.

2. Problem Description

A project with n activities can be represented by an activity-on-node (AON) network denoted as $G = (V, E)$. $V = \{1, 2, ..., n\}$ is the activity set. Activity 1 and n are two dummy activities that indicate the start and finish time of the project respectively. Edge $(i, j) \in E$ denotes the precedence relation between i and j, that is, j can start only after i is finished. $M = \{1, 2, ..., m\}$ is the set of renewable resources of m types needed by the activities of the project. The unit cost of resources of the k-th type is denoted as c_k. Each activity i has a duration d_i and a resource requirement vector $R^i = (r_1^i, r_2^i, ..., r_m^i)$, where r_k^i are the required units of resources of the k-th type needed by activity i over its duration. Variables st_i, ft_i, est_i, lst_i, eft_i, lft_i are the start time, the finish time, the earliest feasible start time, the latest feasible start time, the earliest feasible finish time and the latest feasible finish time of activity i. $succ_i$ and $pred_i$ are the immediate successor set and predecessor set of activity i. D is the deadline of the project. Let $H = \sum_{i=1}^{n} d_i$ be the maximum possible duration of the project, $a_k(t)$ be the usage of resources of the k-th type during the time interval $(t-1, t]$. The resource availability during the project duration is represented as a vector $A = (A_1, A_2, ..., A_m)$ where $A_k = \max_{t \in \{1,...,H\}} \{a_k(t)\}$, and the resource availability cost of a project can be calculated by $\mathcal{C}(A) = \sum_{k \in M} c_k A_k$.

The RACP can be conceptually modeled as:

$$Min\ \mathcal{C}(A) = \sum_{k \in M} c_k A_k \tag{1}$$

s.t.

$$\sum_{t=eft_j}^{lft_j} x_t^j = 1 \tag{2}$$

$$\sum_{t=eft_j}^{lft_j} tx_t^j - \sum_{t=eft_i}^{lft_i} tx_t^i \geq d_j, (i,j) \in E \tag{3}$$

$$\sum_{t=eft_1}^{lft_1} tx_t^1 = 0 \tag{4}$$

$$\sum_{t=eft_n}^{lft_n} tx_t^n \leq D \tag{5}$$

$$A_k \geq \sum_{j \in V} r_k^j x_t^j \geq 0, k \in M, t \in \{1, ..., H\} \tag{6}$$

$$x_t^j \in \{0,1\}, j \in V, t \in \{eft_j, ..., lft_j\} \tag{7}$$

Constraint (7) states that the decision variable x_t^j is one if activity j is completed in time slot $(t-1, t]$ (zero otherwise). Eq.(1) denotes the objective of the project, i.e. the minimization of resource availability cost. Constraint (2) states that every activity must be completed whereas (3) makes sure that the activities are scheduled in accordance with the precedence constraints. Constraint (4) forces the project to start at time zero and constraint (5) determines the deadline constraint of the project. Constraint (6) states that the resource availability of the k-th type resource equals to its maximum usage during all time slots.

Take a RACP project with 10 activities as an example. Assume that there are two types of resources ($m = 2$), the project deadline is $D = 28$, and the unit costs of the two resources are $c_1 = 2$ and $c_2 = 8$ respectively. The project is described by the AON network shown in Figure 1. The two-tuples (R^i, d_i) on the nodes represent the resource requirement vector (R^i) and duration of activity i (d_i). For example, $((3,1),8)$ on node 2 represents that the resource requirement vector of activity 2 is $(3,1)$ and its duration is 8. H is 56.

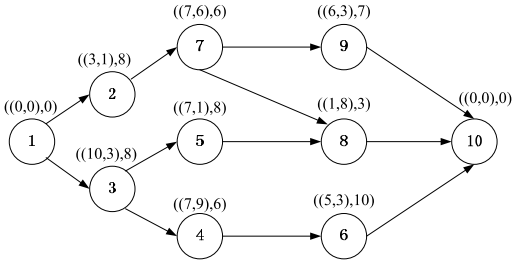


Figure 1: An example of the AON network of a project.

Figure 2 represents a schedule of the project shown in Figure 1. Its makespan is 24. The resource usage variables $a_k(t)$ can be easily obtained. For example, the usage of resource 1 and resource 2 in time slot $(1, 2]$ are $a_1(2) = 13$ and $a_2(2) = 4$ respectively. Also, $a_1(14) = 21$ and $a_2(14) = 16$, $a_1(21) = 11$ and $a_2(21) = 6$, $a_1(25) = 0$ and $a_2(25) = 0$, etc. Therefore, the

resource availability vector is $A = (21, 16)$. The resource availability cost can be calculated by $\mathcal{C}(A) = 2 \times 21 + 8 \times 16 = 170$.

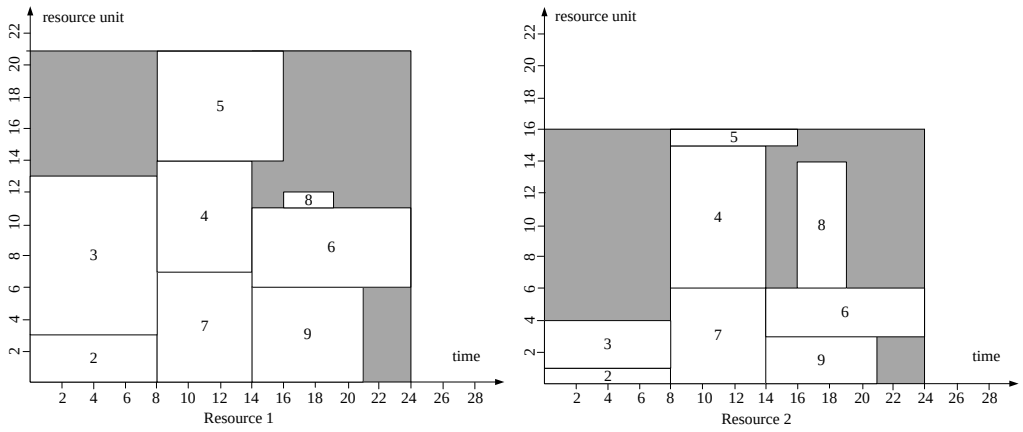


Figure 2: A schedule of the project given in Figure 1.

3. Solution representation and feasible solution

3.1. Solution representation

There are many ways to represent a RACP solution. Yamashita et al. [25] used a resource availability list. By searching for the best resource availability within the upper and lower bounds of the requirments of resource of each type, the problem was transformed into multiple SMRCPSPs. Shadrokh and Kianfar [21] and Van Peteghem and Vanhoucke [23] adopted the combination of a capacity list and an activity list to represent a solution. They also transformed the problem into multiple SMRCPSPs. A priority list was employed to represent a solution by Ranjbar et al. [19]. They explored permutations of the activities and developed a schedule generation scheme.

To represent a solution π in this paper, the combination of an activity list and a resource availability list by Shadrokh and Kianfar [21] is adopted:

$$\pi = (\mathbb{T}, \mathbb{R}) \tag{8}$$

where $\mathbb{T} = (\mathbb{T}_1, \mathbb{T}_2, ..., \mathbb{T}_n)$ is the activity list and $\mathbb{R} = (\mathbb{R}_1, \mathbb{R}_2, ..., \mathbb{R}_m)$ is the resource availability list. $\mathbb{T}_i \in V(1 \leq i \leq n)$ denotes the index of the i-th activity in $\mathbb{T}$. In the activity list the activities at the front have higher priorities than the activities behind.

3.2. Feasible solution generation

The standard serial scheduling scheme (SGS) dated to Kelley [11] is adopted to generate a solution for RACP problem. For solution $\pi = (\mathbb{T}, \mathbb{R})$, under the resource availability constraint of $\mathbb{R}$, a schedule is generated by sequentially choosing activities in the order of $\mathbb{T}$ and ensuring that each activity can start as early as possible. In fact, a schedule can be regarded as a start time list of all the activities. When an new activity is added into the current partial

schedule or the start time of an activity in the current schedule is changed, the resource usage variable $a_k(t)$ is adapted accordingly.

If activity $\mathbb{T}_i$ is added to the current schedule at the start time $st_{\mathbb{T}_i}$, the current $a_k(t)$ will be changed by $a_k(t) \leftarrow a_k(t) + r_k^{\mathbb{T}_i}$, $t \in [st_{\mathbb{T}_i}, st_{\mathbb{T}_i} + d_{\mathbb{T}_i}]$, $k \in M$. It means the resource usage increases by $R^{\mathbb{T}_i}$ during the time slot $[st_{\mathbb{T}_i},\ st_{\mathbb{T}_i} + d_{\mathbb{T}_i}]$. If activity $\mathbb{T}_i$ which starts at time $st_{\mathbb{T}_i}$ is removed from the current schedule, $a_k(t)$ will be changed by $a_k(t) \leftarrow a_k(t) - r_k^{\mathbb{T}_i}$, $t \in [st_{\mathbb{T}_i}, st_{\mathbb{T}_i} + d_{\mathbb{T}_i}]$, $k \in M$. It means the resource usage decreases by $R^{\mathbb{T}_i}$ during the time slot $[st_{\mathbb{T}_i},\ st_{\mathbb{T}_i} + d_{\mathbb{T}_i}]$. Additionally, if activity $\mathbb{T}_i$ which starts at time $st_{\mathbb{T}_i}$ is postponed to start at time $st'_{\mathbb{T}_i}$, $a_k(t)$ will be changed by

$$a_k(t) \leftarrow \begin{cases} a_k(t) - r_k^{\mathbb{T}_i} & t \in [st_{\mathbb{T}_i}, st_{\mathbb{T}_i} + d_{\mathbb{T}_i}], k \in G \\ a_k(t) + r_k^{\mathbb{T}_i} & t \in [st'_{\mathbb{T}_i}, st'_{\mathbb{T}_i} + d_{\mathbb{T}_i}], k \in G \end{cases}$$

It means the resource usage decreases during the time slot $[st_{\mathbb{T}_i},\ st_{\mathbb{T}_i} + d_{\mathbb{T}_i}]$ and increases during the time slot $[st'_{\mathbb{T}_i},\ st'_{\mathbb{T}_i} + d_{\mathbb{T}_i}]$ by $R^{\mathbb{T}_i}$. After all the activities of $\pi = (\mathbb{T}, \mathbb{R})$ have been scheduled, the resource usage variables for m types in all the time slots of the project are obtained. The pseudocode for SGS is described in Algorithm 1.

For a better understanding, the serial scheduling scheme of a solution $\pi = (\mathbb{T}, \mathbb{R})$ of the project in Figure 1 is numerically illustrated as follows: Suppose the activity list of π is $\mathbb{T} = (1, 2, 7, 9, 3, 5, 8, 4, 6, 10)$ and the resource availability list is $\mathbb{R} = (20, 19)$. The earliest feasible start time of each activity $\mathbb{T}_i$ is calculated as $est_{\mathbb{T}_i}$. Initialize $a_k(t) \leftarrow 0$ for all $k \in M$ and $t \in [1, ..., H]$. When $i = 2$, $\mathbb{T}_2 = 2$, the earliest feasible start time of activity 2 in time slot $[0, 56]$ under constraint $\mathbb{R}$ is $w \leftarrow 0$. So $st_2 \leftarrow 0$ and $a_1(t) = 3$, $a_2(t) = 1$, $t \in [1, 8]$. Activity 7 is the successor of activity 2. Since $est_7 = 0 < 8$, then $est_7 \leftarrow 8$. When $i = 3$, $\mathbb{T}_3 = 7$, the earliest feasible start time of activity 7 in time slot $[8, 56]$ under constraint $\mathbb{R}$ is $w \leftarrow 8$. So $st_7 \leftarrow 8$ and $a_1(t) = 7$, $a_2(t) = 6$, $t \in [9, 14]$. Activity 9 and 8 are two successors of activity 7. Since $est_9 = 0 < 14$, $est_8 = 0 < 14$, then $est_9 \leftarrow 14$, $est_8 \leftarrow 14$. The rest i $(= 4, 5, \ldots, 10)$ can be done in the same manner. As a result, the earliest feasible start time of each activity is changed into $est_1 = est_2 = est_3 = 0$, $est_4 = est_5 = est_7 = 8$, $est_6 = 25$, $est_8 = 16$, $est_9 = 14$, $est_{10} = 35$. The final resource usage variables are obtained and showed in Table 1. The schedule derived from π is shown in Figure 3(a). The Makespan of this schedule is 35. It is worth noticing that when $i = 8$, $\mathbb{T}_8 = 4$, the earliest feasible start time of activity 4 in time slot $[8, 56]$ under constraint $\mathbb{R}$ is neither 8 nor 16, but 19.

