Automation in Construction 111 (2020) 103052

Contents lists available at ScienceDirect

Automation in Construction

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

Memetic algorithm for solving resource constrained project scheduling problems

Humyun Fuad Rahman*, Ripon K. Chakrabortty, Michael J. Ryan

Capability Systems Centre, School of Engineering and IT, University of New South Wales, Canberra, ACT 2610, Australia

ARTICLE INFO

Keywords:
Project scheduling
Resource constrained
Makespan
Genetic algorithm
Memetic algorithm

ABSTRACT

The resource constrained project scheduling problem (RCPSP) has a wide variety of practical applications in construction, manufacturing, project planning, and other areas. Since the 1960s many optimization algorithms have been proposed to solve this NP-hard problem, and their performances are evaluated in well-known test problems with different complexities. Although it is desirable to find an algorithm which can provide promising solutions with reasonable computational efforts for any problem under consideration, no single algorithm can meet that condition. To deal with this challenge, we present a genetic algorithm based memetic algorithm (MA) for solving RCPSP. The algorithm is initiated by a critical path-based heuristic and a variant of the Nawaz, Enscore, and Ham (NEH) heuristic. The algorithm involves a similar block order crossover and a variable insertion based local search. An automatic restart scheme is also presented which assists the algorithm to escape from local optima. In addition, a design-of-experiment (DOE) method is used to determine the set of suitable parameters for the proposed MA. Numerical results, statistical analysis and comparisons with state-of-the-art algorithms demonstrate the effectiveness of the proposed approach.

1. Introduction

The resource constrained project scheduling problem (RCPSP) is one of the important scheduling problems that attracts the attention of researchers and practitioners due to its computational complexity (known to be NP-Hard) [1,2] and several real-life applications such as in construction industries, automobile manufacturing, steel production, assembly production scheduling, and job-shop environments [3]. The RCPSP can be defined as the need to minimize the duration of a project (makespan) while satisfying relevant constraints. Normally, a project consists of $(n + 2)$ activities, i.e. $\{1, \ldots j \ldots, n + 2\}$, where the first and the last activity of the project are dummy activities. Each activity j requires either some or all k renewable resources, i.e. $\{r_{1j}, r_{2j} \ldots, r_{kj}\}$ at the same or different level at time s, and for the duration d_j. R_k is the maximum availability of the kth resource. Note that, duration and resource utilization of a dummy activity are both 0. Finally, each activity j needs to follow its precedence constraint, i.e. an activity cannot be started unless all its predecessors, $pred(i)$ are completed.

Generally, a mathematical model of the single mode RCPSPs can be expressed as follows.

Decision variable:

$$a_{jt} = \begin{cases} 1, \text{if jth activity starts to process at time t} \\ 0, \text{otherwise} \end{cases}$$

$$\min\ C_{max} = \sum_{t=0}^{T} a_{(n+2)t}.\ (t + d_{n+2}) \tag{1}$$

subject to:

$$\sum_{t=0}^{T} a_{jt} = 1\ \forall\ i \in n \tag{2}$$

$$\sum_{t=0}^{T} t.\ a_{it} \geq \sum_{t=0}^{T} (t + d_j).\ a_{jt}\ \forall\ i \in n \text{ and } \forall\ j \in pred\ (i) \tag{3}$$

$$\sum_{j=1}^{n+2} \sum_{t-d_j+1}^{t} r_{kj}.\ a_{jt} \leq R_k\ \forall\ k \in K \text{ and } \forall\ t \in T \tag{4}$$

$$a_{jt} \in \{0, 1\} \tag{5}$$

The objective function is the minimization of makespan, C_{max} (Eq. (1)). Eq. (2) ensures that an activity can be executed only once. Eq. (3) ensures that each activity j cannot be started unless all its predecessors have been completed. Eq. (4) ensures that an activity can be started when its required renewable resources (such as workforce, machines, tools or equipment) are available.

Over the years, exact techniques such as branch and bound [4–6], branch and cut [7], and the event based approach [8] have been proposed for the optimal solution of RCPSPs. Due to computational

* Corresponding author.
E-mail address: humyun.fuad@adfa.edu.au (H.F. Rahman).

https://doi.org/10.1016/j.autcon.2019.103052
Received 4 March 2019; Received in revised form 24 November 2019; Accepted 7 December 2019
Available online 24 December 2019

complexity and NP-hard problems, these techniques can only solve small problems. Recently, Coelho and Vanhoucke [9] proposed a branch and bound method, which dynamically selects the lower bounds for solving small and large RCPSPs problem instances. However, this approach is computationally expensive. Thus, researchers have focused on developing heuristics and meta-heuristic techniques, such as genetic algorithm (GA) [10], differential evolution (DE), ant colony algorithm (ACO), and particle swarm optimization (PSO) for finding optimal or near-optimal solutions for RCPSPs within reasonable computational times. A detail review of the heuristic and metaheuristic algorithms is presented in Section 2 which reveals the following gaps in the literature.

- No single approach is suitable for solving all kinds of RCPSPs (from a small number of activity instances to larger numbers of activities) with reasonable solution quality and computational time.
- Even though some of those approaches have shown better performance, their performance is however far from the lower bounds or optimal solutions (if known).

The first point is the consequence of the no-free-lunch theorem [11] and the second point relates to the nature of NP-Hard problems. Hence, neither of the gaps can be closed and therefore, the best exact technique can solve at most 60 activities for RCPSPs, where the instances are not highly constrained by resources [12]. Thus, there is scope to design improved heuristic or meta-heuristic algorithms that may be able to solve RCPSPs effectively. In this study, we address this gap by developing an efficient GA-based memetic algorithm (MA) to handle the entire spectrum of single-mode RCPSPs with makespan minimization as the objective. More specifically, the key contributions of this study can be summarized as follows.

- In order to initialize the proposed algorithm with promising individuals, an initialization method is proposed on the basis of the Nawaz, Enscore, and Ham heuristic (NEH) [13], which is adopted from application to the flow shop scheduling problem. This NEH heuristic also has embedded the well-known critical path-based heuristic, and some efficient priority rules from assembly line balancing problems.
- A modified crossover operator based on similar block order crossover (SBOX) is proposed to keep the block of activities in the same point of the solution and explore more effective solutions.
- To avoid being stuck in local optima, an automated restart mechanism has been implemented. Further to enhance the local search ability of MA, a local search approach based on variable insertion neighborhood search is also proposed.
- Finally, a duplication scheme is presented which minimizes the duplicity of generated schedules and improves diversity.

The efficiency of the proposed approach is evaluated by solving 2040 problem instances available in the standard project scheduling library (PSPLIB) for 30, 60, 90, and 120 activities [14]. Moreover, 1800 instances for RG30, 480 instances for RG300 from http://www.projectmanagement.ugent.be/?q=research/data/RanGen, and some real-world case studies are also solved to verify this proposed approach.

The reminder of this paper is organized as follows. Section 2 provides a background on the existing literature on heuristics and meta-heuristic techniques for RCPSPs. Section 3 gives a detail on the proposed MA. Parameter tuning and a discussion on the experimental results is presented in Section 4. Section 5 describes the importance of the proposed approach for solving construction project scheduling problems, while Section 6 provides some conclusions and highlights future research directions.

2. Related work on solving RCPSPs

Over the last few decades there has been a growing literature in the field of heuristic and meta-heuristics for solving RCPSPs. A review of meta-heuristic algorithms for solving RCPSPs can be found in [12]. Explanation of some selected state-of-the art algorithms is provided below.

2.1. Evolutionary algorithms

Alcaraz and Maroto [15] proposed a GA-based approach for solving RCPSP utilizing three crossover operators (i.e. precedence set, forward-backward and two-point forward-backward) and two mutation operators: (1) exchange each activity with its following one and (2) randomly move the position of each activity to another one. Among these operators, the forward-backward crossover and the second mutation operator outperforms other variants. However, this algorithm is still far away from the optimal or best-known solutions.

A decomposition-based GA (DBGA) was proposed by Debels and Vanhoucke [16] where the algorithm was initiated with random solutions and a serial generation scheme was used to convert the infeasible schedules into a feasible solution. The algorithm decomposes the problem into subproblems and uses a modified version of peak crossover operator [16] for competitively solving RCPSPs. The algorithm shows competitive performance for the PSPLIB problem instances. In the GA proposed by Valls et al. [17], a peak crossover was proposed to combine the good parts of a chromosome and a double justification operator was used as a local search operator. The algorithm shows promising performance to other algorithms reported in that paper, for 60 and 120 activity instances from the PSPLIB benchmark. However, this algorithm is outperformed by more-recent algorithms.

Mendes et al. [18] proposed a GA with random key representation of chromosomes, and the schedules were constructed by a heuristic rule. As the reproduction operator of the GA, they used a uniform and single point crossover and a mutation operator. Although their approach showed competitive solutions, it could not outperform the state-of-the art algorithms. Gonçalves et al. [19] proposed a similar approach which also could not outperform the state-of-the art algorithms and suffered from poor convergence rate.

Zamani [20] proposed a magnet crossover operator-based GA (GA-MBX), which was an enhanced version of two-point crossover operator. In that crossover operation, one contiguous part from the first parent and two contiguous parts from the second parent's genotype are preserved. The performance of the algorithm was enhanced by a local search scheme. Zamani [21] proposed GA and an implicit enumeration search technique (IEST) based algorithm, where the algorithm initiated with IEST and then the set of solutions were improved by GA. Although the algorithm showed promising results, it is far away from the best-known solutions.

Agarwal et al. [22] proposed GA and neural network (NN) based approach for solving RCPSPs where a NN approach is used as a local search. However, performance of the algorithm was inferior to other competitive techniques. Anagnostopoulos and Koulinas [23] proposed a genetic hyper heuristic algorithm where six heuristic were used in the upper level and eight heuristic were used in the lower level. The algorithm was not compared with other algorithms. Using the same algorithmic framework, the same authors proposed another algorithm by replacing GA with a greedy randomized adaptive search procedure [23].

Cheng et al. [24] proposed a fuzzy clustering and a chaotic approach based DE algorithm (FCDE) and solved two real case studies. In that algorithm, a logistic map generated chaotic sequence was then converted to feasible schedules by a serial generator. Although the algorithm showed competitive results, it was tested for small number of

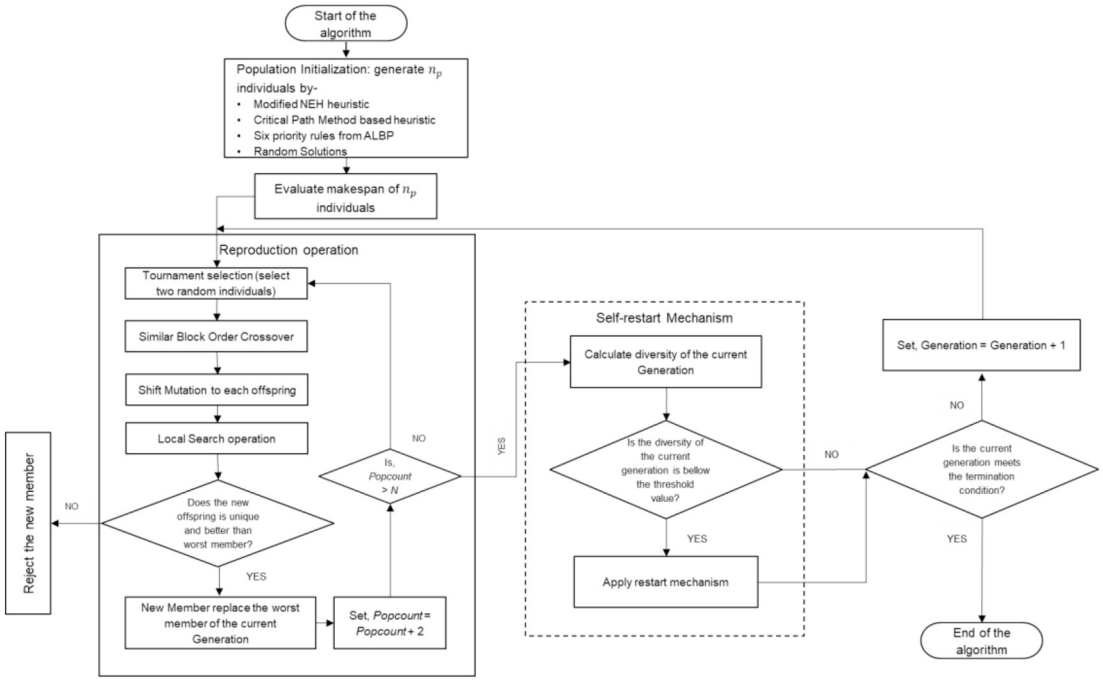


Fig. 1. Flowchart describing the proposed framework of MA.

tested problems and it is therefore difficult to judge its performance against other competitive algorithms. Fang and Wang [25] proposed GA and DE fused shuffled frog-leaping algorithm (SFLA) where a virtual frog was encoded as the extended activity list (EAL) and decoded by a serial schedule generation process. The algorithm was initialized by priority rules and the regret-based sampling method and the population was evolved through a resource-based crossover operator. Diversity of SFLA was maintained by regular shuffling of the virtual frogs and a forward-backward improvement and permutation-based local search. Although the algorithm showed inferior performances for the problem instances with 30 and 60 activities, it showed promising performances for the problem instances with 120 activities.

Recently, Elsayed et al. [26] proposed a two-operator based consolidated evolutionary algorithm in which meta-heuristics are selected self-adaptively.

2.2. Swarm intelligence based algorithms

Zhang et al. [27] proposed two PSO variants, where the first algorithm used priority-based solution representation and the second used serial method with permutation-based representation to convert the particles into feasible solutions. However, recently these algorithms are outperformed by recent PSO-based algorithms [3,28]. Later, Zhang et al. [29] extended this algorithm with parallel method to generate feasible solutions but it is hard to judge their algorithm's competitiveness since they tested their algorithm with small test problems.

Jia and Seo [28] introduced an improved PSO where particles were represented by a rank-priority technique and the solutions were enhanced by a double justification based local search technique. This algorithm showed superior performances than other PSO variants for solving PSPLIB problem instances. However, it is still inferior to other algorithms previously reported in the literature. In Koulinas et al. [30], a PSO based hyper-heuristic was proposed where eight heuristics were used in lower level. Each particle was encoded by random keys and decoded by serial schedule generation scheme. In addition, each particle was improved by a local search. Although the algorithm shows promising results for solving PSPLIB instances, it suffers from low convergence rate.

Ziarati et al. [31] developed three variants of bee algorithms for tackling RCPSPs, namely: (i) bee algorithm (BA), (ii) artificial bee colony (ABC), and (iii) bee swarm optimization (BSO). Each of these algorithms were enhanced by three different local search techniques. Although BSO with FBI approach showed better performance than other two variants, it was inferior to state-of-the-art algorithms. In [32], an ACO, scatter search (SS), and a local search-based hybrid algorithm was proposed to tackle RCPSPs. Firstly, the algorithm was initialized by ACO and then solutions were transferred to SS. The algorithm showed promising performances for 120 activities, but they were not competitive for 30 and 60 activities.

2.3. Other approaches

To tackle RCPSPs, tabu search based approaches were proposed by Klein [33] and Palpant et al. [34]. Bouleimen and Lecocq [35] developed a simulated annealing algorithm for solving the same problem. Wang and Fang [36] proposed an estimation of distribution algorithm where the solution was encoded by EAL and decoded by a serial schedule generation scheme to generate feasible schedules. The algorithm was enhanced by a local search, but it did not provide competitive solutions while solving PSPLIB instances.

3. Proposed GA based MA

As the aim of this study is to provide an efficient methodology to solve RCPSPs with makespan minimization as the objective, a genetic algorithm (GA) based MA has been proposed to solve the problem. The genetic algorithm is an evolutionary algorithm proposed by Holland [37,38], and has been applied for solving complex engineering

optimization problems [37–39]. A genetic algorithm is a convenient choice for RCPSPs given its achievements in all sorts of optimization domains [37,38]. The GA concept considers a population of a set of random solutions, called chromosomes. These chromosomes are evolved by a natural evolution process, namely selection, crossover, and mutation operations to search the best chromosome (i.e. solution) with best fitness. Furthermore, the search process can be enhanced by local search. This framework (i.e. GA with local search process) is known as MA. The search process of MA continues until the stopping criteria is met. The detail framework of the proposed MA is presented in Fig. 1. Following this figure, the detailed structure of the proposed MA is discussed in the following section.

3.1. Solution encoding and population initialization

A chromosome of a GA can be represented as an integer, binary, or real value. For solving RCPSP, solutions can be represented by parallel or serial schedule generation schemes. In this study, a chromosome represents a solution by a serial schedule generation scheme, where the sequence of activities follows a start-to-finish pattern. Traditionally, a GA starts with a set of random population. However, with random initialization, a GA may not converge to quality solutions within a reasonable computational time for solving complex problems [38]. Hence, we propose a non-random initialization in the following way: first two members of the initial population of the proposed MA are generated by two proposed heuristics, namely modified NEH and critical path method (CPM) based heuristic. The following six priority rules are also applied as the following six members of the initial population. Among those rules, maximum rank positional weight and maximum total number of follower tasks have already been implemented for solving RCPSPs [40]. The remaining four rules are adopted from assembly line balancing problems (ALBPs) [41]: maximum activity time, minimum activity time, minimum inverse positional weight, and minimum total number of predecessors activities. The reason to adopt these rules are: (1) these rules are effective in solving ALBPs and (2) these rules are simple and, can be applied for solving RCPSPs without any modification. Finally, the remaining members of the initial population are randomly generated.

3.1.1. Modified NEH heuristic

The NEH heuristic is an effective algorithm for solving the permutation flow shop scheduling problem (PFSP) and its variants [42]. Due to similarity in the attributes of these problems, this heuristic has been adopted to solve RCPSPs. However, unlike PFSPs, RCPSP has precedence and resource constraints, and therefore the original NEH heuristic cannot be employed to solve RCPSPs. Hence, we have proposed a modified NEH heuristic algorithm for solving RCPSPs. The proposed modified NEH heuristic is as follows. In the first stage of original NEH heuristic, all activities (or jobs) are sorted by the longest processing time rule. However, in RCPSPs each activity must occupy single or multiple resources for a certain duration. Hence, in the first stage of the algorithm we sort all the activities based on both processing time and resource utilization. Considering precedence constraints, the generated schedule may be infeasible, which needs to be repaired to make it feasible. Next, a partial schedule is generated by inserting jth activity in all possible positions while considering its precedence relations. The pseudocode of the modified NEH heuristic algorithm is shown in Algorithm 1.

Algorithm 1. Modified NEH heuristic for RCPSPs.

Start of the algorithm
Input: The set of activities (n+2)
Step1. Calculate average resource utilization rate, Rt_j for each activity be using the formula:

$$Rt_j = d_j \times Ar_j \quad \forall j$$

where, average resource utilized by j^{th} activity, $Ar_j = \sum_{k=1}^{k=K}(r_{kj}/R_k) \quad \forall j$

Step2. Sort all the activities based on descending order of Rt_j.
Step3. Repair the infeasible schedule which satisfy the precedence constraints.
Step4. The first and second activities from step 3 are selected and then calculate the makespan, C_{max} of this partial sequence by equation 1. Set j=3.
Step5. *while* $(\boldsymbol{j < n+2})$ *do*
  Step5.1. Take activity, j, $j = \overline{3,n}$ and find the best sequence by placing it in all possible positions of the sequence of activities that are already scheduled in the previous step.
  Step5.2. During this process, precedence constraints cannot be violated, and the relative order of already scheduled activities cannot be changed.
  Step5.3. Apply serial generation scheme to calculate the feasible time slot of an activity based on current resource availability and by earliest precedence.
  Step5.4. Set $j = j + 1$
Step6. Return the activity sequence and makespan, C_{max}
[End of Algorithm 1]

3.1.2. Critical path method (CPM) based heuristic

Delaying activities which are on the critical path (i.e., have zero slack time) may delay the schedule. Hence, the CPM heuristic identifies the activities on the critical path and schedules them as early as possible. The generated schedule is then improved by the insertion process by the fourth and fifth stage of the modified NEH heuristic. Fig. 2 presents a project network of eight activities. From the figure activities 1, 2, 4, 7, and 8 are critical activities. These activities are placed as soon as possible, and the generated partial sequence is 1-2-4-7-8. The remaining or non-critical activities are inserted one by one such that critical activities are placed as soon as possible while satisfying the precedence relationships. For example, consider activity 5 needs to be inserted into the partial schedule. Since activity 5 is the predecessor of activity critical activity 7, in the activity sequence it needs to be placed before activity 7. After inserting all the non-critical activities in the sequence, the final feasible activity sequence is 1-2-4-3-5-7-6-8 with makespan 29. Finally, after applying NEH-based insertion process the final makespan of the schedule is reduced to 23 (with sequence 1-3-6-5-2-4-7-8). The pseudo code of the CPM heuristic is shown in Algorithm 2. The relevant data of the example problem is available in Table A1 in the Appendix A.

Algorithm 2. CPM heuristic for RCPSPs.

Start of the algorithm
Input: The set of activities (n+2)
Step1. Apply CPM to identify the critical path and critical activities.
Step2. Generate a sequence by placing critical activities as soon possible and place non-critical activities in the activity sequence.
Step3. Repair the infeasible schedule with respect to precedence constraints.
Step4. The first and second activities from step 3 are selected and then calculate the makespan, C_{max} of this partial sequence by equation 1. Set j =3.
Step5. while $(\boldsymbol{j < n + 2})$ do
Step5.1. Take activity, j, $j = \overline{3,n}$ and find the best sequence by placing it in all possible positions of the sequence of activities that are already scheduled in the previous step.
Step5.2. During this process, precedence constraints cannot be violated, and the relative order of already scheduled activities cannot be changed.
Step5.3. Apply serial generation scheme to calculate the feasible time slot of an activity based on current resource availability and by earliest precedence.
Step5.4. Set $j = j + 1$
Step6. Return the activity sequence and makespan, C_{max}
[End of Algorithm 2]

3.2. Selection operator and generational scheme

In the proposed algorithm, a classical tournament selection process is employed for reproduction operations. After reproduction operations, the newly generated candidate solution replaces the worst member of the current generation if its makespan is better than that member. It is important to note that, in tightly constrained combinatorial optimization problems such as RCPSPs, after a few iterations many duplicate individuals are generated, which reduces the diversity of an algorithm and it converges to a local optimum. To avoid this problem, a candidate solution replaces the worst member in the generation if it is non-duplicate or unique. Note that, in RCPSPs there can be many different activity sequences with the same makespan (i.e. fitness). So, duplicate individuals that have same activity sequence and the sequence of activities of each member in the current generations are checked to identify duplicate individuals.

3.3. Similar block order crossover operator (SBOX) and shift mutation

In GA, the crossover operator plays a vital role in reproduction operation [43]. Our proposed MA utilizes a modified single point similar block order crossover (SBOX) [44] designed for flow shop scheduling problems for two reasons:

1. Preliminary experiments show that, with the evolution process there are many blocks of activities are generated within a sequence. These blocks are often broken apart by traditional single point crossover operation which results worst offspring.
2. Since precedence constraints are not violated within a block, maintaining the blocks in crossover process can reduces the violation of predecessor-successor relationships.