A special case of Algorithm 1 is when deciding the start time of an activity under no resource constraint and thus a fast schedule is generated. In the fast schedule, all activities are started as early as possible leading to maximum parallelism. Therefore, the resource availability vector of a fast schedule is the upper bound of resource availabilities to complete the project which can be defined as $\bar{A}$. For a RACP project, since different activity lists result in the same fast schedule, $\bar{A}$ can be calculated in advance.

If the makespan of the newly generated schedule is not greater than the deadline, it is called a feasible schedule. If a solution can produce a feasible schedule, it is called a feasible solution. The feasibility of a solution can be attempted by the forward-backward-procedure (FBP) [6]. The FBP algorithm is also adjusted with the necessary adaptations

Table 1: The final value of resource usage variable $a_k(t)$.

$t \in$	$k=1$	$k=2$
$[1,8]$	13	4
$[9,14]$	14	7
$[15,16]$	13	3
$[17,19]$	7	11
$[20,21]$	13	12
$[22,25]$	7	9
$[26,35]$	5	3
$[36,56]$	0	0

Algorithm 1: Serial Generation Scheme (SGS) (π)

begin
  Initialize the resource usage variable $a_k(t) \leftarrow 0$ for all $k \in G$ and $t \in [1, ..., H]$;
  for *each activity* $\mathbb{T}_i \in \mathbb{T}$ **do**
    $est_{\mathbb{T}_i} \leftarrow 0$;
  for $i = 1$ **to** n **do**
    Find the earliest feasible start time w of activity $\mathbb{T}_i$ under the constraint $\mathbb{R}$ in $[est_{\mathbb{T}_i}, H]$;
    Add $\mathbb{T}_i$ to the current schedule at the start time w ;
    for $k = 1$ **to** m **do**
      for $t = w$ **to** $w + d_{\mathbb{T}_i}$ **do**
        $a_k(t) \leftarrow a_k(t) + r_k^{\mathbb{T}_i}$;
    $st_{\mathbb{T}_i} \leftarrow w$;
    for *each activity* $j \in succ_{\mathbb{T}_i}$ **do**
      if $est_j < (st_{\mathbb{T}_i} + d_{\mathbb{T}_i})$ **then**
        $est_j \leftarrow (st_{\mathbb{T}_i} + d_{\mathbb{T}_i})$;

to work with the proposed solution representation. In FBP, after constructing a schedule from π by the SGS (Algorithm 1), a backward adjustment of all activities begins. Under the resource availability constraint of $\mathbb{R}$, the schedule is adjusted by choosing activities in the reverse order of $\mathbb{T}$ and postponing them until as late as possible. $a_k(t)$ is updated along with the adjustment process. After all the activities have been adjusted the makespan is recalculated. If the makespan is not greater than D the solution is feasible, otherwise the solution is unfeasible. The pseudocode of FBP is described in Algorithm 2.

The backward adjustment process conducted on the schedule in Figure 3(a) can be illustrated as follows: For each activity $\mathbb{T}_i \in \mathbb{T}$, initialize $lst_{\mathbb{T}_i} \leftarrow 35$. When $i = 9$, $\mathbb{T}_9 = 6$, the latest feasible start time of activity 6 in time slot $[25, 35]$ under constraint $\mathbb{R} = (20, 19)$ is 25. So st_6 and $a_k(t)$ keep their original values. Activity 4 is the predecessor of activity 6.

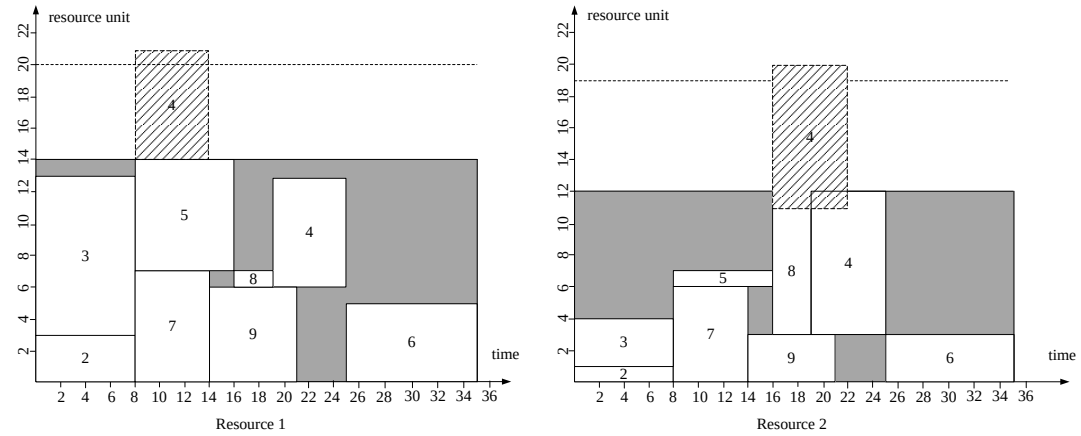
Algorithm 2: Forward Backward Procedure (FBP) (π)

```
begin
    Call SGS(π) to generate a schedule;
    flag ← false;
    for each activity 𝕋_i ∈ 𝕋 do
        lst_{𝕋_i} ← st_{𝕋_n};
    for i = n to 1 do
        Find the latest feasible start time w of activity 𝕋_i under the constraint ℝ in
        [st_{𝕋_i}, lst_{𝕋_i}];
        Start time of activity 𝕋_i is postponed from st_{𝕋_i} to w ;
        for k = 1 to m do
            for t = st_{𝕋_i} to st_{𝕋_i} + d_{𝕋_i} do
                a_k(t) ← a_k(t) − r_k^{𝕋_i} ;
            for t = w to w + d_{𝕋_i} do
                a_k(t) ← a_k(t) + r_k^{𝕋_i} ;
        st_{𝕋_i} ← w;
        for each activity j ∈ pred_{𝕋_i} do
            if lst_j > (st_{𝕋_i} − d_j) then
                lst_j ← (st_{𝕋_i} − d_j) ;
    if ((st_{𝕋_n} + d_{𝕋_n}) − st_{𝕋_1}) ≤ D then
        flag ← true;
    return flag.
```

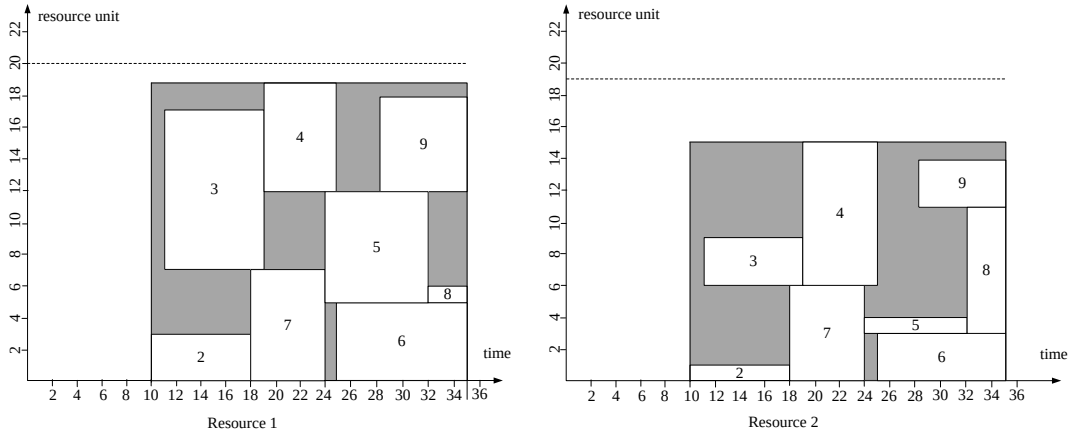
Since $lst_4 = 35 > (25 - 6)$, then $lst_4 \leftarrow 19$. When $i = 8$, $\mathbb{T}_8 = 4$, the latest feasible start time of activity 4 in time slot $[19, 35]$ is 19. So st_4 and $a_k(t)$ keeps the original value. Activity 3 is the predecessor of activity 4. Since $lst_3 = 35 > (19 - 8)$, then $lst_3 \leftarrow 11$. When $i = 7$, $\mathbb{T}_7 = 8$, the latest feasible start time of activity 8 in time slot $[16, 35]$ is 32. So $st_8 \leftarrow 32$. Then $a_1(t) \leftarrow (7 - 1) = 6$, $a_2(t) \leftarrow (11 - 8) = 3$, $t \in [17, 19]$, and $a_1(t) \leftarrow (5 + 1) = 6$, $a_2(t) \leftarrow (3 + 8) = 11$, $t \in [33, 35]$. The rest i $(= 6, 5, \ldots, 1)$ can be done in the same manner. The final resource usage variables are obtained and shown in Table 2. The adjusted schedule is shown in Figure 3(b). Since $((st_{\mathbb{T}_n} + d_{\mathbb{T}_n}) - st_{\mathbb{T}_1}) = (35 - 10) = 25 < 28$, then solution $\pi = \big((1, 2, 7, 9, 3, 5, 8, 4, 6, 10), (20, 19)\big)$ is feasible.

4. Multi-start iterative search algorithm

In this paper, a multi-start iterative search algorithm (MSIS) is developed to solve the RACP. The procedure consists of two processes: generation of the sequence of activities and the determination of the resources requirements. All the activity lists produced in each step of the algorithm are guaranteed to be precedence feasible. The main idea of MSIS is as



(a) A schedule generated by conducting serial scheduling scheme.



(b) The adjusted schedule after conducting backward process.

Figure 3: An example of FBP process.

follows: First, N_{pop} initial start points are constructed by the initial start points generation strategy (ISPG) in which the priority rule is adopted. An elite pool, which is initially empty, is built to reserve N_{elite} best found solutions. The following processes are repeated until the termination criterion is satisfied. The start points are assigned as the current solutions. All the current solutions are improved by the local search method (LS). The elite pool is updated by adding the current solutions by the given acceptance rules. The best solution of the elite pool is reserved as the global best solution. If any of the current solutions can be added into the elite pool, path-relinking (PR) is conducted on all pairs of solutions in the elite pool for further exploitation. The good solutions found among all the paths are reserved and enhanced by the local search method. The elite pool is updated by adding the good solutions by the given acceptance rules. The global best solution is updated by the new best solution of the elite pool. If the elite pool or the global best solution is not updated for a given number of generations, all the current solutions are refreshed by the restart strategy. If it is not the case, the new start point generation strategy (NSPG) is pair-wisely applied

Table 2: The final value of resource usage variable $a_k(t)$ after FBP process.

$t \in$	$k=1$	$k=2$
$[1,10]$	0	0
$[11,11]$	3	1
$[12,18]$	13	4
$[19,19]$	17	9
$[20,24]$	14	15
$[25,25]$	14	10
$[26,28]$	12	4
$[29,32]$	18	7
$[33,35]$	12	14

to the current solutions to produce N_{pop} new start points for the next iteration. Finally, the global best solution is returned. The framework of the proposed algorithm is depicted in Algorithm 3 and further details of the proposed operators are introduced in the following subsections.

Algorithm 3: Multi-Start Iterative Search algorithm (MSIS)

```
begin
    Initialize N_pop feasible start points by ISPG and an empty elite pool ;
    repeat
        Assign the start points as the current solutions;
        Improve the current solutions by LS;
        Update the elite pool by adding the current solutions by the acceptance rules;
        Reserve the best solution of the elite pool as the global best solution;
        if the elite pool is updated then
            Apply PR to all solutions of the elite pool pair-wisely to generate new
            good solutions;
            Improve the new good solutions by LS;
            Update the elite pool by adding the new good solutions by the acceptance
            rules;
            Update the global best solution by the new best solution in the elite pool;
        if the elite pool or the global best solution is not updated for a given number of
        generations then
            Generate N_pop new start points by the restart strategy;
        else
            Apply NSPG to the current solutions pair-wisely to produce N_pop new start
            points;
    until (Termination criterion is satisfied);
    return the global best solution.
```

As mentioned before, the sequencing sub-problem of the RACP is actually a permutation optimization problem with a precedence constraint, whose solution space is composed of precedence feasible activity lists. For the general permutation optimization problem, the solution space can be regarded as a continuous area and the local optima spread in this area. However, for the permutation optimization problem with precedence constraints, the solution space is not continuous but consists of many isolated islands made of feasible solutions [10]. The remaining areas among these islands are non-feasible solutions. Local optima lie on the islands. Therefore, designing some effective search strategies to explore and exploit these islands is important for solving the considered problem. In this paper, two different search approaches LS and PR are proposed for exploitation enhancement and a new start points generation method, NSPG, is presented for exploration enhancement. Taking advantage of these strategies, the solution space can be sufficiently searched.

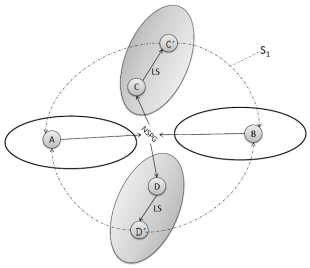


Figure 4: Search space by new start points generation and local search.

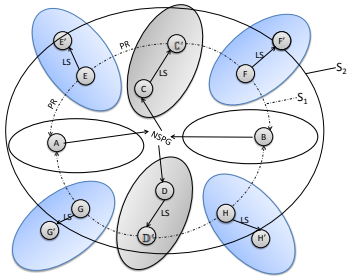


Figure 5: Extended search space by path-relinking.

Figure 4 illustrates the solution space by NSPG and LS processes. Points A and B are two local optima found in the last iteration, points C and D are new start points generated by conducting NSPG on points A and B. Taking C and D as the centers, local optima C' and D' are found respectively. The coverage of the search is S_1. Nevertheless, the search space can be extended by path-relinking as shown in Figure 5. Path-relinking is applied between the new local optima C' and D' and the previous ones A and B. During the path-relinking procedure, all candidate solutions on the path $C' \to A$, $C' \to B$, $D' \to A$, $D' \to B$ are accessed and the new good solutions E, F, G, H are obtained to conduct the local search. Then, the new local optima E', F', G', H' are produced. As a result, the coverage of the search is extended to S_2. Start points for the next iteration are generated by conducting NSPG on these new local optima. Some points lie within S_2 and some lie outside S_2. The search starting from the former can improve exploitation and the latter can enhance exploration.

4.1. Sequencing problem

The sequencing problem is to determine the scheduling order of activities. After a scheduling order of activities has been determined, the solution of the RACP can be easily obtained by applying a heuristic for the resource decision problem (in Sect.4.2).

4.1.1. ISPG: Initial start points generation strategy

Since the RCPSP aims at minimizing the makespan of a project subject to precedence relations among activities and limited renewable resource availabilities, the minimization of makespan and resource usage of the project should be considered in the design of activity priorities. Likewise, these priorities can also be used to generate initial activity lists for the sequencing problem of the RACP. Demeulemeester and Herroelen [3] classified the priority rules for the RCPSP into 5 categories and analyzed the most commonly used rules in each category. Yang [26] compared 13 priority rules. Computational results showed that rules GCRR (Greatest Cumulative Resource Requirement), RPW (Rank Positional Weight) and MAS (Minimum Activity Slack) were superior to the others in most cases. Chtourou and Haouari [1] tested 16 priority rules and concluded that rule RPW outperformed the others. In this paper, three priority rules GCRR, RPW and LFT (Latest Finish Time of activity), which are commonly used for RCPSP, are adopted to generate the initial activity lists $\mathbb{T}^o$ of the RACP. Because the proposed algorithm is based on multi-start points, the priority rules are modified by combining a randomly generated number. The new priority of activity j is depicted as:

$$PI'(j) = PI(j) + rand \tag{9}$$

where $PI \in \{GCRR, RPW, LFT\}$ is the priority rule for the RCPSP, $rand$ is a random number uniformly distributed in the range $[0, PI^{\max}]$, and $PI^{\max}$ is the maximum possible value of priority PI. The final three priority rules are denoted as $PI' \in \{GCRRR, RPWR, LFTR\}$. Tests are made on these three rules and a fully random priority (RAN) and the best one is chosen to generate $\mathbb{T}^o$, see Sec.5.3 for more details.

Based on the initial activity lists $\mathbb{T}^o$, a fast schedule can be constructed in advance. Then, a feasible initial solution $\pi = (\mathbb{T}^o, \mathbb{R}^o)$ is obtained, where $\mathbb{R}^o$ is the upper bound of resource availabilities, i.e. $\bar{A}$. After that, the Backward Path Elimination Procedure (BPEP) (in Sec.4.2) is conducted on π to adjust the resource availability. The resulting solution is regarded as a feasible start point for the algorithm. The pseoducode of the initial start points generation process is depicted in Algorithm 4, where POP is the set of initial start points of size $\mathcal{N}_{pop}$.

4.1.2. Feasible neighbourhood and local search

Let $M(i, j)$ $(1 < i < n, 1 < j < n, i \neq j, i \neq j-1)$ be a move (also called shift [9]) operation which indicates the movement of the activity at position i of an activity list to position j. After applying $M(i, j)$ to a feasible solution $\pi = (\mathbb{T}, \mathbb{R})$, a new solution $\pi' = (\mathbb{T}', \mathbb{R})$ is generated. In order to directly generate a precedence feasible activity list $\mathbb{T}'$, the move operation $M(i, j)$ is redefined as follows: For the activity $\mathbb{T}_i$ in the i-th position of the activity list $\mathbb{T}$, the largest position p of its immediate predecessors and the smallest position q of its immediate successors are calculated. A random integer $j \in (p, q)\,(j \neq i)$, is generated. Then, activity $\mathbb{T}_i$ is moved to position j to generate a new activity list. Obviously, the new activity list is precedence feasible.

On each start point π, a VNS [16] based local search (LS) is conducted to search for the local optimum. For a feasible solution $\pi = (\mathbb{T}, \mathbb{R})$, the neighbourhood $N_r(\pi)$ is defined as

Algorithm 4: Initial Start Points Generation, ISPG (POP)

```
1  begin
2      while (|POP| ≤ N_pop) do
3          Calculate the priority of each activity according to the priority rule PI' ;
4          while activity set V is not empty do
5              The activity with no predecessor in AON network and maximal priority is
               removed from V and added into T^0 ;
6          R^0 ← Ā, π ← (T^0, R^0);
7          Call BPEP(π) to allocate resources;
8          if π ∉ POP then
9              POP ← POP ∪ {π};
10     return POP.
```

a solution set where each solution is obtained by applying $M(i,j)$ to π r times and where $r = 1, 2, \ldots, K$, K is the maximum neighbourhood number and i is a random integer in $(1, n)$. For example, suppose the activity list of a solution π of the project in Figure 1 is $\mathbb{T} = (1, 2, 7, 9, 3, 5, 8, 4, 6, 10)$. If $r = 2$, $M(i,j)$ is applied to π twice to create a neighbour solution. In the first round, assume $i = 7 \in (1, 10)$, i.e. activity 8 is selected. For activity 8, two immediate predecessors are activity 5 in position 6 and activity 7 in position 3, thus $p = 6$. The immediate successor is activity 10 in position 10, thus $q = 10$. Let $j = 9 \in (6, 10)$, then activity 8 is moved to position 9 to generate the new activity list $\mathbb{T} = (1, 2, 7, 9, 3, 5, 4, 6, 8, 10)$. In the second round, assume $i = 6 \in (1, 10)$, i.e. activity 5 is selected. For activity 5, the immediate predecessor is activity 3 in position 5 and the immediate successor is activity 8 in position 9, thus $p = 5, q = 9$. Let $j = 7 \in (5, 9)$, then activity 5 is moved to position 7. Finally, the activity list of the neighbour solution is $\mathbb{T} = (1, 2, 9, 3, 5, 4, 7, 6, 8, 10)$.

The local search process can be described as follows: Initially set $r \leftarrow 1$. Move operation $M(i,j)$ is applied to the current solution π to create its neighbourhood $N_r(\pi)$. Solution π' is randomly chosen from $N_r(\pi)$ and a feasibility verification is conducted by FBP. If π' is feasible, a two stage resource adjustment procedure (TSRAP, in Sec.4.2) is adopted to adjust the resource usage of π'. If π' is better than π, then $\pi \leftarrow \pi'$ and set $r \leftarrow 1$. The processes above is repeated until there is no feasible solution from $N_r(\pi)$ or no improvement on π for a number of λ (the maximum number of attempts) iterations, then $r \leftarrow r + 1$ and the next iteration starts. All steps above are repeated until $r = K$. The pseudocode of local search is described in Algorithm 5.

4.1.3. PR: Path-relinking

Path-relinking was originally proposed by Glover et al. [8] as an approach to integrate intensification and diversification strategies in tabu search. It explores the trajectories linking two high quality solutions, starting from an initial solution (π^s) and moving towards a guiding solution (π^g). This is accomplished by selecting moves that introduce attributes

Algorithm 5: Local Search, LS (π, λ, K)

```
1  begin
2      r ← 0;
3      while r ≤ K do
4          r ← r + 1;
5          for i = 1 to λ do
6              π′ ← randomly choose a solution from N_r(π);
7              feasible ← FBP(π′);
8              if feasible then
9                  Call TSRAP(π′) to allocate resources;
10                 if C(ℝ′) < C(ℝ) then
11                     π ← π′, r ← 0;
12                     Goto Step 4;
13     return π.
```

contained in the guiding solution. In this paper, path-relinking is used to search for promising solutions on the paths between the current local optima and some of the previous local optima. This strategy is used to find high quality solutions that have been missed in the previous searches efficiently and to help exploit the solution space. In order to reduce the time consumption, path-relinking is applied only between all pairs of solutions in the elite pool. The pseudocode of path-relinking is presented in Algorithm 6.

Algorithm 6: Path Relinking, PR (π^s, π^g)

```
1  begin
2      min ← +∞, π* ← ∅;
3      for i = 1 to n do
4          if 𝕋^s_i ≠ 𝕋^g_i then
5              Find the position p of activity 𝕋^g_i in 𝕋^s, remove activity 𝕋^s_p from 𝕋^s and
               insert into position i;
6              ℝ^s ← Ā, π ← (𝕋^s, ℝ^s);
7              Call BPEP(π) to allocate resources;
8              if C(π) < min then
9                  min ← C(π), π* ← π;
10     return π*.
```

Theorem 1 For a pair of precedence feasible activity lists, the path-relinking described in Algorithm 6 creates precedence feasible activity lists.