In SBOX, block of activities from both parents are identified and inserted directly to the same positions of each offspring. Then, the activities prior to the random crossover point from each parent are directly inserted to each offspring and finally, the missing activities of each offspring are inherited from the other parent. The motivation behind selecting SBOX over few other similar one-point crossover operators (i.e., similar job order crossover, similar point order crossover) are outlined in Section 4.1 (i.e., Fig. 6). Due to precedence constraints, step 2 and 3 of SBOX (Algorithm 3) operator cannot be used directly to RCPSPs. Hence, in step 2 and step 3 of modified SBOX, activities are inserted from parents to offspring in sequence while checking the predecessor-successor relationships and an activity is placed in an activity sequence only if the precedence constraints are not violated. An example of modified SBOX is shown in Fig. 3. Pseudo codes for that modified SBOX are also highlighted in Algorithm 3. From the Fig. 3, activity block 7-6-8 is directly inserted at the same position of both offspring without any modification. Activity 1, 2, 3 from parent 1 are inserted to offspring 1 and activity 1, 2, 4 are inserted from parent 2 to offspring 2. Finally, missing elements of both offspring are inherited from other parents, (i.e. activities 3 and 5 are inserted to offspring 1 from parent 2 and activity 4 and 5 are inserted to offspring 2 from parent 1) while considering precedence constraints as shown in Fig. 2.

Algorithm 3. Modified SBOX process.

Start of the algorithm
Input: The selected parents for crossover operation
Step 1: Within the range of $j = 1\ to\ n$, generate a crossover point
Step 2: Identify the block of activities by checking every position of activity sequence and inset these blocks directly at the same positions to offspring 1 and offspring 2.
Step 3. Insert the activities from parent 1 to offspring 1 and parent 2 to offspring 2 until the crossover point.
Step 4. Insert the missing activities of each offspring from other parent while checking the relative activity order and precedence constraints.
[End of the Algorithm 3]

In our proposed MA, shift mutation [45,46] is employed as mutation operator, since it shows promising performance in the preliminary experiments. In this mutation operation, an activity (except the first and last activity) from the activity sequence is randomly extracted and re-inserted in another randomly selected position while maintaining the precedence relationships.

3.4. Self-restart mechanism

In GA, the evolution process may lose diversity over generations and this may cause the algorithm to stall in a local optimum. To avoid this issue, the proposed MA measures the diversity value, ∂ of its population at each generation and it restarts when the diversity of the algorithm is below a restart threshold value, Δ. Our proposed restart mechanism is inspired by [47] approach, where a job/activity position-based restart mechanism was proposed for solving lot-streaming flow shop scheduling problem. However, they ignored the frequency of activities and block of activities in a set of solutions, which is an important factor to consider while measuring the diversity. Hence, in this study, we propose an improved and simple diversity measurement technique based on position and frequency of activities and activity blocks in the sequence, which is described as follows.

Algorithm 4. Self-restart mechanism.

Start of the algorithm
Input: Set of solutions from current generation.
Step1. Calculate position matrix $[PM_{jk}]_{n+2\times n+2}$ by

$$[PM_{jk}]_{n+2\times n+2}=\begin{bmatrix} PM_{1,1} & PM_{2,1} & \dots & PM_{1,n+2} \\ PM_{2,1} & PM_{2,2} & \cdots & PM_{2,n+2} \\ \dots & \dots & \dots & \dots \\ PM_{n+2,1} & PM_{n+2,2} & \dots & PM_{n+2,n+2} \end{bmatrix}$$

Where $[PM_{jk}]_{n+2\times n+2}$ counts the number of times jth activity is positioned in the kth position of the activity-sequence.
Step2. Calculate block matrix $[BM_{xy}]_{n+2\times n+2}$ by

$$[BM_{xy}]_{n+2\times n+2}=\begin{bmatrix} BM_{1,1} & BM_{2,1} & \dots & BM_{1,n+2} \\ BM_{2,1} & BM_{2,2} & \cdots & BM_{2,n+2,2} \\ \dots & \dots & \dots & \dots \\ BM_{n+2,1} & BM_{n+2,2} & \dots & BM_{n+2,n+2} \end{bmatrix}$$

Where $[BM_{xy}]_{n+2\times n+2}$ counts the number of times xth activity is positioned before the yth activity in the sequence.
Step3. Calculate the position frequency, ρ by

$$\rho=\left(\sum_{k=1}^{k=n+2}\sum_{j=1}^{j=n+2}1/PM_{jk}\right)/(n+2)^2 \qquad PM_{jk}\geq 1,\forall j\ and\ \forall k$$

Step4. Calculate the block frequency, ω by

$$\omega=\left(\sum_{x=1}^{x=n+2}\sum_{y=1}^{y=n+2}1/BM_{xy}\right)/(n+2)^2 \qquad BM_{xy}\geq 1,\forall x\ and\ \forall y$$

Step5. Calculate average diversity of population by

$$\partial=(\rho+\omega)/2$$

Step6. If $(\partial\leq\Delta)$ then divide all individuals of the current generation into three subsections: 10% individuals with best makespan so far, middle 50%, and worst 40%.
Step6.1. The individuals in the middle section are replaced by shift mutation process.
Step6.2. Worst individuals are replaced by random feasible solutions.
[End of Algorithm 3]

It is obvious that higher value of ∂ means the algorithm is more diversified (i.e. each activity occupies different positions of each sequence and there is less or no block of activities among the individuals). For example, consider three individuals of a generation with five activities: $s(1)=\{1,2,3,4,5\}$, $s(2)=\{1,3,2,4,5\}$, $s(3)=\{1,4,2,3,5\}$. First, $[PM_{jk}]_{n+2\times n+2}$ and $[BM_{xy}]_{n+2\times n+2}$ are calculated as follows.

$$[PM_{jk}]_{n+2\times n+2}=\begin{bmatrix} 3 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 2 & 1 & 0 & 0 \\ 0 & 0 & 1 & 2 & 0 \\ 0 & 0 & 0 & 0 & 3 \end{bmatrix} \quad \text{and} \quad [BM_{xy}]_{n+2\times n+2}=\begin{bmatrix} * & 1 & 1 & 1 & 0 \\ 0 & * & 2 & 1 & 0 \\ 0 & 1 & * & 1 & 1 \\ 0 & 1 & 0 & * & 2 \\ 0 & 0 & 0 & 0 & * \end{bmatrix}$$

Then, $\rho=0.267$ and $\omega=0.36$, and final diversity score of three individuals are $\partial=0.3135$.

Further, an elitism strategy is used to save the best individual from the current generation and transfer it to the next generation. In our proposed algorithm, we set elite size to one.

3.5. Local search operation

It is obvious to add a local search into MA to enhance intensification process. In our proposed MA, we employ a variable insertion neighborhood search (VINS) scheme as a local search operator, which has already been proved to be successful in solving other complex scheduling problems [48]. In traditional VINS mechanism, an activity is selected from kth position in the sequence and place in pth position. With this movement of a task, if the new activity sequence has a better fitness value than the current best activity sequence, then the current sequence is replaced by that sequence. The process continues until all $(n-1)^2$ are evaluated. However, due to precedence constraints and dummy activities in the sequence traditional VINS cannot be adopted for solving RCPSPs. Hence, we modified VINS algorithm in the following way: (1) first and last activities are dummy activities and their positions are fixed and excluded from the insertion operation, (2) an activity is inserted into all possible position which satisfies its

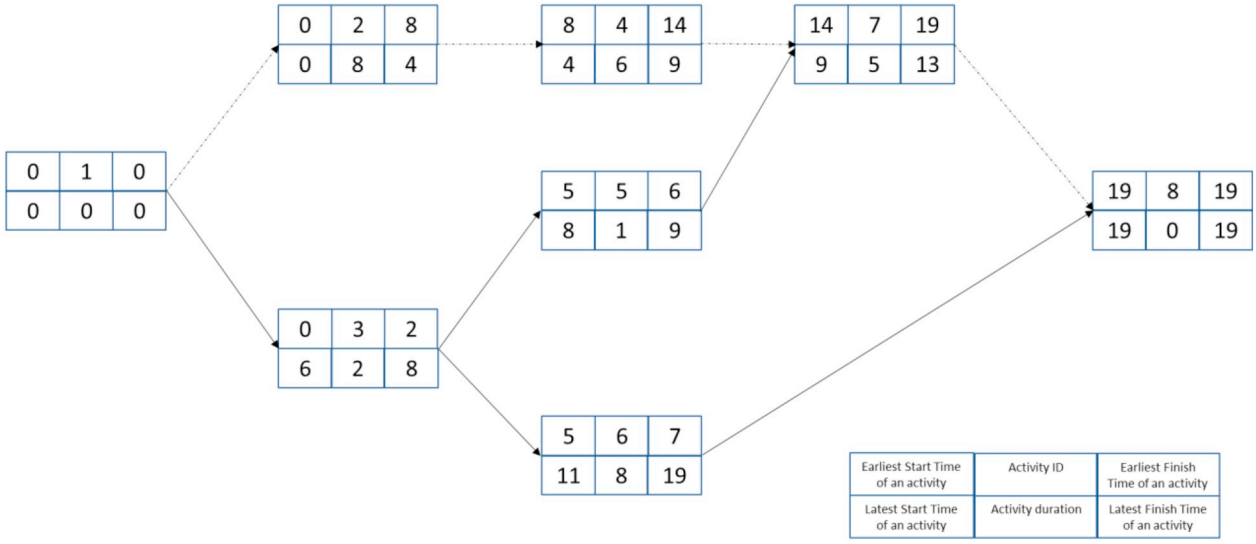


Fig. 2. Project network.

precedence constraints i.e. without violating its predecessor and successor relationships. For example, consider the sequence 1-2-4-3-5-7-6-8 (from Fig. 2 Section 3.1), where activity 3 is selected to be inserted in all possible feasible positions, which is shown in Fig. 4. In this case, either the second or third sequence can replace the original sequence since they provide a better makespan value ($26 < 29$) than the original sequence.

Considering all sub-algorithms and processes, the final pseudo code of the proposed GA-based MA is presented as follows:

Algorithm 4. A GA based MA for solving RCPSPs.

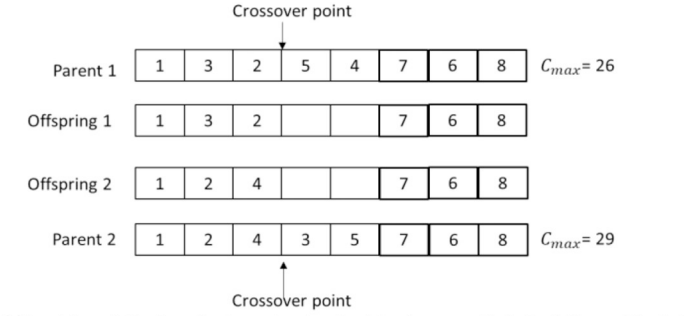
Input: All relevant paraments of the algorithms.
Step 1: Non-random initialization with the modified NEH algorithm, CPM based heuristics, six priority rules, and feasible random activity sequence.
Step2: $while\ (no.ofGen \leq max.Gen)$
  Step 2.1. $while\ (Popcount \leq n_p)$
    Step 2.1.1. Select two parents by tournament selection process.
    Step 2.1.2. Considering the probability of crossover, two offspring are generated from the two selected parents by the modified SBOX.
    Step 2.1.3. Considering the probability of mutation, selected offspring is mutated by the shift mutation scheme.
    Step 2.1.4. Based on the local search probability, the selected offspring is enhanced by the modified VINS scheme.
    Step 2.1.5. The worst member in the current generation is replaced by the better and non-duplicated offspring.
    Step 2.1.6. Set, $Popcount = Popcount + 2$
  Step 2.2. Apply elitism strategy to preserve and transfer the best member of the current generation to the next generation.
  Step 2.3. If $(\partial \leq \Delta)$ restart the algorithm.
  Step 2.4. Set, $no.ofGen = no.ofGen + 1$
Step 3. Save and return the **best** C_{max}
[End of the Algorithm]

3.6. Computational complexity analysis

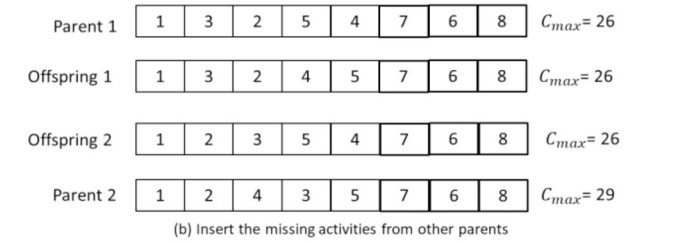
Let us assume that n_g is the number of generations and n_p is the number of populations per generation. The initial generation has the time complexity of $O(n^2)$, since the complexity of the modified NEH algorithm and CPH based heuristic has the complexity of $O(n^4K)$. The complexity of the modified SBOX operation is $O(nlogn)$. The selection and shift mutation require $O(n)$. The computational complexity of the self-restart mechanism is $O(n_p n^2)$ in the worst case. The modified VINS has the complexity of $O(n^4K)$. The overall complexity is $O[n_p(n^4K + (n_g - 1) * (nlogn + n + n^4K) + n^2)]$. Therefore, the overall complexity of the algorithm can be expressed as $O(n_g n_p n^4 K)$.