Proof. Let $\mathbb{T}^s = (a_1^s, a_2^s, ..., a_n^s)$ and $\mathbb{T}^g = (a_1^g, a_2^g, ..., a_n^g)$ be a pair of precedence feasible activity

lists. Apply the path-relinking to them from left to right. Suppose that $\mathbb{T}^s_p \neq \mathbb{T}^g_p$, $1 \leq p \leq n$. There exists a position q, such that $\mathbb{T}^s_q = \mathbb{T}^g_p$ $(1 \leq q \leq n)$. Obviously, $\mathbb{T}^s_i = \mathbb{T}^g_i$, $1 \leq i < p$. Activities $\mathbb{T}^s_j$ $(p \leq j < q)$ are behind activity $\mathbb{T}^s_q$ in $\mathbb{T}^g$ and ahead of $\mathbb{T}^s_q$ in $\mathbb{T}^s$, so no precedence relations exist between them. When activity $\mathbb{T}^s_q$ is moved to position p, neither the relative precedence relations between activities $\mathbb{T}^s_k$ $(q < k \leq n)$ and activity $\mathbb{T}^s_q$ nor the relative precedence relations between activities $\mathbb{T}^s_k$ $(q < k \leq n)$ and activity $\mathbb{T}^s_j$ are changed. Therefore, the new activity list obtained is precedence feasible. □

4.1.4. NSPG: New start points generation strategy

The two-point crossover operator by Shadrokh and Kianfar [21], which has been proven to be able to create precedence feasible activity lists for a pair of precedence feasible parents is used to generate new start points. In the two-point crossover process, two random cutting sites cs_1 and cs_2 are drawn with $1 \leq cs_1 < cs_2 \leq n$ and with n the length of the parent sequences. The sequence of child1 is then generated by copying the first cs_1 and the last $n - cs_2$ positions from the parent 2. The remaining positions are copied from the parent 1. The sequence of the child 2 is constructed in a similar way, copying the first and third part from the parent1 and obtaining the middle part from the parent 2. The crossover procedure is illustrated in Figure 6. The generation process of new start points is shown in Algorithm 7. Each pair of local optima π^f and π^m found in the current iteration are combined to generate a pair of new start points π^s and π^d. If the number of different new start points generated is greater than $\mathcal{N}_{pop}$, only the best $\mathcal{N}_{pop}$ start points will be reserved. Otherwise, the remaining start points are supplemented by the ISPG method described in Algorithm 4.

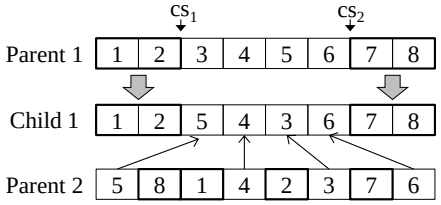


Figure 6: An illustration of two-point crossover.

Algorithm 7: New Start Points Generation, NSPG (π^f, π^m)

```
1 begin
2     (T^s, T^d) ← Conduct two-point crossover between T^f and T^m;
3     X ← (s, d);
4     for h = 1 to 2 do
5         R^{X_h} ← Ā, π^{X_h} ← (T^{X_h}, R^{X_h});
6         Call BPEP(π^{X_h}) to allocate resources;
7     return (π^s, π^d).
```

4.1.5. Acceptance rules

Each local optimal solution obtained during the search process is considered as a candidate to be inserted into the elite pool. The usual rules for inclusion of a new solution into the elite set are also adopted in the proposed algorithm. The pool is initially empty with the maximum size N_{elite}. If the pool is not yet full, a candidate is accepted if it differs from all other solutions in the pool. Otherwise, the worst solution in the pool is replaced by a candidate if the former is worse than the latter. Two solutions are called "different" if the permutation of activities in their activity lists are not the same.

4.1.6. Restart strategy

After a number of iterations, the algorithm may be trapped in a local optima solution. In order to avoid repetitive explorations of the same paths in the solution space and to explore new solutions, it is necessary to enhance the diversification by refreshing the elite pool with new solutions. Therefore, a restart strategy is introduced into the proposed algorithm. If the elite pool has not been updated for δ continuous iterations or the best solution found remains unchanged for $2 * \delta$ continuous iterations, the current solutions will be refreshed by regenerating N_{pop} new start points by ISPG.

4.2. Resource decision problem

After determining the activity list schedule, the resource availabilities have to be calculated. A backward peak elimination procedure (BPEP) and a two stage resource adjustment procedure (TSRAP) are proposed for the resource decision problem.

Let $\underline{A} = (\underline{A}_1, \underline{A}_2, ..., \underline{A}_m)$ be the lower bound of resource availabilities to complete the project. During the adjustment of the resource availability, the availability of resource of the k-th type cannot be less than $\underline{A}_k$. A simple definition of $\underline{A}_k$ $(k = 1, \ldots, m)$ is given as [7]:

$$\underline{A}_k = \max \left\{ \max_{i \in V} \{r_k^i\}, \left\lceil \frac{\sum_{i \in V} d_i \times r_k^i}{\min\{D, \sum_{i \in V, r_k^i > 0} d_i\}} \right\rceil \right\} \tag{10}$$

4.2.1. BPEP: Backward peak elimination procedure

The serial scheduling scheme described in Algorithm 1 may overuse a resource which means the resource usage concentrates on a small period of the project cycle. A resource usage peak which determines the resource availability cost of the project will arise in this period. BPEP is constructed to distribute resource usage in the project cycle to reduce the resource usage peak by moving activities backward according to a defined strategy.

Suppose that a schedule of solution $\pi = (\mathbb{T}, \mathbb{R})$ has been generated already by Algorithm 1. Activities are moved backward in the reversed order of $\mathbb{T}$ subject to the reduction of resource availabilities $\mathbb{R}$. For each time point between st_i and lst_i of activity i, the decrement of resource availability cost caused by moving activity i to that time point is calculated. When an activity is moved, the resource usage variable $a_k(t)$ is updated and the corresponding resource availability vector A is recalculated. The decrement of resource availability cost is calculated by:

$$\mathcal{C}_{dre} = \sum_{k=1}^{m} c_k \times (A_k - \mathbb{R}_k) \tag{11}$$

Activity i will be moved to the latest time point with maximum decrement of resource availability cost. The BPEP process is described in Algorithm 8.

Figure 7 shows a schedule created by conducting BPEP on the schedule shown in Figure 2. The new resource availability vector $A = (18, 15)$ and the resource availability cost $\mathcal{C}(A) = 156$. The new schedule has a cost reduction of 14 from the original cost of 170.

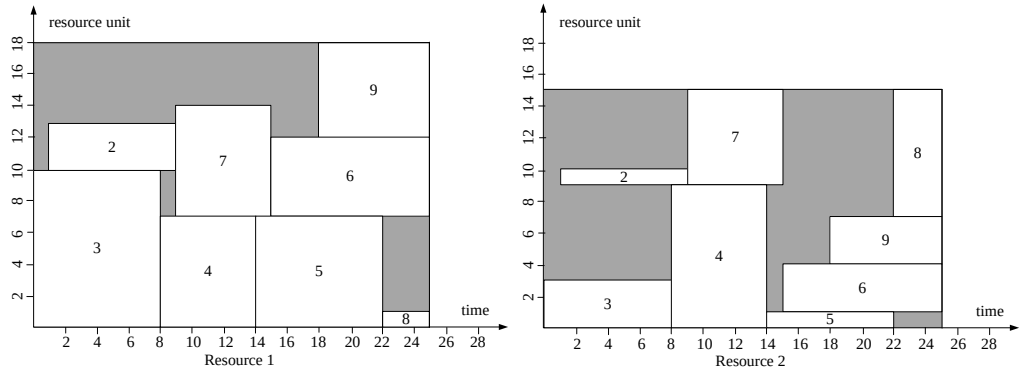


Figure 7: A new schedule generated by conducting BPEP on the schedule in Figure 2.

4.2.2. TSRAP: Two stage resource adjustment procedure

In most existing work, the resource availability cost is reduced by decreasing the availability of each resource progressively. However, the fact that the availability increment of some resources can decrease the availability of other resources is usually ignored. This situation may further reduce the resource availability cost of the project. In this paper, both of these two situations are considered and a two stage resource adjustment procedure (TSRAP) is constructed.

TSRAP consists of two stages: 1) reduction of resource availability cost (RRAC), that is, the availability of resource of each type declines; 2) fluctuant decline of resource availability cost (FDRAC), which is accomplished by decreasing the availability of expensive resources and increasing the availability of cheap resources.

In the RRAC stage, all resources are sorted in descending order of their costs. For a solution $\pi = (\mathbb{T}, \mathbb{R})$, the availability of resources of the k-th type is decreased by one unit and the FBP method is used to verify the feasibility of the solution. If it is feasible, the procedure is repeated. Otherwise, the availability of resources of the k-th type is increased by one unit and the resource of the $(k+1)$-th type is checked. Finally, after all the resources have been considered, the BPEP is applied to improve the solution.

The FDRAC stage is processed after RRAC. For a solution $\pi = (\mathbb{T}, \mathbb{R})$, the initial resource availability $\mathbb{R}'$ is obtained by setting the availability of resources of the k-th type to be $\mathbb{R}_k$ and the others to be the lower bound values (Eq. 10). Activities in $\mathbb{T}$ are sequentially scheduled under the restriction that $\mathbb{R}'_k \leq \mathbb{R}_k$. For each time point between est_j and the latest start time by the forward and backward pass method [3] of activity j, which are calculated without considering the resource availabilities, the increment of resource availability cost caused by scheduling activity j is calculated. When an activity is scheduled, the resource

usage variable $a_k(t)$ is updated and the new resource availability vector A is obtained. The increment of resource availability cost is computed as

$$\mathcal{C}_{incre} = \sum_{k=1}^{m} c_k \times \max(0, A_k - \mathbb{R}'_k) \tag{12}$$

If scheduling activity j at each time point causes an overuse of resources, i.e., $A_k > \mathbb{R}_k$ or $\mathcal{C}(A) > \mathcal{C}(\mathbb{R})$, the algorithm stops. Otherwise, activity j is scheduled at the earliest time point with the least cost increment. If the resource availability vector A changes, $\mathbb{R}'$ is updated. The activities are moved as early as possible between their $est(s)$ and current start times. Meanwhile the $est(s)$ of their immediate successors are also updated. For solution π with restriction on the availability of resources of the k-th type, the FDRAC process can be described in Algorithm 9.

Figure 8 displays another schedule for the project of Figure 1. The resource availability vector is $A = (14, 14)$ and the resource availability cost is $\mathcal{C}(A) = 140$. After FDRAC, the schedule is changed as shown in Figure 9. The new resource availability vector is $A = (19, 11)$ and the new resource availability cost is $\mathcal{C}(A) = 126$. Hence, the new schedule has a cost reduction of 14. The resource availability cost of the project in Figure 8 cannot be reduced by RRAC. However it can be reduced by FDRAC through decreasing the availability of expensive resources and increasing the availability of cheaper resources.

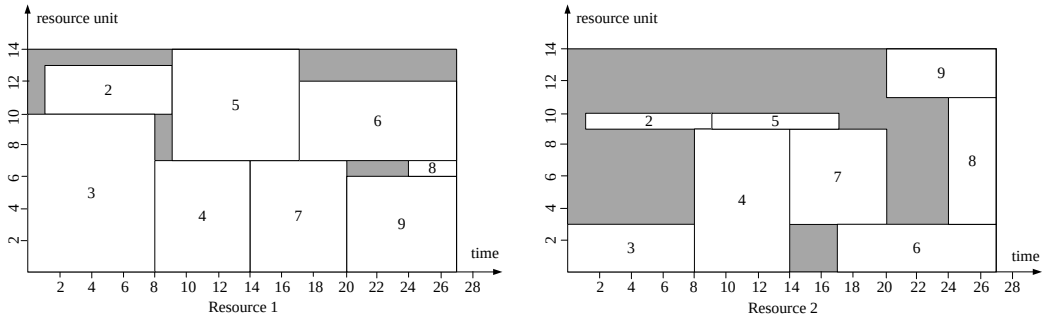


Figure 8: Another schedule of the project in Figure 1.

In TSRAP, these two stages are repeated until neither of them can reduce the resource availability cost anymore and then they stop. The pseudocode of TSRAP is described in Algorithm 10.

4.2.3. Overview of the entire algorithm

An overview of the entire heuristic (MSIS) is provided as follows. The heuristic consists of two processes (problems): generation of the sequencing of activities and the determination of resource requirements. The sub-procedures initial start point generation (ISPG), Local Search (LS), Path-relinking (PR), restart strategy and new start point generation (NSPG) are designed for solving the sequencing problem. By these sub-procedures, sequences of activities are generated. The sub-procedures Backward peak elimination procedure (BPEP) and two stage resource adjustment procedure (TSRAP) are concentrated on the resourcing

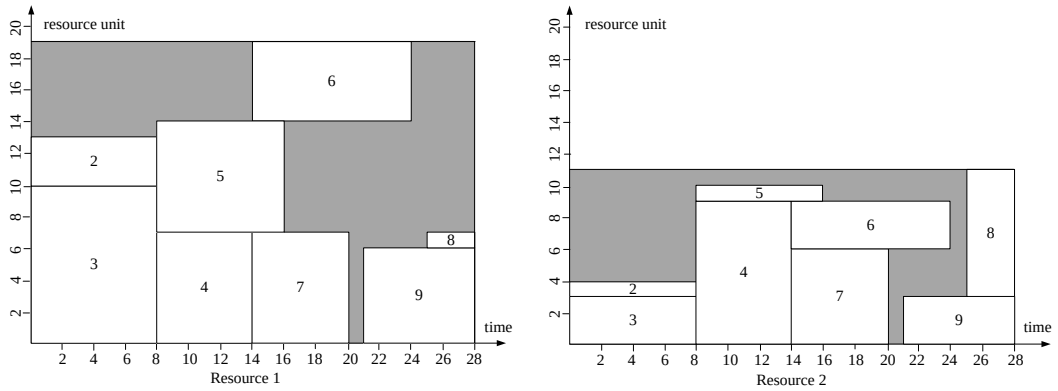


Figure 9: A new schedule generated by conducting FDRAC on the schedule in Figure 8.

problem. In both BPEP and TSRAP, the evaluated solutions are constructed by assigning values to the resource availability variables and executing the improvement heuristic in order to obtain the makespan. The BPEP is embedded in ISPG, PR and NSPG to improve the solutions and diversify the population to a certain extent. The TSRAP is embedded only in the local search procedure to enhance the intensification of the algorithm. In addition, the serial generation scheme (SGS) is used in ISPG, the forward and backward procedure (FBP) and BPEP to forwardly generate a schedule, and FBP is adopted to verify the feasibility of a solution.

5. Computational results

To test the performance of the proposed algorithm, it is compared with the best-existing algorithms for the considered problem: the scatter search by Yamashita et al. [25] and the PR and GA by Ranjbar et al. [19]. Additionally, different operators and levels of the parameters of the proposed algorithm are calibrated by the DOE (Design of Experiments) approach.

5.1. Instances generation

There are no existing benchmarks for the RACP. The well-known PSPLib instances for the RCPSP have been adapted to this problem in most of the existing work [25],[21],[19]. In this paper, experiments are also carried out on the well-known PSPLib instances [13] with $n = 30, 60, 90, 120$ activities and $m = 4$ types of resources. When constructing a project, two important factors are considered: the network complexity NC and the resource factor RF. $NC \in \{1.5, 1.8, 2.1\}$ is the average number of immediate successors of an activity. $RF \in \{0.25, 0.5, 0.75, 1\}$ is the density of resources of different types needed by an activity. For each combination of NC and RF, 20 instances are generated. 10 of the instances are used for parameter calibration and start points generation analysis and the rest are for performance comparison. According to Drexl and Kimms [7], the deadline is defined as $D = \theta \times est_n$, where θ is a deadline factor set as 1.2 and est_n is the earliest start time of the last dummy activity computing by the forward and backward pass method [3]. The costs of the resources are

randomly drawn from a uniform distribution $U[1, 10]$. Therefore, in total $4 \times 3 \times 4 \times 20 = 960$ PSPLib instances are created.

Demeulemeester et al. [5] designed a random network generator (RanGen) to generate AON networks and accompanying data for different classes of project scheduling problems. Experiments prove that RanGen avoids the shortcoming of other existing network generators. Van Peteghem and Vanhoucke [23] have used RanGen to generate instances for the RACP. In this paper, RanGen is also adopted to generate instances for performance comparison. Instances with $n = 30$ activities and $m = 4, 6, 8$ types resources are generated and parameter $OS \in \{0.25, 0.5, 0.75\}$ indicates the network complexity. For each combination of OS and RF, five instances are used. The deadline factor θ takes the values 1.1,1.2,1.3,1.4,1.5. The costs of the resources are also randomly drawn from $U[1, 10]$. Hence, there are $3 \times 4 \times 3 \times 5 \times 5 = 900$ RanGen instances generated in total.

5.2. Termination criterion and performance measure

To fairly compare the performance of the proposed algorithm to known techniques from the literature, all the compared methods are re-implemented and share most functions. All algorithms are coded in Java and run on a PC with 3.4 GHZ Intel Core i7 processor and 1G RAM. For a fair comparison, the number of schedules generated is adopted as the termination criterion [12] as it is commonly used in the project scheduling literature. In our experiments, the termination criterion is set as 1000, 5000 and 10000 generated schedules. For each instance, every algorithm is independently run for 5 replications. The average value of these five replication is as the resource availability cost obtained by the algorithm. RPD (relative percentage deviation) is adopted to evaluate the performance which is defined as

$$RPD = \frac{f(H) - f^*}{f^*} \times 100 \tag{13}$$

where $f(H)$ represents the resource availability cost obtained by the corresponding algorithm H and f^* is the best resource availability cost obtained by all the compared algorithms in this paper.

5.3. Calibration of the algorithms

For our experiments, a number of variants of the proposed algorithm (MSIS) are composed with different combinations of the operators and parameters of previous sections. The first variant does not include the path-relinking phase, local search phase or restart phase and is denoted as MSIS_0; the second variant uses the restart strategy and is denoted as MSIS_1; the third variant uses the local search and is denoted as MSIS_2. The remaining variants are represented in Table 3, in which "1" stands for include and "0" stands for exclude:

MSIS and its variants are calibrated taking into account the different choices for the operators and values for the parameters. Each algorithm is calibrated separately and we make extensive use of the DOE approach. A full factorial design is employed on five parameters as follows.

- Initial start points generation strategy (ISPG): four variants (GCRRR, RPWR, LFTR, RAN).
- Number of start points generated (N_{pop}): five levels (6, 7, 8, 9, 10).
- Size of the elite pool (N_{elite}): five levels (3, 4, 5, 6, 7).
- Maximum number of neighbourhood (K): seven levels (8, 14, 20, 26, 32, 38, 44).
- Maximum attempts to find a better solution in the neighbourhood of a solution (λ): four levels (1, 2, 3, 4)

For the MSIS_0 and MSIS_1, there are two factors for calibration (ISPG and N_{pop}), thus a total of $4 \times 5 = 20$ different combinations are tested. For the MSIS_2 and MSIS_3, there are four factors for calibration (ISPG, N_{pop}, K and λ), so $4 \times 5 \times 7 \times 4 = 560$ different combinations are tested. For the MSIS_4 and MSIS_5 calibration, there are three factors (ISPG, N_{pop}, N_{elite}), so $4 \times 5 \times 5 = 100$ combinations are tested. In the last calibration for the MSIS_6 and MSIS, all factors are included so there are a total of $4 \times 5 \times 5 \times 7 \times 4 = 2800$ different combinations.

The experiment is executed under PSPLib instances with 5000 generated schedules as the termination criterion. The results are analyzed by means of a multi-factor analysis of variance (ANOVA). Three main hypotheses of ANOVA are checked: normality, homoscedasticity and independence of the residuals [18]. Residuals from the results satisfy all three hypotheses. Table 4 shows a summary with the different parameter values for the algorithms MSIS and its variants after the calibration experiments (details about the statistical analysis are not shown due to space restrictions).

In addition, δ is the only parameter of the restart strategy which determines the maximum continuous iterations without updating the elite pool. It exerts an indirect impact on the MSIS algorithm. The parameter calibration would be very time-consuming if we tested all combinations of δ with all the other parameters. Instead, we carry out a simple one factor at a time analysis to calibrate δ. The other parameters are fixed to the values determined previously. The means plot is shown in Figure 10. Figure 10 shows that different levels of δ are not significantly different. This suggests that the MSIS and its variants are rather robust with respect to δ. Therefore, we set δ as 5 in the following experiments.

5.4. Performance comparison on PSPLib instances

To analyze the performance of the algorithms, a considerable amount of experiments have been carried out. The statistical significance of the observed differences in solution quality and CPU-time are validated by the ANOVA technique. First of all, a single-factor ANOVA is carried out. This analysis has a single controlled factor which is the type of algorithm with 11 levels (the 8 tested MSIS variants and 3 competing methods from the literature). The response variables are given by the RPD (the aspect of effectiveness) and elapsed CPU-time (the aspect of efficiency) of every instance. The means plots along with Tukey confidence intervals (at the 95% confidence level) are given in Figure 11 and Figure 12 for both response variables respectively. As can be seen, under the same termination criterion (the number of generated solutions), MSIS and all its variants (MSIS_0 $\sim$ MSIS_6)

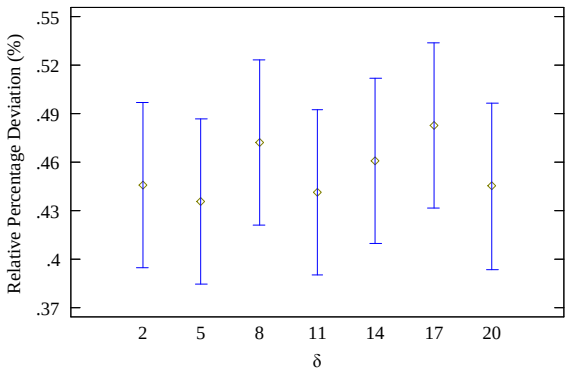


Figure 10: Means plot of RPD 95% confidence level Tukey HSD intervals for various settings of the parameter δ in MSIS on PSPLib with 5000 generated solutions as the termination criterion.

perform better statistically than GA, PR and SS not only in terms of effectiveness, but also efficiency. The differences are statistically significant. SS is statistically better than GA and PR. GA and PR are statistically equal. However, it can be seen from Figure 11 that there are many overlapping confidence intervals among the MSIS and its variants. This may be due to the fact that the differences between MSIS and its variants and the other algorithms are too large to make the small difference among MSIS and its variants hard to observed. Therefore, for a better comparison, a more in-depth analysis on MSIS and its variants is provided later.