4. Experimental results and analysis

In this section, we present, discuss, and analyze the computational results obtained by the proposed GA-based MA on 2040 instances from the PSPLIB benchmark [14] datasets with J30, J60, J90 and J120. In addition, we have also solved the RG30 and RG300 problem sets taken from the http://www.projectmanagement.ugent.be/?q=research/data/RanGen weblink, which was primarily cited by Debels and Vanhoucke [16]. The J30, J60 and J90 datasets contain 480 problems each and J120 dataset contains 600 problem instances. The RG30 dataset contains 1800 instances and the RG300 dataset contains 480 instances. The complexity of these problem instances depends on three parameters: (1) network complexity (NC); (2) the resource factor (RF); and



(a) Insert the activities from direct parent up to cut point and common block of activities are inherited



(b) Insert the missing activities from other parents

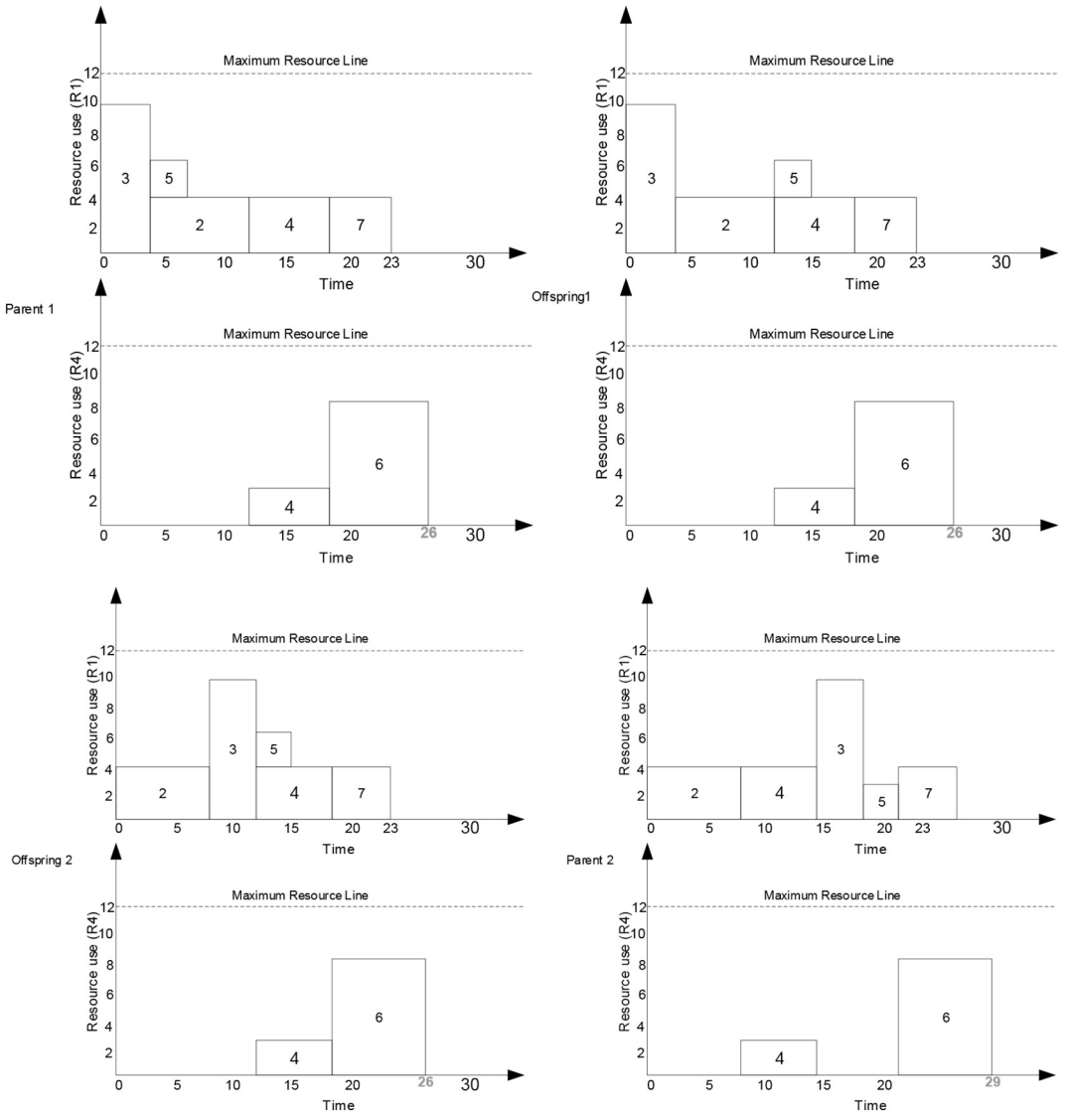


Fig. 3. Similar block order crossover (SBOX) for RCPSPs.

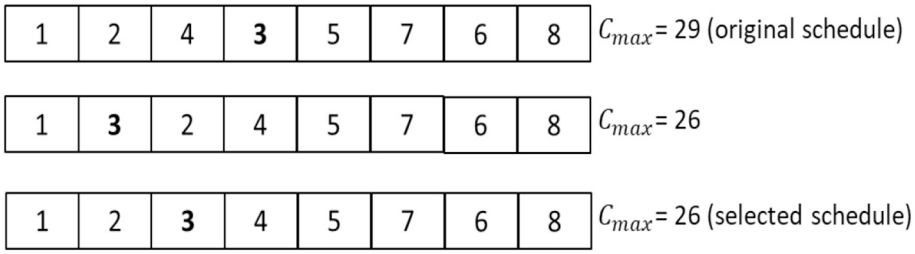


Fig. 4. VINS for RCPSPs.

Table 1
Combination of parameter values.

Parameters	Parameter level		
	1	2	3
Population size	20	30	50
Crossover probability (%)	50	70	90
Mutation probability (%)	5	10	15
Local search probability (%)	2	3	4
Restart threshold	0.02	0.06	0.08

resource strength (RS). In this paper, we compare the proposed MA with some typical state-of-the-art exact and heuristic algorithms in the literature. All of the numerical experiments were conducted on an Intel core i7 processor with a 3.40 GHz CPU and 16 GB RAM. The algorithm was coded in C++.

To evaluate the performance of our algorithm, we use the average percentage of deviation from the best-known solutions and the critical path based lower bound.

For PSPLIB benchmark instances, the average deviations from critical path (lower bound) can be calculated as:

$$ARD_{CP} = \sum_{p=1}^{p=P} \left[\frac{(C_{max,alg,p} - C_{max,CP,p})}{C_{max,CP,p}} \times 100\% \right] / p \quad (6)$$

Similarly, for RG benchmark instances, the average deviations from upper bound (best known solution) and lower bound can be calculated as:

$$ARD_{UB} = \sum_{p=1}^{p=P} \left[\frac{(C_{max,alg,p} - C_{max,UB,p})}{C_{max,UB,p}} \times 100\% \right] / p \quad (7)$$

$$ARD_{LB} = \sum_{p=1}^{p=P} \left[\frac{(C_{max,alg,p} - C_{max,LB,p})}{C_{max,LB,p}} \times 100\% \right] / p \quad (8)$$

where P is the maximum of problem instances from J30, J60, J90 and J120 problem and p^{th} problem instance: $C_{max,\ alg,\ p}$ is the makespan obtained by MA, $C_{max,\ CP,\ p}$ is the makespan obtained by critical path method while ignoring the violations of resource constraints, $C_{max,\ UB,\ p}$ is the best known or optimal makespan, $C_{max,\ UB,\ p}$ is the best known or optimal makespan, and $C_{max,\ LB,\ p}$ is the lower bound/critical path method value for each instances. The upper bound (UB) and lower bound (LB) values of RG instances are taken from the Excel file created by Vanhoucke and Coelho [49] and are downloaded from the weblink: http://www.projectmanagement.ugent.be/sites/default/files/files/datasets/RG300.zip.

4.1. Parameter analysis

As we mentioned earlier, the proposed MA has parameters that require calibration to obtain the best performance. In order to calibrate the algorithm, we use Taguchi's design of experiment (DOE) technique which reduces the computational times and required tests for identifying the best possible parameter combination for MA. The combination of parameters for MA is shown in Table 1. We employ the orthogonal array $L_{27}(3^5)$ since the number of parameters is 5 and each of the parameters has three levels. So, there are 27 treatments in the DOE, which are tested in 40 most complex problem instances from J60. Considering NC, RF, RS, these complex instances are: J60_9_1 to J60_11_10 and J60_13_1 to J60_14_10 [50]. For parameter tuning, we set 1000 schedules as the stopping criterion. The orthogonal array and the ARD_{CP} values for 27 treatment are listed in Table 2.

In accordance with to the orthogonal table the trend (response value and the significance rank) of parameters is listed and illustrated in Table 3 and Fig. 5 respectively. From Table 3, we can note that the local search probability is the most significant parameter. As expected, a large value of local search probability enhances the intensification process of the algorithm. In addition, mutation rate (ranked second) plays an essential role to maintain diversity of the evolution process. As well as mutation, the value for restart also plays an important role in maintaining diversity in the population. Hence, the choice of proper values for diversity threshold affect the convergence of the algorithm and the possibility of finding the global minimum. The population size is ranked in fourth position. Consequently, we can say that large population size may boost the optimization process but is computationally expensive. On the other hand, small population size may potentially be unable to evolve the algorithm which may become stuck in a local minimum. Finally, crossover rate has ranked in the last of the set. According to this analysis, a good choice of parameter can be suggested as: Population size = 50, crossover rate = 50%, mutation rate = 1.5%, local search probability = 4%, and restart threshold = 0.02.

Following the parameter analysis, we further investigate the performance of modified SBOX operator over its similar variants, e.g. single point similar job (i.e. activity) order crossover (SJOX) and single point ordered crossover (SPOX). To evaluate the performance, all optimal parameter combinations which are identified through DOE are fixed for MA, and the algorithm solved 40 most complex problem instances for J60 (i.e., J60_9_1 to J60_11_10 and J60_13_1 to J60_14_10) used for parameter tuning, under the same stopping condition. Fig. 6 shows $\boldsymbol{ARD_{CP}}$ values for three crossover operators and, as shown in the figure, SBOX outperforms the other two crossover operators.

4.2. Comparisons of MA with the existing powerful algorithms

This section provides a comparison of the proposed MA with some exact and powerful heuristic algorithms from the literature. For fair comparison, we set 5000 schedules generation in MA as the evaluation criteria, since this stopping criterion is widely used in the literature. Table 4 presents the performance of our proposed approach with respect to ARD_{CP} values and computational time, C_T for all 4320 (=2040 instances from PSPLIB + and 2280 instances from RG30 and RG300) problem instances. Considering these ARD_{CP} values, in Table 5, we demonstrate the comparisons between our proposed approach with the heuristic and metaheuristic algorithms available in the literature: consolidated optimization algorithm [26], PSO based hyper-heuristic algorithm (PSO-HH) of Koulinas et al. [30], the genetic algorithm for project scheduling (GAPS) of [18], the hybrid algorithm, ACOSS (combination of ant colony optimization, local search strategy and a

Table 2
Orthogonal table and ARD_{CP} values.