A multi-factor ANOVA is performed to analyze the interaction between the algorithm and other controlled factors. Figure 13 shows the means plot of the interaction between the type of algorithm and n factors on effectiveness. In this plot it can be seen that for all algorithms, increasing the value of n results in worse performance. The differences are significant for GA and PR in all cases but small for SS when $n = 30, 60, 90$. It turns out to be significant for SS when $n = 120$. However in all cases, the differences are very small for MSIS_0, MSIS_1, MSIS_4 and MSIS_5 and not even statistically significant for MSIS_2, MSIS_3, MSIS_6 and MSIS. It can also be observed that MSIS and its variants are statistically better than GA and PR when $n = 60, 90, 120$, and statistically better than SS when $n = 120$. Furthermore, SS is statistically better than GA and PR when $n = 60, 90, 120$. GA and PR are statistically equal.

Figure 14 shows the means plot of interaction between the type of algorithm and the termination criterion. As can be seen, the increase in the number of generated solution results in better performance only for SS but has little effect on the other algorithms. This means all algorithms demonstrate a stable performance in relation to effectiveness except SS. Compared to GA and PR, the MSIS and its variants perform better statistically with significant differences in all cases. Compared to SS, the MSIS and its variants perform better statistically with significant differences for termination criteria 1000 and 5000, but with a small difference for termination criterion 10000.

In order to closely study the difference in performance among the proposed MSIS and its variants, a zoomed analysis is provided and the mean plots are shown in Figure 15

and Figure 16. Figure 15 shows that MSIS is statistically better than all its variants except MSIS_6. There is no statistically significant difference between MSIS and MSIS_6. Figure 16 reveals that MSIS needs less CPU-time than MSIS_6 to obtain a given number of solutions. Therefore, MSIS is the best performing method among all the compared algorithms. It can also be found that: i) a variant with a restart strategy performs a little better than without it (e.g. MSIS_1 is statistically better than MSIS_0 with significant difference); ii) a variant with a local search performs better than without it (e.g. MSIS_3 is statistically better than MSIS_1 with significant difference); iii) a variant with path-relinking performs better than without it (e.g. MSIS_4 is statistically better than MSIS_0 with significant difference).

To verify the effectiveness of heuristics BPEP, TSRAP and the two stages of the TSRAP, the proposed MSIS algorithm is transformed into another four algorithms: i) MSIS_BPEP: the MSIS variant in which the TSRAP (used in the LS) is replaced by the BEPE; ii) MSIS_TSRAP: the MSIS variant in which the BEPE (used in the ISPG, PR and NSPG) is replaced by the TSRAP; iii) MSIS_woR: the MSIS without stage RRAC; and iv) MSIS_woF: the MSIS without stage FDRAC. They are compared with the proposed MSIS and the best of literature SS [25]. Figure 17 shows that, MSIS and all its variants are statistically better than SS. From the zoomed mean plots shown in Figure 18 it can been seen that, MSIS_TSRAP is statistically better than MSIS_BPEP, which means TSRAP is an effective mechanism for the algorithm. But MSIS_TSRAP is statistically worse than MSIS. The reason for this may be that, BPEP is a simple resource evaluation method which not only reduce the resource usage cost in a fast way, but also construct more schedules to diversify the population. However, TSRAP is a complex resource evaluation method whose two phases are "deep digging" methods and need to spend plenty of time to search for near-optimal solutions. If TSRAP is used in all the sub-procedures, it may incur fast-convergence or premature which has a negative effect on the final result. It can also be seen that, both two stages of TSRAP are of great significance to the entire algorithm.

The comparison of the average results are also made between MSIS and the best-existing algorithms GA, PR and SS. It confirms our previous observation that MSIS outperforms the other algorithms in instances with a different number of activities and different termination criteria in terms of both effectiveness and efficiency. However, no details are given here due to space considerations. Instead, we have put together the results as on-line materials.

5.5. Performance comparison on RanGen instances

The ANOVA-based statistical analysis results on the RanGen instances are displayed in Figure 19, Figure 20, Figure 23 and Figure 24. The results are similar to those of PSPLib instances.

Of particular interest is the interaction between the type of algorithm and m (the number of types of resources) and θ (the deadline) factors. It can be seen from Figure 21 that GA and PR are statistically equal. They are statistically better than SS when $m = 8$, but worse than SS when $m = 4$ and $m = 6$. However as m increases, there are no statistically significant differences for MSIS and its variants. MSIS and its variants are statistically better than GA and PR, which is even true for them with $m = 4$ when compared to GA and PR with $m = 8$.

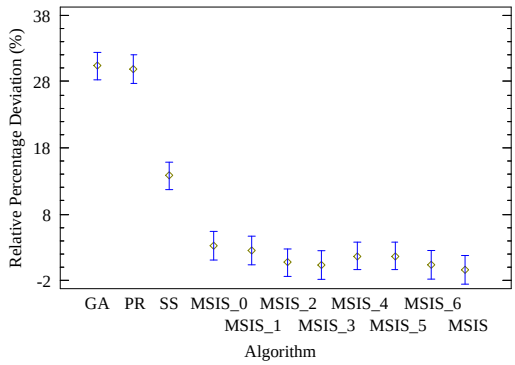


Figure 11: Means plot of RPD 95% confidence level Tukey HSD intervals for all the algorithms tested on the PSPLib instances.

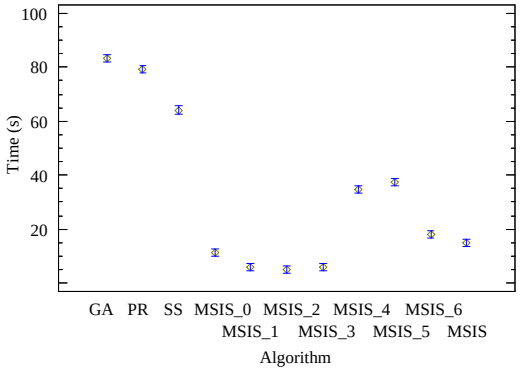


Figure 12: Means plot of CPU-time 95% confidence level Tukey HSD intervals for all the algorithms tested on the PSPLib instances.

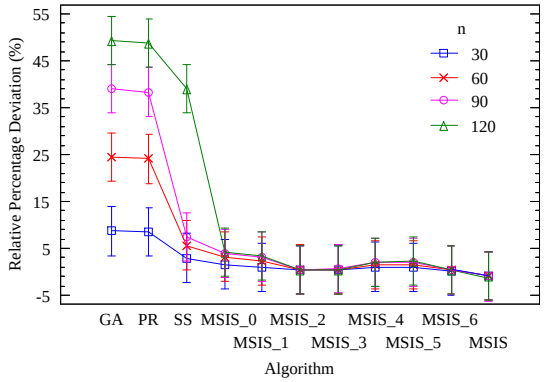


Figure 13: Interaction plot between the type of algorithm and the number of activities with 95% Tukey HSD intervals on the PSPLib instances.

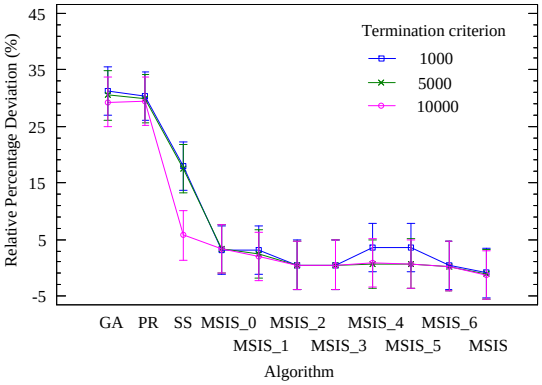


Figure 14: Interaction plot between the type of algorithm and the different values of generated solutions for the termination criterion with 95% Tukey HSD intervals on the PSPLib instances.

Figure 22 reveals that increasing the value of θ results in worse performance for GA and PR with significant differences but it results in better performance for SS, although for $\theta = 1.4$ and $\theta = 1.5$ the differences are very small. However, there is no statistically significant difference for MSIS and its variants. From the plot it is also clear that MSIS and its variants are statistically better than the other algorithms in all cases and by a significant margin.

Similar to those of PSPLib instances, Figure 25 and Figure 26 also show the importance of the proposed resource decision methods on RanGen instances.

6. Conclusions and future research

In this paper, a multi-start iterative search heuristic named MSIS is proposed for the project scheduling problem with resource availability cost (RACP). Instead of transforming the RACP into multiple SMRCPSPs as is commonly done in the literature, we divide the

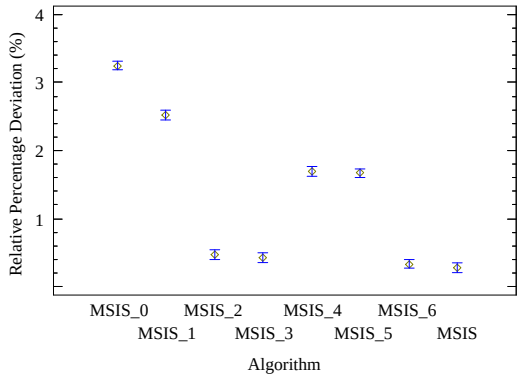


Figure 15: Means plot of RPD 95% confidence level Tukey HSD intervals for the MSIS and its variants on the PSPLib instances.

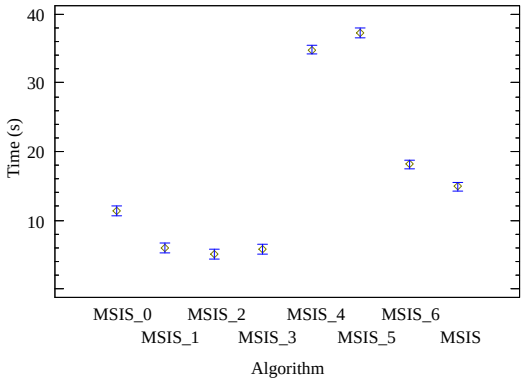


Figure 16: Means plot of CPU-time 95% confidence level Tukey HSD intervals for the MSIS and its variants on the PSPLib instances.

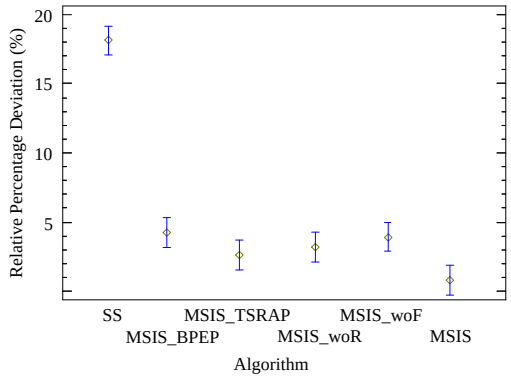


Figure 17: Means plot of RPD 95% confidence level Tukey HSD intervals for SS, MSIS_BPEP, MSIS_TSRAP, MSIS_woR, MSIS_woF and MSIS algorithms tested on the PSPLib instances.

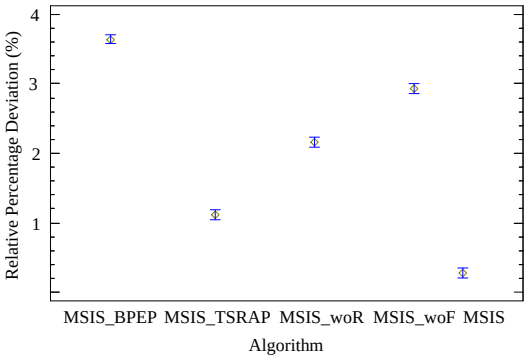


Figure 18: Means plot of RPD 95% confidence level Tukey HSD intervals for the MSIS_BPEP, MSIS_TSRAP, MSIS_woR, MSIS_woF and MSIS on the PSPLib instances.

problem into two sub-problems: the sequencing problem and the resource decision problem, and these two sub-problems are sequentially addressed. A feasible neighbourhood and path-relinking process are constructed to solve the sequencing problem efficiently. Two heuristics, BPEP and TSRAP, are developed for the resource decision problem. BPEP makes sure that the activities are as distributed as possible in the project period to reduce the usage of resources. In TSRAP, both the consistent and fluctuant decline of resource availability cost are considered.

The proposed algorithm has been calibrated by means of a design of experiments approach that involves the evaluation of many different alternative. After the calibration, the best tested combination of operators and parameters for each proposed alternative is obtained.

An extensive comparison of the proposed algorithm against the best existing approaches

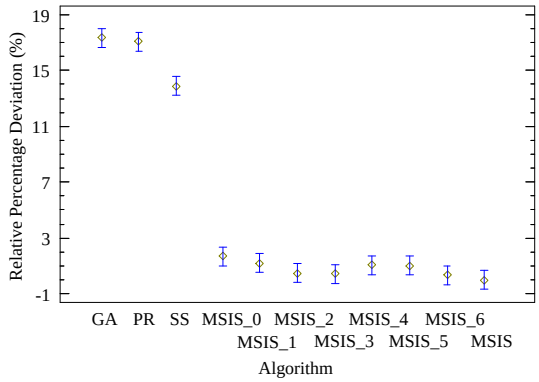


Figure 19: Means plot of RPD 95% confidence level Tukey HSD intervals for the algorithms tested on the RanGen instances.

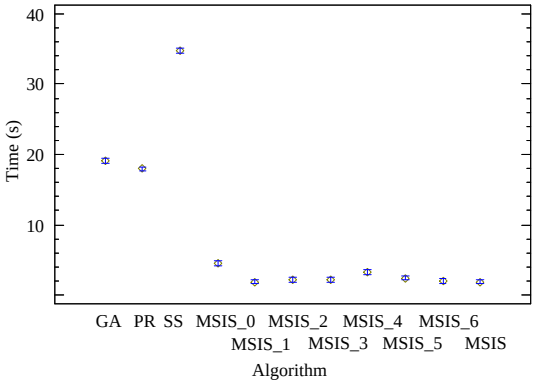


Figure 20: Means plot of CPU-time 95% confidence level Tukey HSD intervals for the algorithms tested on the RanGen instances.

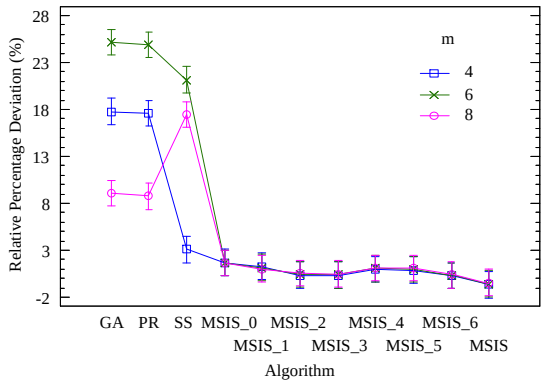


Figure 21: Interaction plot between the type of algorithm and the number of type of resources with 95% Tukey HSD intervals on the RanGen instances.

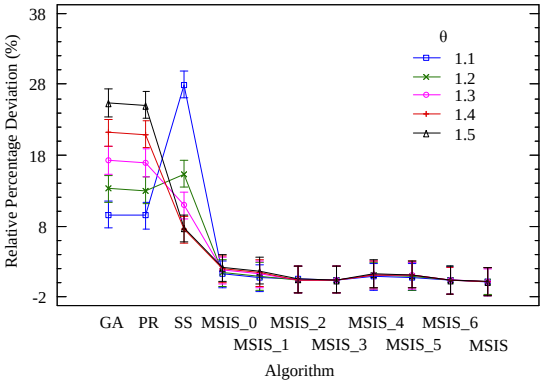


Figure 22: Interaction plot between the type of algorithm and the deadline factor with 95% Tukey HSD intervals on the RanGen instances.

is carried out. A comprehensive calibration, the study of interactions and a careful algorithm engineering process has resulted in a competitive experiment. The experimental and statistical results show that the proposed algorithm is superior to the others in terms of both effectiveness and efficiency by a considerable margin. Of course, this is, at least in part, due to the extensive experimentation carried out, which evidences the need of proper and sound statistical experimentation in scientific research.

Future research avenues involve the consideration of the improvement of the proposed framework. For example, historical information can be introduced as a memory mechanism for guiding the search and some of the proposed operators may be modified to reduce the time consumption. It also seems worthwhile applying the proposed algorithm to other resource constrained project scheduling problems in practical applications.

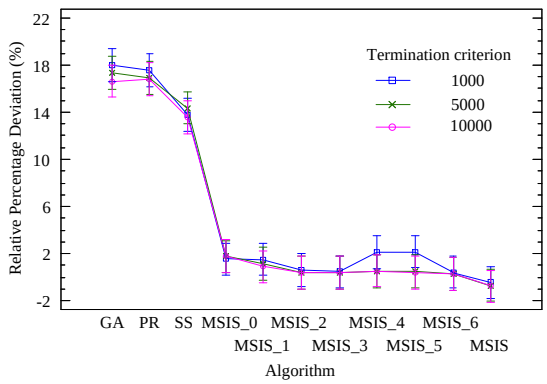


Figure 23: Interaction plot between the type of algorithm and the different values of generated solutions as termination criterion with 95% Tukey HSD intervals on the RanGen instances.

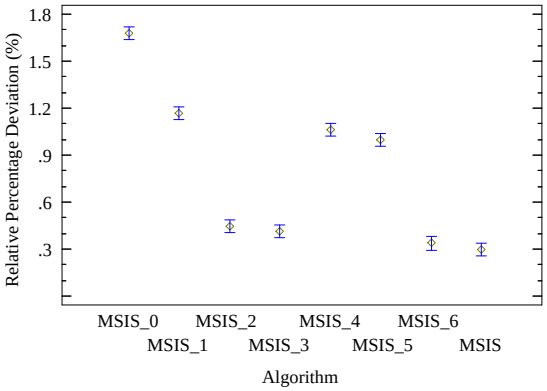


Figure 24: Means plot of RPD 95% confidence level Tukey HSD intervals for the MSIS and its variants on the RanGen instances.

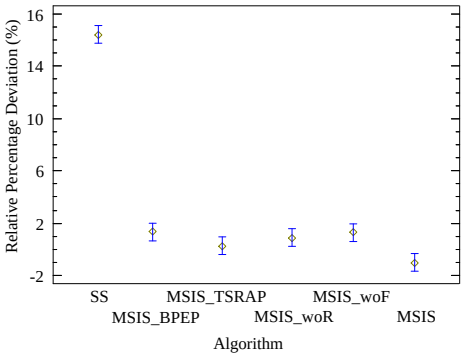


Figure 25: Means plot of RPD 95% confidence level Tukey HSD intervals for SS, MSIS_BPEP, MSIS_TSRAP, MSIS_woR, MSIS_woF and MSIS algorithms tested on the RanGen instances.

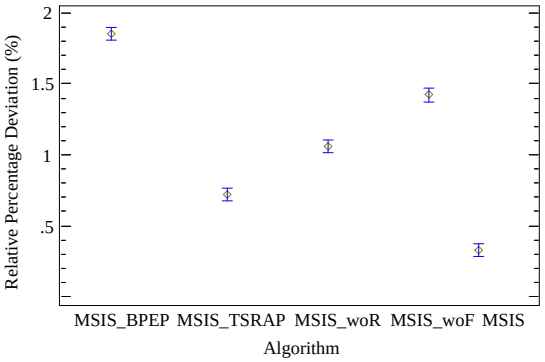


Figure 26: Means plot of RPD 95% confidence level Tukey HSD intervals for the MSIS_BPEP, MSIS_TSRAP, MSIS_woR, MSIS_woF and MSIS on the RanGen instances.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (Nos. 61572127, 61272377), the Key Research & Development program in Jiangsu Province (No. BE2015728) and the Collaborative Innovation Center of Wireless Communications Technology. Rubén Ruiz is partially supported by the Spanish Ministry of Economy and Competitiveness, under the project "SCHEYARD - Optimization of Scheduling Problems in Container Yards" with reference DPI2015-65895-R.

References

[1] Hédi Chtourou and Mohamed Haouari. A two-stage-priority-rule-based algorithm for robust resource-constrained project scheduling. *Computers & Industrial Engineering*, 55(1):183–194, 2008.

[2] Erik Demeulemeester. Minimizing resource availability costs in time-limited project networks. *Management Science*, 41(10):1590–1598, 1995.
[3] Erik Demeulemeester. *Project scheduling: a research handbook*, volume 102. Springer, 2002.
[4] Erik Demeulemeester and Willy Herroelen. A branch-and-bound procedure for the multiple resource-constrained project scheduling problem. *Management Science*, 38(12):1803–1818, 1992.
[5] Erik Demeulemeester, Mario Vanhoucke, and Willy Herroelen. Rangen: A random network generator for activity-on-the-node networks. *Journal of Scheduling*, 6(1):17–38, 2003.
[6] Erik Leuven Demeulemeester and Willy S Herroelen. *Project scheduling: a research handbook*, volume 49. Springer Science & Business Media, 2006.
[7] Andreas Drexl and Alf Kimms. Optimization guided lower and upper bounds for the resource investment problem. *Journal of the Operational Research Society*, 52(3):340–351, 2001.
[8] Fred Glover, Manuel Laguna, and Rafael Martí. Fundamentals of scatter search and path relinking. *Control and cybernetics*, 39(3):653–684, 2000.
[9] Sönke Hartmann and Rainer Kolisch. Experimental evaluation of state-of-the-art heuristics for the resource-constrained project scheduling problem. *European Journal of Operational Research*, 127(2):394–407, 2000.
[10] Willy Herroelen and Roel Leus. Project scheduling under uncertainty: Survey and research potentials. *European journal of operational research*, 165(2):289–306, 2005.
[11] James E Kelley. The critical-path method: Resources planning and scheduling. *Industrial scheduling*, 13:347–365, 1963.
[12] Rainer Kolisch and Sönke Hartmann. Experimental investigation of heuristics for resource-constrained project scheduling: An update. *European Journal of Operational Research*, 174(1):23–37, 2006.
[13] Rainer Kolisch and Arno Sprecher. PSPLIB-a project scheduling problem library: OR software-ORSEP operations research software exchange program. *European Journal of Operational Research*, 96(1):205–216, 1997.
[14] Haitao Li and Norman K Womer. Solving stochastic resource-constrained project scheduling problems by closed-loop approximate dynamic programming. *European Journal of Operational Research*, 246(1):20–33, 2015.
[15] Weimin Ma, Yangyang Che, Hu Huang, and Hua Ke. Resource-constrained project scheduling problem with uncertain durations and renewable resources. *International Journal of Machine Learning and Cybernetics*, pages 1–9, 2015.
[16] Nenad Mladenović and Pierre Hansen. Variable neighborhood search. *Computers and Operations Research*, 24(11):1097–1100, 1997.
[17] Rolf H Möhring. Minimizing costs of resource requirements in project networks subject to a fixed completion time. *Operations Research*, 32(1):89–120, 1984.
[18] Douglas C Montgomery. *Design and analysis of experiments (8th edition)*. John Wiley & Sons, 2012.
[19] Mohammad Ranjbar, Fereydoon Kianfar, and Shahram Shadrokh. Solving the resource availability cost problem in project scheduling by path relinking and genetic algorithm. *Applied Mathematics and Computation*, 196(2):879–888, 2008.
[20] Sávio B Rodrigues and Denise S Yamashita. An exact algorithm for minimizing resource availability costs in project scheduling. *European Journal of Operational Research*, 206(3):562–568, 2010.
[21] Shahram Shadrokh and Fereydoon Kianfar. A genetic algorithm for resource investment project scheduling problem, tardiness permitted with penalty. *European Journal of Operational Research*, 181(1):86–101, 2007.
[22] Pilar Tormos and Antonio Lova. An efficient multi-pass heuristic for project scheduling with constrained resources. *International Journal of Production Research*, 41(5):1071–1086, 2003.
[23] Vincent Van Peteghem and Mario Vanhoucke. An artificial immune system algorithm for the resource availability cost problem. *Flexible Services and Manufacturing Journal*, 25(1-2):122–144, 2013.
[24] Mario Vanhoucke and José Coelho. An approach using sat solvers for the rcpsp with logical constraints. *European Journal of Operational Research*, 249(2):577–591, 2016.
[25] Denise Sato Yamashita, Vinícius Amaral Armentano, and Manuel Laguna. Scatter search for project

scheduling with resource availability cost. *European Journal of Operational Research*, 169(2):623–637, 2006.