Experiment number	Parameters					ARD_{CP}
	Population size	Crossover rate	Mutation rate	Local search probability	Restart threshold	
1	20	50	0.05	2	0.02	12.11
2	20	50	0.05	2	0.05	12.05
3	20	50	0.05	2	0.08	11.77
4	20	70	1.00	3	0.02	11.37
5	20	70	1.00	3	0.05	11.76
6	20	70	1.00	3	0.08	11.75
7	20	90	1.50	4	0.02	11.30
8	20	90	1.50	4	0.05	11.46
9	20	90	1.50	4	0.08	11.62
10	30	50	1.00	4	0.02	11.16
11	30	50	1.00	4	0.05	11.38
12	30	50	1.00	4	0.08	11.83
13	30	70	1.50	2	0.02	11.77
14	30	70	1.50	2	0.05	11.68
15	30	70	1.50	2	0.08	12.02
16	30	90	0.05	3	0.02	11.69
17	30	90	0.05	3	0.05	11.70
18	30	90	0.05	3	0.08	11.86
19	50	50	1.50	3	0.02	11.51
20	50	50	1.50	3	0.05	11.50
21	50	50	1.50	3	0.08	11.35
22	50	70	0.05	4	0.02	11.42
23	50	70	0.05	4	0.05	11.66
24	50	70	0.05	4	0.08	11.46
25	50	90	1.00	2	0.02	11.91
26	50	90	1.00	2	0.05	11.82
27	50	90	1.00	2	0.08	11.87

Table 3
Response table for means.

Level	Population size	Crossover rate	Mutation rate	Local search probability	Restart threshold
1	11.69	11.63	11.75	11.89	11.59
2	11.68	11.66	11.65	11.62	11.67
3	11.62	11.70	11.58	11.48	11.73
Delta	0.08	0.06	0.17	0.41	0.14
Rank	4	5	2	1	3

scatter search) of [32], the genetic algorithm with forward-backward improvement (GA-FBI) of Gonçalves et al. [19], the scatter search electromagnetism meta-heuristic (SS-EM) of [51], the multiple justification PSO (MJPSO) algorithm of [52], the hybrid genetic algorithm of Valls et al. [17], the justification particle swarm optimization (JPSO) of [3], the Neurogenetic approach of [22], and event-based list and scatter search by Paraskevopoulos et al. [53].

From the results presented in Table 5, for J30 problem sets, our proposed algorithm almost always outperforms other approaches available in the literature. Furthermore, for J60, J90 and J120 problem sets, this proposed MA shows competitive performance comparing to all

Fig. 5. Factor level trend for each parameter.

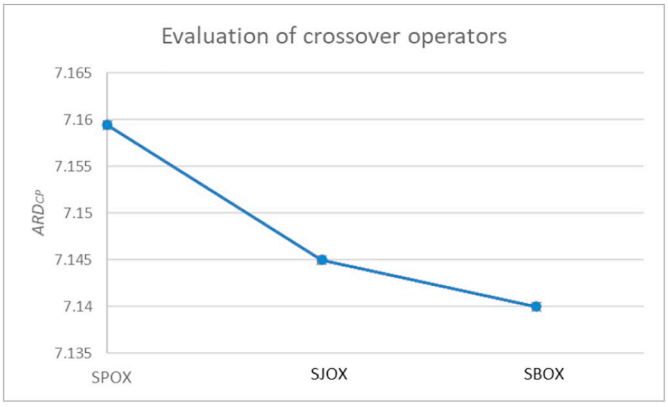


Fig. 6. Evaluation of different crossover operators.

Table 4
Performance of MA for PSPLIB datasets.

Algorithm	Schedule	Problem	J30	J60	J90	J120
MA	5000	ARD_{CP}	0.00	10.72	10.37	32.76
		Ct (second)	0.034	1.16	11.41	18.14
	50,000	ARD_{CP}	0.00	10.55	9.94	31.12
		Ct (second)	0.21	9.06	90.78	140.08

cited algorithms. In addition, to verify the performance of this proposed MA against large datasets, we have solved all available benchmark instances for RG30 and RG300 and compared with the average percentage deviation from critical path values (i.e., ARD_{cp}) against the reported results by Debels and Vanhoucke [16] and Vanhoucke and Coelho [49]. As reported earlier, the upper bound (UB) and lower bound (LB) values of RG instances are taken from the Excel file created by Vanhoucke and Coelho [49]. As can be seen from Table 6, the proposed MA could not outperform any of those available results for RG300 dataset. On the contrary, for RG30, ARD_{CP} values are almost similar to the ones reported in Vanhoucke and Coelho [49] (i.e., we obtained 39.34% deviations while the best reported is 39.33%). Meanwhile, the percentage deviations from the best-known UBs (ARD_{UB}) and LBs (ARD_{LB}) are only 1.64% and 7.28% respectively for RG300 instances after executing for 50,000 schedules. Overall deviations for RG30 are also very insignificant for 50,000 schedules (0.01% and 0.40% respectively). Hence, even though our proposed MA did not obtain any improved solutions or upper bounds compared to existing approaches, we can claim that this proposed approach is highly competitive and an effective alternative approach to solve harder and larger activity-based instances. Critical path values (CP), UBs, LBs and obtained makespan values for each of the RG30 and RG300 instances are available on request to the corresponding author.

In addition, even though the homogeneous interval algorithm (HIA) of Valls et al. [54] and large neighborhood search of Palpant et al. [34] did not provide any information about number of schedule as the stopping condition, our proposed algorithm still shows better performance than those approaches. In a nutshell, MA shows exceptional consistency and faster solving capability while solving RCPSPs.

4.3. Statistical analysis

Since the proposed MA shows a stochastic nature, to better reflect any statistical differences among existing algorithms, this work carried out a number of statistical and nonparametric tests. As illustrated in the previous section, under all three different instance sets, there is no firm evidence on the superiority of any algorithm over the others. Hence to assist decision making and to evaluate the overall performance of the proposed MA, we have executed a ranking procedure based on a nonparametric test, called the Friedman test [57]. The Friedman test is often used for a randomized complete block design when the normality assumption is not satisfied, or the data is ordinal. Under each instance type, for each of the algorithms, relative ranks are calculated by considering ARD_{cp}. All relative ranks are addressed in Table 7. As evident, MA exhibits the *sec*ond-least rank value of 3.33 after GH-SS among all recent algorithms. Hence, it can be claimed that the performance of MA is highly competitive and consistent among all statistically dispersed results.

For better justification, we have also executed the Wilcoxon signed ranks test [58] to analyze the differences between the best and the near-best results according to Table 7 (i.e., MA, GH-SS and SS-FBI). Using a 5% significance level, the obtained p values are: 0.317 (GH-SS - MA), 0.180 (SS-FBI - MA) and 0.180 (SS-FBI – GH-SS). Meanwhile, based on the relative positive and negative ranks, the Z values are as follows: $Z_{(GH\_SS-MA)} = -1.00$, $Z_{(SS\_FBI-MA)} = -1.342$ and $Z_{(SS\_FBI-GH\_SS)} = -1.342$. Hence, it can be ensured that there is no unitary evidence of significant dominance of any algorithm over the others (as $p > 0.05$), even though the proposed MA algorithm has been outperformed by very few existing algorithms (i.e., GH_SS), particularly for the larger instance set (i.e., J120).

Table 5
Comparison of the average deviation (%) among PSPLIB datasets.

Algorithms [reference]	J30		J60		J90		J120	
	5000	50,000	5000	50,000	5000	50,000	5000	50,000
	ARD_{CP}	ARD_{CP}	ARD_{CP}	ARD_{CP}	ARD_{CP}	ARD_{CP}	ARD_{CP}	ARD_{CP}
MA [our proposed approach]	0.00	0.00	10.72	10.55	10.37	9.94	32.76	31.12
COA [26]	0.00	0.00	10.77	10.58	–	–	32.90	31.22
GH-SS (LS) [53]	0.01	0.00	10.72	10.55	–	–	32.12	30.78
Sequential (SS-FBI) [55]	0.02	0.00	10.93	10.58	10.40	9.96	32.52	31.16
GA [21]	0.07	0.01	11.08	10.71	–	–	33.36	31.81
PSO-HH [30]	0.04	0.01	11.13	10.68	–	–	32.59	31.23
GA-FBI [19]	0.01	0.01	11.56	10.56	–	–	35.94	32.76
ACOSS [32]	0.04	0.01	10.98	10.68	–	–	32.48	30.56
GAPS [18]	0.01	0.01	11.04	10.67	–	–	33.03	31.44
SS-EM [51]	0.11	0.01	11.71	10.71	–	–	33.57	31.57
MJPSO [52]	0.02	0.02	11.19	10.85	–	–	33.78	32.40
HGA [17]	0.06	0.02	11.10	10.73	–	–	32.54	31.24
GA-FBI2 [16]	0.04	0.02	10.95	10.68	10.35	9.90	32.18	30.69
JPSO [3]	0.14	0.04	10.43	11.00	–	–	33.88	32.89
GANS [56]	1.27	0.71	10.53	10.52	–	–	31.51	30.45
NG-FBI [22]	0.10	–	11.29	–	11.29	–	34.15	–

Table 6
Performance of MA for RG instances.

Schedule	Problem	RG30						RG300
		S1	S2	S3	S4	S5	All	
5000	ARD_{UB}	0.10%	0.14%	0.08%	0.16%	0.14%	0.12%	3.63%
	ARD_{LB}	0.42%	0.77%	0.08%	0.91%	0.39%	0.52%	9.44%
	ARD_{CP} [this paper]	42.82%	45.68%	34.31%	34.80%	34.80%	39.52%	984.43%
	ARD_{CP} [16]	–	–	–	–	–	–	817.36%
	ARD_{CP} [49]	–	–	–	–	–	–	–
	Ct (sec)	0.042	0.044	0.041	0.043	0.043	0.043	29.07
50,000	ARD_{UB}	0.01%	0.02%	0.003%	0.01%	0.01%	0.01%	1.64%
	ARD_{LB}	0.33%	0.65%	0.003%	0.77%	0.26%	0.40%	7.28%
	ARD_{CP} [this paper]	42.67%	45.45%	34.18%	34.58%	34.58%	39.34%	968.57%
	ARD_{CP} [16]	–	–	–	–	–	–	809.93%
	ARD_{CP} [49]	–	–	–	–	–	39.33%	956.71%
	Ct (sec)	0.287	0.289	0.301	0.310	0.278	0.293	202.14

For better illustration, Table 7 has been used to generate performance profiles and to execute nonparametric tests for the PSPLIB datasets. Fig. 7 represents the performance profile for the PSPLIB dataset (based on the results obtained after 50,000 schedules), which has been used as a tool for evaluating and comparing the performance of optimization algorithms. If the set of problems P is suitably large and representative of problems that are likely to occur in applications, then algorithms with large probability $\rho_s(\tau)$ are to be preferred. The performance profile for an algorithm is the (cumulative) distribution function for a performance metric. For a better understanding of performance profiles, see the work of Dolan and Moré [59]. In this paper we use the ARD_{cp} as a performance metric. As reflected from Fig. 7, the proposed MA algorithm has the second-best probability (after GH + SS approach) of obtaining smallest deviations and converges fast towards 100% than a significant number of existing algorithms.

4.4. Case study

To demonstrate the effectiveness of the proposed MA in solving real-life problems, we have also considered five real-life projects from three different areas: construction industry, information technology, and healthcare systems. As reflected in Table 8, cases 1–3 are directly adapted from their cited references. Meanwhile, case 4 (Commercial IT Project-CIT), case 5 (Patient Transport System-PTS), and case 6 (Nursing Home Noordhinder) are taken from the online database (http://www.or-as.be/research/database) [60]. The PTS project deals with the design, installation (including staff training) and evaluation of an ICT-supported patient transport system in the general hospital Sint-Jan in Bruges (Belgium). The overall project networks and resource distributions are provided in Appendix A and Tables A2 and A3 respectively. Meanwhile the CIT project deals with the complete developing process of a software program for magazine publishers, including a preliminary market analysis. The project consists of activity, resource and cost data that were created by the user. The overall project networks and resource distributions are also provided in Appendix A and Tables A4 and A5 respectively. The Nursing Home Noordhinder is a construction project for building a nursing home in Knokke-Heist (Belgium). Project card excel file and all relevant information of this case study can be found from this online source: http://www.or-as.be/research/database.

The performance of the proposed approach has been compared with an exact approach (i.e., B&B algorithm) and with the existing approaches available in the literature. The stopping criteria for our proposed algorithm is set to 100 schedules evaluation and B&B algorithm was coded with CPLEX embedded in Matlab R2017b. Each of the algorithms was run for 30 times and the results are tabulated in Table 8. However, owing to the inability to solve higher activity instances (i.e., Case 2, 4, 5, and 6) by B&B exact approach, we have highlighted the best obtained results after 1000 s of execution. As can be seen from that table, the proposed algorithm shows promising results in finding optimal makespan with significantly lower computational effort (i.e., only within 100 schedule evaluations), even for case studies with a higher number of activities.

5. Applicability of the proposed approach in construction project scheduling

In the construction industry, project management decisions relevant to time and budget are made based on schedules that are developed in the planning phase of a project. Hence, one of the challenging tasks for the construction project manager is to minimize the total project duration with limited resource availability. Although one traditional project scheduling technique, namely CPM, has been used as a scheduling tool in the construction industry since the 1950s [63], it can provide optimal solutions when the availability of resource is unlimited. However, as discussed in Section 2, adding resource constraints to project scheduling problem makes it NP-hard and as shown from the example in Section 3.1, CPM based schedule can be possibly improved in RCPSPs. In today's competitive construction industry, the availability of resources should be limited (e.g., construction equipment, workers) to make it profitable and therefore, it is essential to develop an efficient decision-making tool that can provide optimal or near optimal schedules in reasonable computational time. Another important aspect is that the complexity in modern projects varies with respect to the number of activities, resources, maximum resource availabilities, and project managers need to deal with such complex project network frequently [64]. Therefore, it is desirable to have a solution approach that can provide competitive solutions, within computational time, for project scheduling problems with any degree of complexity.

To serve these purposes, we propose a MA that is easy to implement. The performance of the proposed approach has been evaluated with a wide range of complex scheduling problems, from two different artificial standard benchmark data (PSPLIB and RG). Furthermore, we have

Table 7
Rank measurement of different algorithms using Friedman test.

MA	COA	GH-SS	SS-FBI	GA	PSO-HH	GA-FBI	ACOSS	GAPS	SS-EM	MJPSO	HGA	GA-FBI2	JPSO	GANS
3.33	5.00	**3.00**	**4.67**	10.33	8.17	8.50	6.17	8.17	10.00	13.00	11.33	8.00	14.67	5.67

The bold values signifies the first, second, and third best performing algorithm algorithms among the list using the Friedman test.

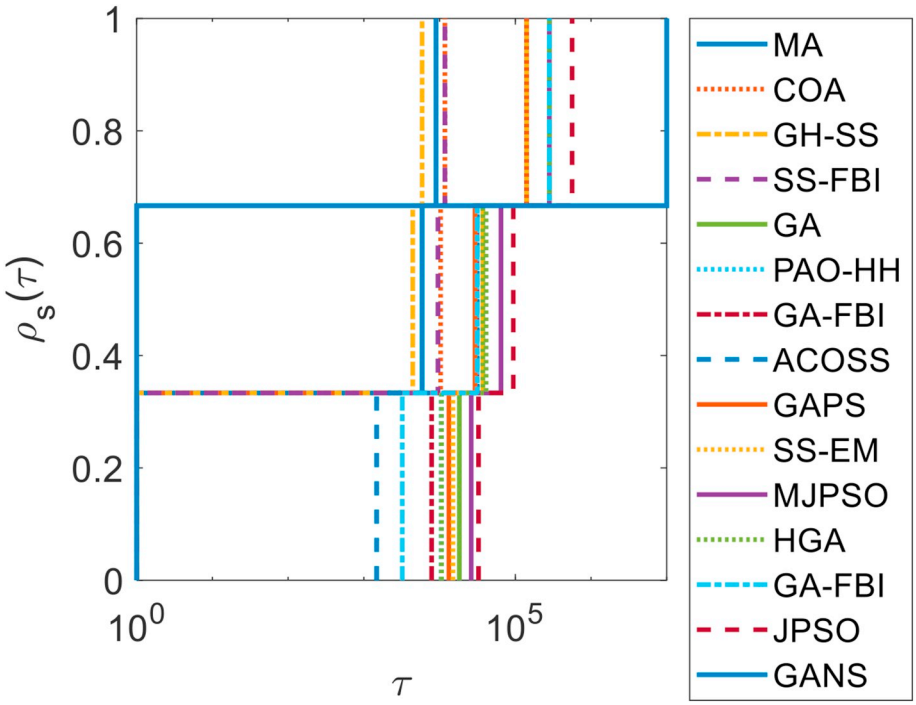


Fig. 7. Performance profile of different competitive algorithms.

evaluated our proposed approach for solving practical project scheduling problems, where relevant empirical data were collected from the construction and other industries (as discussed in Section 4). From the experimental evaluations it can be safely said that for all these set of problems, our proposed approach shows competitive performance in terms of solution quality and computational speed. In a nutshell, adaptation of our proposed approach can be a useful tool for practitioners to significantly reduce time and financial losses, and to predict future budgets more carefully.

Table 8
Result comparison between MA and Other approaches in practical problems.

Problem [source]	Problem size		Average C_{max} value			Average computational times (seconds)		
	Number of activities	Number of resources	Existing approach	B&B	MA	Existing approaches	B&B	MA
Case 1: construction project [24,61]	20	1	42.53 (PSO) 42.51 (GA) 42.50 (DE) 42.07 (CDE) 42.17 (FDE) 42 (FCDE-RCPSP)	42	42	Not available	68.87	0.013
Case 2: construction project [24,61]	37	1	192.47 (PSO) 191.33 (GA) 191.27 (DE) 190.47 (CDE) 191.20 (FDE) 190.13 (FCDE-RCPSP)	236[a]	190	Not available	1000	0.015
Case 3: highway bridge construction project [62]	43	3	118.25 (PSO) 118.35 (GA) 117.75 (DE) 117.7 (ABC) 117 (ABCDE-RC)	117	117	Not available	100.52	0.041
Case 4: commercial IT project [60]	42	8	Not available	340	309	Not available		0.038
Case 5: patient transport system [60]	49	5	Not available	1136[a]	1068	Not available	1000	0.047
Case 6: nursing home noordhinder [60]	153	12	Not available	10931[a]	10,676	Not available	1000	0.12

[a] Best obtained result after 1000 s.

6. Conclusion and future research directions

Despite of significant number of approaches to solve RCPSPs over the decades, no single algorithm can offer sound performance in terms of accuracy and speed for all range of RCPSP datasets. Often, these algorithms have complex structures which makes their implementation difficult. As a result, in this study, we bring forward a unique and competitive GA-based MA algorithm in the quest for better optimization performance. This proposed approach initiates with a mix of quality solutions generated by two simple heuristics and some priority rules, and some random solutions, which eliminates the drawback of random initialization of the original GA. During the evaluation process a single-point similar block order crossover finds the simpler blocks of activities which act as the building blocks of quality solutions. This crossover operator has shown better performance than the traditional single point ordered crossover operator. Further, a shift mutation operation ensures population diversity and a self-restart mechanism measures population diversity at each iteration and restarts automatically to avoid local optima, when needed. Therefore, this mechanism helps MA to avoid entanglement in the local optima. Further, a variable insertion neighborhood search was introduced to further enhance performance, which show excellent performance, considering their simplicity and speed. More importantly, the proposed algorithm is simple and easy to implement, and generates competitive solutions.

The parameters of the algorithm were adjusted by DOE and the effectiveness of the algorithm was verified by solving the popular PSPLIB and RG benchmark problems, with J30, J60, J90, J120, RG30 and RG300 problem sets. The results showed that the proposed algorithm is capable of converging to high quality solutions within reasonable computational efforts, especially for J30, J60 and J90 sets. Further, the proposed algorithm was compared with the state-of-the-art algorithms in terms of solution quality and it showed best performances for the lower activity problem sets (i.e., for J30, J60 and J90 instances) and promising performance for J120, RG30 and RG300 problem sets. Six real-world projects from three different areas are also used to evaluate the comparative effectiveness of the proposed algorithm in handling real-world resource constrained problems.

In future work, further investigations are needed to accelerate the performance of the algorithm for J120 and RG300 problem sets. The work presented in this paper is very relevant to modern construction and manufacturing industries to assist project managers with real-time decision support, to allow project scheduling decisions to be taken in a short computation time within the resource constraints. The developed MA can be employed by project managers to execute projects more efficiently and utilizing the available resources more effectively, and thus reduce project execution costs. Hence, the proposed approach assists project managers obtain a satisfactory schedule within reasonable computational time.

Moreover, future studies might extend the presented algorithms to other interesting project scheduling problems from real-life, such as multi-mode resource constraint project scheduling problems. Another research opportunity is to study the extension of existing problems while considering multiple temporal constraints such as including aging and learning effects in activity durations, transportation time, alternative supplier shipping times, and project inspection time, and so on. There are also multiple conflicting objectives, including minimizing makespan and maximizing net present value, that need to be optimized simultaneously in real applications. Finally, it is also observed that in industry there are uncertainties in project durations and resource availability and its utilization, and hence it will be interesting to extend our proposed method to study these realistic cases.

Declaration of competing interest

These authors declare that they have no conflict of interests on the work reported in this paper.

Appendix A

Data for example problem
Number of activities: 8
Maximum available resources: 4
Resource availabilities: [12 13 4 12]

Table A1
Activity durations and successors.

Activity ID	Activity duration	Resource use				Number of successors	Successors	
		R1	R2	R3	R4			
1	0	0	0	0	0	2	2	3
2	8	4	0	0	0	1	4	
3	4	10	0	0	0	2	5	6
4	6	4	0	0	3	1	7	
5	3	3	0	0	0	1	7	
6	8	0	0	0	8	1	8	
7	5	4	0	0	0	1	8	
8	0	0	0	0	0	0		

Table A2
Patient transport system.