[26] Kum-Khiong Yang. A comparison of dispatching rules for executing a resource-constrained project with estimated activity durations. *Omega*, 26(6):729–738, 1998.

Algorithm 8: Backward Peak Elimination Procedure, BPEP (π)

begin
  Call SGS(π) to generate a schedule;
  for *each activity* $\mathbb{T}_i \in \mathbb{T}$ **do**
    $lst_{\mathbb{T}_i} \leftarrow$ Calculate the latest start time of activity $\mathbb{T}_i$ by the critical-path based forward and backward pass method [3] ;
  for $i = n$ **to** 1 **do**
    $min \leftarrow 0$;
    for $w = st_{\mathbb{T}_i}$ **to** $lst_{\mathbb{T}_i}$ **do**
      Start time of activity $\mathbb{T}_i$ is postponed from $st_{\mathbb{T}_i}$ to w and $a_k(t)$ is updated as:
      for $k = 1$ **to** m **do**
        for $t = st_{\mathbb{T}_i}$ **to** $st_{\mathbb{T}_i} + d_{\mathbb{T}_i}$ **do**
          $a_k(t) \leftarrow a_k(t) - r_k^{\mathbb{T}_i}$;
        for $t = w$ **to** $w + d_{\mathbb{T}_i}$ **do**
          $a_k(t) \leftarrow a_k(t) + r_k^{\mathbb{T}_i}$;
      $dre \leftarrow \mathcal{C}(A) - \mathcal{C}(\mathbb{R})$;
      if $dre \leq min$ **then**
        $min \leftarrow dre$, $w' \leftarrow w$;
      `/* Retrieve the original` $a_k(t)$ `for the next loop */`
      for $k = 1$ **to** m **do**
        for $t = w$ **to** $w + d_{\mathbb{T}_i}$ **do**
          $a_k(t) \leftarrow a_k(t) - r_k^{\mathbb{T}_i}$;
        for $t = st_{\mathbb{T}_i}$ **to** $st_{\mathbb{T}_i} + d_{\mathbb{T}_i}$ **do**
          $a_k(t) \leftarrow a_k(t) + r_k^{\mathbb{T}_i}$;
    Start time of activity $\mathbb{T}_i$ is postponed from $st_{\mathbb{T}_i}$ to w' and $a_k(t)$ is updated as:
    for $k = 1$ **to** m **do**
      for $t = st_{\mathbb{T}_i}$ **to** $st_{\mathbb{T}_i} + d_{\mathbb{T}_i}$ **do**
        $a_k(t) \leftarrow a_k(t) - r_k^{\mathbb{T}_i}$;
      for $t = w'$ **to** $w' + d_{\mathbb{T}_i}$ **do**
        $a_k(t) \leftarrow a_k(t) + r_k^{\mathbb{T}_i}$;
    $st_{\mathbb{T}_i} \leftarrow w'$;
    for *each activity* $j \in pred_{\mathbb{T}_i}$ **do**
      if $lst_j > (st_{\mathbb{T}_i} - d_j)$ **then**
        $lst_j \leftarrow (st_{\mathbb{T}_i} - d_j)$;
    $\mathbb{R} \leftarrow A$;
  Sort $\mathbb{T}$ in the increasing order of $st_{\mathbb{T}_i}$, where $\mathbb{T}_i \in \mathbb{T}$;

Algorithm 9: Fluctuant Decline of Resource Availability Cost, FDRAC (π, k)

begin
  $\pi' \leftarrow \pi$, $\mathbb{R}' \leftarrow \underline{A}$, $\mathbb{R}'_k \leftarrow \mathbb{R}_k$;
  Initialize the resource usage variable $a_k(t) \leftarrow 0$ for all $k \in M$ and $t \in [1, ..., H]$;
  for *each activity* $\mathbb{T}'_i \in \mathbb{T}'$ **do**
    $est_{\mathbb{T}'_i} \leftarrow 0$;
    $lst_{\mathbb{T}'_i} \leftarrow$ Calculate the latest start time of activity $\mathbb{T}'_i$ by the critical-path based forward and backward pass method;
  for $i = 1$ **to** n **do**
    $min \leftarrow +\infty$;
    for $w = est_{\mathbb{T}'_i}$ **to** $lst_{\mathbb{T}'_i}$ **do**
      Add $\mathbb{T}'_i$ to the current schedule at the start time w ;
      for $k = 1$ **to** m **do**
        for $t = w$ **to** $w + d_{\mathbb{T}'_i}$ **do**
          $a_k(t) \leftarrow a_k(t) + r_k^{\mathbb{T}'_i}$;
      if $A_k \leq \mathbb{R}_k$ *and* $\mathcal{C}(A) < \mathcal{C}(\mathbb{R})$ **then**
        $incre \leftarrow \mathcal{C}(A) - \mathcal{C}(\mathbb{R}')$;
        if $incre < min$ **then**
          $min \leftarrow incre$, $w' \leftarrow w$;
      /* Retrieve the original $a_k(t)$ for the next loop */
      for $k = 1$ **to** m **do**
        for $t = w$ **to** $w + d_{\mathbb{T}'_i}$ **do**
          $a_k(t) \leftarrow a_k(t) - r_k^{\mathbb{T}'_i}$;
    if $min < +\infty$ **then**
      Add $\mathbb{T}'_i$ to the current schedule at the start time w' ;
      for $k = 1$ **to** m **do**
        for $t = w'$ **to** $w' + d_{\mathbb{T}'_i}$ **do**
          $a_k(t) \leftarrow a_k(t) + r_k^{\mathbb{T}'_i}$;
      $st_{\mathbb{T}'_i} \leftarrow w'$;
      for *each activity* $j \in succ_{\mathbb{T}'_i}$ **do**
        if $est_j < (st_{\mathbb{T}'_i} + d_{\mathbb{T}'_i})$ **then**
          $est_j \leftarrow (st_{\mathbb{T}'_i} + d_{\mathbb{T}'_i})$;
      $\mathbb{R}' \leftarrow A$, $\mathbb{R}'_k \leftarrow \mathbb{R}_k$;
      if $min > 0$ **then**
        for $j = 1$ **to** i **do**
          $est_{\mathbb{T}'_j} \leftarrow 0$;
        for $j = 1$ **to** i **do**
          Find the earliest feasible start time w for activity $\mathbb{T}'_j$ under the constraint of $\mathbb{R}'$ within $[est_{\mathbb{T}'_j}, st_{\mathbb{T}'_j}]$;
          Start time of $\mathbb{T}'_j$ is postponed from $st_{\mathbb{T}'_j}$ to w;
          for $k = 1$ **to** m **do**
            for $t = st_{\mathbb{T}'_j}$ **to** $st_{\mathbb{T}'_j} + d_{\mathbb{T}'_j}$ **do**
              $a_k(t) \leftarrow a_k(t) - r_k^{\mathbb{T}'_j}$;
            for $t = w$ **to** $w + d_{\mathbb{T}'_j}$ **do**
              $a_k(t) \leftarrow a_k(t) + r_k^{\mathbb{T}'_j}$;
          $st_{\mathbb{T}'_j} \leftarrow w$;
          for *each activity* $k \in succ_{\mathbb{T}'_j}$ **do**
            if $est_k < (st_{\mathbb{T}'_i} + d_{\mathbb{T}'_i})$ **then**
              $est_k \leftarrow (st_{\mathbb{T}'_i} + d_{\mathbb{T}'_i})$;
    else
      $\pi' \leftarrow \pi$, **return** π';
  $\mathbb{R}' \leftarrow A$, sort $\mathbb{T}'$ in the increasing order of $st_{\mathbb{T}'_i}$, $\mathbb{T}'_i \in \mathbb{T}'$, **return** π'.

Algorithm 10: Two Stage Resource Adjustment Procedure, TSRAP (π)

begin
  Sort resources in $\mathbb{R}$ in decreasing order of their costs;
  repeat
    /* ---The RRAC procedure--- */
    $flag \leftarrow$ **false**, $\pi' \leftarrow \pi$;
    for $k = 1$ **to** m **do**
      if $\mathbb{R}'_k > \underline{A}_k$ **then**
        $\mathbb{R}'_k \leftarrow \mathbb{R}'_k - 1$, $feasible \leftarrow \mathrm{FBP}(\pi')$;
        if *not feasible* **then**
          $\mathbb{R}_k \leftarrow \mathbb{R}'_k + 1$;
    Call BPEP(π') to allocate resources;
    if $\mathcal{C}(\mathbb{R}') < \mathcal{C}(\mathbb{R})$ **then**
      $\pi \leftarrow \pi'$, $flag \leftarrow$ **true**;
    /* ---The FDRAC procedure--- */
    $min \leftarrow 0$;
    for $k = 1$ **to** m **do**
      if $\mathbb{R}_k > \underline{A}_k$ **then**
        $\mathbb{R}_k \leftarrow \mathbb{R}_k - 1$, $\pi' \leftarrow \mathrm{FDRAC}(\pi, \mathrm{k})$, $dre \leftarrow \mathcal{C}(\mathbb{R}') - \mathcal{C}(\mathbb{R})$;
        if $dre < min$ **then**
          $min \leftarrow dre$, $\pi^* \leftarrow \pi'$;
        $\mathbb{R}_k \leftarrow \mathbb{R}_k + 1$;
    if $min < 0$ **then**
      $\pi \leftarrow \pi^*$, $flag \leftarrow$ **true**;
  until *not* $flag$;

Table 3: Compositions of the proposed MSIS and its variants.

Path-relinking	Local search	Restart	Algorithm
0	0	0	MSIS_0
0	0	1	MSIS_1
0	1	0	MSIS_2
0	1	1	MSIS_3
1	0	0	MSIS_4
1	0	1	MSIS_5
1	1	0	MSIS_6
1	1	1	MSIS

Table 4: Operators and parameter values used for MSIS and its variants after calibration.

Factor	MSIS_0	MSIS_1	MSIS_2	MSIS_3	MSIS_4	MSIS_5	MSIS_6	MSIS
ISPG	RPWR	GCRRR	GCRRR	GCRRR	GCRRR	GCRRR	GCRRR	GCRRR
N_{pop}	10	6	9	10	9	8	10	10
N_{elite}	-	-	-	-	6	6	5	5
K	-	-	32	32	-	-	32	26
λ	-	-	2	1	-	-	1	2