ID	Name	Successors	Duration
1	Compose survey for the staff	FS2 + 1 w 1 d	4 h
2	Brainstorming and filling in the survey	FS3 + 7 w 2 h	2 h
3	Collecting and analysing the surveys	FS4	1 d
4	Get-together PC, ICT committee & board	FS5 + 2 h	4 h
5	Enrolment of the meeting	FS6	2 h
6	Registering the technical requirements	FS7 + 2 w	2 d
7	Set-up a detailed budget study	FS8 + 3 w	2 d
8	Putting together the formal document	FS9;FS20 + 1 w	3 d
9	Research suppliers transport systems	FS10 + 1 w 1 d 4 h	3 d
10	Meeting with board for final proposal	FS11 + 2 w	4 h
11	Contacting suppliers and sending proposal	FS12 + 3 w	2 d
12	Analysing the received offers	FS13 + 4 w	3 d
13	Presentation to board & ICT staff	FS14 + 3 w	1 d
14	Collecting and registering comments on offers	FS15 + 6 w	1 d
15	Taking final decision and contacting supplier	FS16	2 d
16	Discussing project development with supplier	FS17 + 7 w 3 d 4 h	3 d
17	Discussing project development with board	FS18 + 6 w	4 h
18	Final agreement with supplier	FS19	2 d
19	Signing contract	FS27 + 4 d 4 h	4 h
20	Meeting with ICT committee	FS21 + 1 w 2 d	2 d
21	Contacting Siemens and sending proposal	FS22 + 2 w	3 d
22	Discussing project development with supplier	FS23 + 1 w	2 d
23	Discussing progress with board	FS24 + 2 d	2 h
24	Discussing progress with ICT committee	FS25 + 1 d 6 h	4 h
25	Final agreement with supplier	FS26	4 h
26	Signing contract	FS27 + 8 w	4 h
27	Information session	FS28 + 4 h	2 h
28	Explanation project implementation to staff	FS29;FS30;FS31;FS34	2 h
29	Installation transport system	FS36	6 d
30	Installation DACS	FS38;FS44	6 d
31	Training coordinator	FS32;FS36	1 d
32	Train de trainer	FS38;FS44	1 d
33	Training rapportage	FS47	1 d
34	Surveying the required functionalities	FS35 + 4 w 4 d	1 d
35	Creating codes	FF37 + 1 w	2 d
36	Registration data about employees	FS37 + 3 w 2 d	4 d
37	Registration hospital data	FS38;FS44	6 d
38	Training team 1	FS39	2 d
39	Training team 2	FS40	2 d
40	Training team 3	FS41	2 d
41	Training team 4	FS42	2 d
42	Training team 5	FS43	2 d
43	Training team 6	FS46	2 d
44	Check-up by PC	FS45 + 2 w	1 d
45	Check-up by ICT committee	FS46	1 d
46	Live pilot period	FS33	71 d
47	Discussing project with board	FS48 + 4 d 2 h	2 h
48	Discussing project with head nurses	FS49	4 h
49	Enrolment of the entire project		1 d

Table A3
Resource distribution for patient transport system.

General				Resource demand
ID	Name	Type	Availability	Assigned to
1	Project co-ordinator	Renewable	1 #persons	1;3;4;5;6;7;8;9;10;11;12;13;14;15;16;17;18;19;20;21;22;23;24;25;26;27;28;31;32;33;34;35;36;37;38;39;40;41;42;43;44;47;48;49;
2	Head nurses	Renewable	65 #persons	2[65.00 Head nurses];28[65.00 Head nurses];40[17.00 Head nurses];41[16.00 Head nurses];42[16.00 Head nurses];43[16.00 Head nurses];48[65.00 Head nurses];
3	ICT committee	Renewable	5 #persons	4[5.00 ICT committee];6;10;13[5.00 ICT committee];20[5.00 ICT committee];24[5.00 ICT committee];27[5.00 ICT committee];29[2.00 ICT committee];30[2.00 ICT committee];35;45[5.00 ICT committee];
4	Board of directors	Renewable	7 #persons	4[7.00 Board of directors];10[7.00 Board of directors];13[7.00 Board of directors];17[7.00 Board of directors];19;23[7.00 Board of directors];26;27[7.00 Board of directors];47[7.00 Board of directors];
5	Stretcher-bearer/logisticians	Renewable	24 #persons	2[24.00 Stretcher-bearer/logisticians];28[24.00 Stretcher-bearer/logisticians];34[24.00 Stretcher-bearer/logisticians];38[12.00 Stretcher-bearer/logisticians];39[12.00 Stretcher-bearer/logisticians];

Table A4
Commercial IT (CIT) project.

ID	Name	Predecessors	Successors	Duration
0	Commercial IT project			56 d 5 h
22	Explore the market		FS23	10 d
23	Set up a survey	22FS	FS24	3 d
24	Analysis of the market	23FS	FS1	15 d
1	Problem study/analysis	24FS	FS2;FS3;FS39;FS40	3 d
39	Negotiate and buy server	1FS	FS41	2 d
40	Negotiate and buy computer	1FS	FS42	2 d
2	Processmodels (BPMN)	1FS	FS4;FS7	7 h
3	Conceptual datamodel (ER)	1FS	FS4;FS7	1 d
4	Normalisation DB	2FS;3FS	FS5	4 h
5	Datamap	4FS	FS6	1 h
6	Logic database model	5FS	FS8;FS9	2 h
7	Software design (UML-class diagram)	2FS;3FS	FS10;FS11	2 d
8	Develop database Table 1	6FS	FS12	6 h
9	Develop database Table 2	6FS	FS13	1 d
10	Design & develop GUI 1	7FS	FS18;FS25;FS26	2 d
11	Design & develop GUI 2	7FS	FS19;FS25;FS26	2 d 5 h
25	Design CD-box	10FS;11FS		10 d
26	Design CD-labels	10FS;11FS		10 d
12	Develop SQL's 1	8FS	FS14	2 d 2 h
13	Develop SQL's 2	9FS	FS15	2 d 4 h
14	Debug SQL's 1	12FS	FS16	1 d 2 h
15	Debug SQL's 2	13FS	FS17	1 d 4 h
16	Implement SQL's in JAVA 1	14FS	FS18	1 h
17	Implement SQL's in JAVA 2	15FS	FS19	1 h
18	Develop program 1	16FS;10FS	FS20	3 d
19	Develop program 2	17FS;11FS	FS21	3 d 5 h
20	Testing & debugging 1	18FS	FS27;FS37;FS38	1 d
21	Testing & debugging 2	19FS	FS27;FS37;FS38	1 d
38	Visitations by sales representatives	20FS;21FS		10 d
37	Direct mailings	20FS;21FS		2 h
27	Make up report + presentation alpha version	20FS;21FS	FS28	2 d
28	Presentation alpha version	27FS	FS29	2 h
29	Reading of management report + approval alpha	28FS	FS31	5 h
31	Testing phase alpha version	29FS	FS32	3 d
32	Fix outstanding problems from alpha	31FS	FS33	1 d 2 h
33	Make up report + presentation beta version	32FS	FS34	1 d
34	Presentation beta version	33FS	FS35	2 h
35	Reading of management report + approval beta	34FS	FS36;FS41;FS42	5 h
36	Launch and installation beta version	35FS		5 d
41	Installation server	35FS;39FS		5 d
42	Installation computer	35FS;40FS		5 d

Table A5
Resource distribution for commercial IT (CIT) project.

General				Resource demand
ID	Name	Type	Availability	Assigned to
1	Programmers	Renewable	3 Programmers	10;11[1.50Programmers];16[2.00Programmers];17[2.50 Programmers];18[2.00 Programmers];19[3.00 Programmers];20[2.00 Programmers];21[3.00 Programmers];31[2.00 Programmers];32;
2	IT Manager	Renewable	1 IT manager	1[0.20 IT Manager];39[0.10 IT Manager];40[0.10 IT Manager];28;29[0.50 IT Manager];34;35[0.50 IT Manager];
3	Database administrator	Renewable	1 Database administrator	4;5[0.50 Database administrator];6;8;9;12[0.40 Database administrator];13[0.60 Database administrator];14[0.40 Database administrator];15[0.60 Database administrator];31[0.30 Database administrator];
4	Analyst programmer	Renewable	1 Analyst programmer	1;2;3;5[0.50 Analyst programmer];7[0.30 Analyst programmer];27[0.30 Analyst programmer];28;31[0.10 Analyst programmer];33[0.20 Analyst programmer];34;
5	Secretary	Renewable	1 Secretary	37;27;33;
6	Marketing manager	Renewable	1 Marketing manager	39[0.50 Marketing manager];40[0.50 Marketing manager];37[0.50 Marketing manager];
7	Sales representatives	Renewable	2 Sales representatives	38[2.00 Sales representatives];
8	Technicians	Renewable	3 Technicians	36[3.00 Technicians];41;42;

References

[1] P. Brucker, Scheduling and constraint propagation, Discret. Appl. Math. 123 (1–3) (2002) 227–256, https://doi.org/10.1016/S0166-218X(01)00342-0.

[2] J. Blazewicz, J.K. Lenstra, A.R. Kan, Scheduling subject to resource constraints: classification and complexity, Discret. Appl. Math. 5 (1) (1983) 11–24, https://doi.org/10.1016/0166-218X(83)90012-4.

[3] R.-M. Chen, Particle swarm optimization with justification and designed mechanisms for resource-constrained project scheduling problem, Expert Syst. Appl. 38 (6) (2011) 7102–7111, https://doi.org/10.1016/j.eswa.2010.12.059.

[4] A. Mingozzi, V. Maniezzo, S. Ricciardelli, L. Bianco, An exact algorithm for the resource-constrained project scheduling problem based on a new mathematical formulation, Manag. Sci. 44 (5) (1998) 714–729, https://doi.org/10.1287/mnsc.

44.5.714.
[5] U. Dorndorf, E. Pesch, T. Phan-Huy, A branch-and-bound algorithm for the resource-constrained project scheduling problem, Math. Meth. Oper. Res. 52 (3) (2000) 413–439, https://doi.org/10.1016/S0377-2217(97)00335-4.
[6] P. Brucker, S. Knust, A. Schoo, O. Thiele, A branch and bound algorithm for the resource-constrained project scheduling problem, Eur. J. Oper. Res. 107 (2) (1998) 272–288, https://doi.org/10.1016/S0377-2217(97)00335-4.
[7] R.K. Chakrabortty, R.A. Sarker, D.L. Essam, Resource constrained project scheduling with uncertain activity durations, Comput. Ind. Eng. 112 (2017) 537–550, https://doi.org/10.1016/j.cie.2016.12.040 10/01/2017.
[8] R. Chakrabortty, R.A. Sarker, D.L. Essam, Event based approaches for solving multi-mode resource constraints project scheduling problem, IFIP International Conference on Computer Information Systems and Industrial Management, Vietnam, 8838 Springer, Berlin, Heidelberg, 2014, pp. 375–386, , https://doi.org/10.1007/978-3-662-45237-0_35.
[9] J. Coelho, M. Vanhoucke, An exact composite lower bound strategy for the resource-constrained project scheduling problem, Comput. Oper. Res. 93 (2018) 135–150, https://doi.org/10.1016/j.cor.2018.01.017 05/01/2018.
[10] A. Lova, P. Tormos, M. Cervantes, F. Barber, An efficient hybrid genetic algorithm for scheduling projects with resource constraints and multiple execution modes, Int. J. Prod. Econ. 117 (2) (2009) 302–316, https://doi.org/10.1016/j.ijpe.2008.11.002.
[11] D.H. Wolpert, W.G. Macready, No free lunch theorems for optimization, IEEE Trans. Evol. Comput. 1 (1) (1997) 67–82, https://doi.org/10.1109/4235.585893.
[12] R. Pellerin, N. Perrier, F. Berthaut, A survey of hybrid metaheuristics for the resource-constrained project scheduling problem, Eur. J. Oper. Res. 280 (2) (2020) 395–416, https://doi.org/10.1016/j.ejor.2019.01.063.
[13] M. Nawaz, E.E. Enscore Jr., I. Ham, A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem, Omega 11 (1) (1983) 91–95, https://doi.org/10.1016/0305-0483(83)90088-9.
[14] R. Kolisch, A. Sprecher, PSPLIB—a project scheduling problem library: OR software-ORSEP Operations Research Software Exchange Program, Eur. J. Oper. Res. 96 (1) (1997) 205–216 1997/01/10/ https://doi.org/10.1016/S0377-2217(96)00170-1.
[15] J. Alcaraz, C. Maroto, A robust genetic algorithm for resource allocation in project scheduling, Ann. Oper. Res. 102 (1–4) (2001) 83–109, https://doi.org/10.1023/A:1010949931021.
[16] D. Debels, M.J.O.R. Vanhoucke, A decomposition-based genetic algorithm for the resource-constrained project-scheduling problem, Oper. Res. 55 (3) (2007) 457–469, https://doi.org/10.1287/opre.1060.0358.
[17] V. Valls, F. Ballestin, S. Quintanilla, A hybrid genetic algorithm for the resource-constrained project scheduling problem, Eur. J. Oper. Res. 185 (2) (2008) 495–508, https://doi.org/10.1016/j.ejor.2006.12.033.
[18] J.J. Mendes, J.F. Gonçalves, M.G. Resende, A random key based genetic algorithm for the resource constrained project scheduling problem, Comput. Oper. Res. 36 (1) (2009) 92–109, https://doi.org/10.1016/j.cor.2007.07.001.
[19] J.F. Gonçalves, M.G. Resende, J.J. Mendes, A biased random-key genetic algorithm with forward-backward improvement for the resource constrained project scheduling problem, J. Heuristics 17 (5) (2011) 467–486, https://doi.org/10.1007/s10732-010-9142-2.
[20] R. Zamani, A competitive magnet-based genetic algorithm for solving the resource-constrained project scheduling problem, Eur. J. Oper. Res. 229 (2) (2013) 552–559, https://doi.org/10.1016/j.ejor.2013.03.005.
[21] R. Zamani, An evolutionary implicit enumeration procedure for solving the resource-constrained project scheduling problem, Int. Trans. Oper. Res. 24 (6) (2017) 1525–1547, https://doi.org/10.1111/itor.12196.
[22] A. Agarwal, S. Colak, S. Erenguc, A neurogenetic approach for the resource-constrained project scheduling problem, Comput. Oper. Res. 38 (1) (2011) 44–50, https://doi.org/10.1016/j.cor.2010.01.007.
[23] K. Anagnostopoulos, G. Koulinas, Resource-constrained critical path scheduling by a GRASP-based hyperheuristic, J. Comput. Civ. Eng. 26 (2) (2011) 204–213, https://doi.org/10.1061/(ASCE)CP.1943-5487.0000116.
[24] M.-Y. Cheng, D.-H. Tran, Y.-W. Wu, Using a fuzzy clustering chaotic-based differential evolution with serial method to solve resource-constrained project scheduling problems, Autom. Constr. 37 (2014) 88–97, https://doi.org/10.1016/j.autcon.2013.10.002.
[25] C. Fang, L. Wang, An effective shuffled frog-leaping algorithm for resource-constrained project scheduling problem, Comput. Oper. Res. 39 (5) (2012) 890–901, https://doi.org/10.1016/j.cor.2011.07.010.
[26] S. Elsayed, R. Sarker, T. Ray, C.C. Coello, Consolidated optimization algorithm for resource-constrained project scheduling problems, Inf. Sci. 418 (2017) 346–362, https://doi.org/10.1016/j.ins.2017.08.023.
[27] H. Zhang, X. Li, H. Li, F. Huang, Particle swarm optimization-based schemes for resource-constrained project scheduling, Autom. Constr. 14 (3) (2005) 393–404, https://doi.org/10.1016/j.autcon.2004.08.006.
[28] Q. Jia, Y. Seo, An improved particle swarm optimization for the resource-constrained project scheduling problem, Int. J. Adv. Manuf. Technol. 67 (9–12) (2013) 2627–2638, https://doi.org/10.1007/s00170-012-4679-x.
[29] H. Zhang, H. Li, C. Tam, Particle swarm optimization for resource-constrained project scheduling, Int. J. Proj. Manag. 24 (1) (2006) 83–92, https://doi.org/10.1016/j.ijproman.2005.06.006.
[30] G. Koulinas, L. Kotsikas, K. Anagnostopoulos, A particle swarm optimization based hyper-heuristic algorithm for the classic resource constrained project scheduling problem, Inf. Sci. 277 (2014) 680–693, https://doi.org/10.1016/j.ins.2014.02.155.
[31] K. Ziarati, R. Akbari, V. Zeighami, On the performance of bee algorithms for resource-constrained project scheduling problem, Appl. Soft Comput. 11 (4) (2011) 3720–3733, https://doi.org/10.1016/j.asoc.2011.02.002.
[32] W. Chen, Y. Shi, H. Teng, X. Lan, L. Hu, An efficient hybrid algorithm for resource-constrained project scheduling, Inf. Sci. 180 (6) (2010) 1031–1039, https://doi.org/10.1016/j.ins.2009.11.044.
[33] R. Klein, Project scheduling with time-varying resource constraints, Int. J. Prod. Res. 38 (16) (2000) 3937–3952, https://doi.org/10.1080/00207540050176094.
[34] M. Palpant, C. Artigues, P. Michelon, LSSPER: solving the resource-constrained project scheduling problem with large neighbourhood search, Ann. Oper. Res. 131 (1–4) (2004) 237–257, https://doi.org/10.1023/B:ANOR.0000039521.26237.62.
[35] K. Bouleimen, H. Lecocq, A new efficient simulated annealing algorithm for the resource-constrained project scheduling problem and its multiple mode version, Eur. J. Oper. Res. 149 (2) (2003) 268–281, https://doi.org/10.1016/S0377-2217(02)00761-0.
[36] L. Wang, C. Fang, A hybrid estimation of distribution algorithm for solving the resource-constrained project scheduling problem, Expert Syst. Appl. 39 (3) (2012) 2451–2460, https://doi.org/10.1016/j.eswa.2011.08.095.
[37] H.F. Rahman, R. Sarker, D. Essam, A real-time order acceptance and scheduling approach for permutation flow shop problems, Eur. J. Oper. Res. 247 (2) (2015) 488–503, https://doi.org/10.1016/j.ejor.2015.06.018 12/01/2015.
[38] H.F. Rahman, R. Sarker, D. Essam, A genetic algorithm for permutation flow shop scheduling under make to stock production system, Comput. Ind. Eng. 90 (Supplement C) (2015) 12–24, https://doi.org/10.1016/j.cie.2015.08.006.
[39] H.F. Rahman, I. Nielsen, Scheduling automated transport vehicles for material distribution systems, Appl. Soft Comput. 82 (2019) 105552, , https://doi.org/10.1016/j.asoc.2019.105552.
[40] R. Kolisch, Efficient priority rules for the resource-constrained project scheduling problem, J. Oper. Manag. 14 (3) (1996) 179–192, https://doi.org/10.1016/0272-6963(95)00032-1.
[41] S. Ponnambalam, P. Aravindan, G.M. Naidu, A multi-objective genetic algorithm for solving assembly line balancing problem, Int. J. Adv. Manuf. Technol. 16 (5) (2000) 341–352, https://doi.org/10.1007/s001700050166.
[42] R. Ruiz, C. Maroto, A comprehensive review and evaluation of permutation flowshop heuristics, Eur. J. Oper. Res. 165 (2) (2005) 479–494, https://doi.org/10.1016/j.ejor.2004.04.017.
[43] D.E. Goldberg, J.H. Holland, Genetic algorithms and machine learning, Mach. Learn. 3 (2) (1988) 95–99, https://doi.org/10.1023/A:1022602019183.
[44] R. Ruiz, C. Maroto, J. Alcaraz, Two new robust genetic algorithms for the flowshop scheduling problem, Omega 34 (5) (2006) 461–476, https://doi.org/10.1016/j.omega.2004.12.006.
[45] H.F. Rahman, R.A. Sarker, D.L. Essam, A memetic algorithm for permutation flow shop problems," in IEEE Congress on Evolutionary Computation (CEC), IEEE (2013) 1618–1625, https://doi.org/10.1109/CEC.2013.6557755.
[46] H.F. Rahman, R.A. Sarker, D.L. Essam, G. Chang, A memetic algorithm for solving permutation flow shop problems with known and unknown machine breakdowns," in IEEE Congress on Evolutionary Computation (CEC), IEEE (2014) 42–49, https://doi.org/10.1109/CEC.2014.6900242.
[47] Q.-K. Pan, R. Ruiz, An estimation of distribution algorithm for lot-streaming flow shop problems with setup times, Omega 40 (2) (2012) 166–180, https://doi.org/10.1016/j.omega.2011.05.002.
[48] H.F. Rahman, R. Sarker, D. Essam, Multiple-order permutation flow shop scheduling under process interruptions, Int. J. Adv. Manuf. Technol. 97 (5–8) (2018) 2781–2808, https://doi.org/10.1007/s00170-018-2146-z.
[49] M. Vanhoucke, J. Coelho, A tool to test and validate algorithms for the resource-constrained project scheduling problem, Comput. Ind. Eng. 118 (2018) 251–265 2018/04/01/ https://doi.org/10.1016/j.cie.2018.02.001.
[50] R. Chakrabortty, R. Sarker, D. Essam, Single mode resource constrained project scheduling with unreliable resources, Oper. Res. (2018) 1–35, https://doi.org/10.1007/s12351-018-0380-7.
[51] D. Debels, B. De Reyck, R. Leus, M. Vanhoucke, A hybrid scatter search/electromagnetism meta-heuristic for project scheduling, Eur. J. Oper. Res. 169 (2) (2006) 638–653, https://doi.org/10.1016/j.ejor.2004.08.020.
[52] A. Fahmy, T.M. Hassan, H. Bassioni, Improving RCPSP solutions quality with stacking justification–application with particle swarm optimization, Expert Syst. Appl. 41 (13) (2014) 5870–5881, https://doi.org/10.1016/j.eswa.2014.03.027.
[53] D.C. Paraskevopoulos, C.D. Tarantilis, G. Ioannou, Solving project scheduling problems with resource constraints via an event list-based evolutionary algorithm, Expert Syst. Appl. 39 (4) (2012) 3983–3994, https://doi.org/10.1016/j.eswa.2011.09.062.
[54] V. Valls, F. Ballestín, S. Quintanilla, A population-based approach to the resource-constrained project scheduling problem, Ann. Oper. Res. 131 (1–4) (2004) 305–324, https://doi.org/10.1023/B:ANOR.0000039524.09792.c9.
[55] F. Berthaut, R. Pellerin, A. Hajji, N. Perrier, A path relinking-based scatter search for the resource-constrained project scheduling problem, Int. J. Proj. Organ. Manag. Sci. 10 (1) (2018) 1–36, https://doi.org/10.1504/IJPOM.2018.090372.
[56] S. Proon, M. Jin, A genetic algorithm with neighborhood search for the resource-constrained project scheduling problem, Nav. Res. Logist. 58 (2) (2011) 73–82, https://doi.org/10.1002/nav.20439.
[57] D.W. Zimmerman, B.D. Zumbo, Relative power of the Wilcoxon test, the Friedman test, and repeated-measures ANOVA on ranks, J. Exp. Educ. 62 (1) (1993) 75–86, https://doi.org/10.1080/00220973.1993.9943832.
[58] E.A. Gehan, A generalized Wilcoxon test for comparing arbitrarily singly-censored samples, Biometrika 52 (1–2) (1965) 203–224, https://doi.org/10.2307/2333825.
[59] E.D. Dolan, J.J. Moré, Benchmarking optimization software with performance profiles, Math. Program. 91 (2) (2002) 201–213, https://doi.org/10.1007/s101070100263.
[60] J. Batselier, M. Vanhoucke, Construction and evaluation framework for a real-life

project database, Int. J. Proj. Manag. 33 (3) (2015) 697–710, https://doi.org/10.1016/j.ijproman.2014.09.004.
[61] P.-H. Chen, H. Weng, A two-phase GA model for resource-constrained project scheduling, Autom. Constr. 18 (4) (2009) 485–498, https://doi.org/10.1016/j.autcon.2008.11.003.
[62] D.-H. Tran, M.-Y. Cheng, M.-T. Cao, Solving resource-constrained project scheduling problems using hybrid artificial bee colony with differential evolution, J. Comput. Civ. Eng. 30 (4) (2016), https://doi.org/10.1061/(ASCE)CP.1943-5487.0000544.
[63] D. Castro-Lacouture, G.A. Süer, J. Gonzalez-Joaqui, J.K. Yates, Construction project scheduling with time, cost, and material restrictions using fuzzy mathematical models and critical path method, J. Constr. Eng. Manag. 135 (10) (2009) 1096–1104, https://doi.org/10.1061/(ASCE)0733-9364(2009)135:10(1096.
[64] X. Hu, N. Cui, E. Demeulemeester, L. Bie, Incorporation of activity sensitivity measures into buffer management to manage project schedule risk, Eur. J. Oper. Res. 249 (2) (2016) 717–727, https://doi.org/10.1016/j.ejor.2015.08.066.