Evolutionary Intelligence (2023) 16:519–537
https://doi.org/10.1007/s12065-021-00675-x

RESEARCH PAPER

A hybrid metaheuristic method for solving resource constrained project scheduling problem

Ohiduzzaman Shuvo[1] · Swajan Golder[1] · Md. Rafiqul Islam[1]

Received: 14 July 2020 / Revised: 3 August 2021 / Accepted: 10 October 2021 / Published online: 3 November 2021

Abstract
Resource constrained project scheduling problem (RCPSP) is a renowned variant of the scheduling problem. RCPSP is very important in production and management but computationally hard. It is widely used in many fields like job shop scheduling, flow shop scheduling, transactional planning, wireless communication etc. The objective of solving RCPSP is to obtain minimum makespan maintaining all constraints. There are some exact, approximate, heuristic and metaheuristic algorithms which were proposed to solve this problem. RCPSP is an NP-hard problem. Chemical reaction optimization (CRO) is a population based metaheuristic method to solve such problems and it shows better performance comparing with some other existing algorithms. CRO explores the large search space both locally and globally using its four operators. Genetic algorithm (GA) is also a nature inspired algorithm which is used to solve various optimization problems. In this paper, we are proposing a hybrid metaheuristic approach that integrates chemical reaction optimization (CRO) and genetic algorithm (GA) named CRO-GA to solve RCPSP. We have redesigned the basic operators of CRO and GA to find out the solutions. An additional operator called priority based selection operator is used in CRO to adjust with GA. Our proposed method is compared with other related approaches such as adaptive particle swarm optimization (A-PSO), multi agent optimization algorithm (MAOA), artificial bee colony (ABC), genetic algorithm (GA) which are state of the art for the RCPSP. The experimental results show that our proposed methodology gives better results than other existing algorithms to solve RCPSP with less computational time.

Keywords Chemical reaction optimization · NP-hard · Resource constrained project scheduling problem · Makespan

1 Introduction

Scheduling has an intrinsic impact on the production or manufacturing process and it is also requisite in optimization engineering. Scheduling is the mechanism of organizing, governing, and optimizing work and workloads of a project. Ordinarily, a project gets diverse forms in various business, production, management, and engineering. But every project is comprised of numerous tasks and each task is fixed with a given start and end time. Each task also requires one or more resources to perform its execution. As resources and time are not perpetual, projects demand tight scheduling for diminishing project continuation. With the tremendous revolution in science and technology, industries and manufacturing companies are initiating immense complex projects which tend to succeed if they are drawn on an optimized scheduled way. In the late 1950's, many project organizing procedures like PERT (Program Evaluation and Review Technique) and CPM (Critical Path Method) helped to perceive minimum duration schedules for projects where different resources required for project fulfillment were assumed not to be a limiting factor [9]. In the practical field, projects require various resources but resources have bounded availability which directly dominates project objectives, time optimization, and profit gaining. Consequently, project scheduling with tight resource allocation has become the focus of many recent researchers.

A classical resource constrained project scheduling problem (RCPSP) involves some activities with precedence

✉ Ohiduzzaman Shuvo
oheduzzamanshuvo@gmail.com

Swajan Golder
swajan.cse.ku@gmail.com

Md. Rafiqul Islam
dmri1978@gmail.com

[1] Computer Science and Engineering Discipline, Khulna University, Khulna 9208, Bangladesh

relationship and some limited renewable resources. In RCPSP, our main challenge is to obtain the activity sequence with resource allocation such that the project can be done in the shortest possible time considering precedence and resource constraints. More formally, we can define the RCPSP as follows:

1. A RCPSP is comprised of a set of n activities, $A\{a_0, a_1, a_2, \dots, a_{n+1}\}$.
2. A set P which represents the zero lag, finish to start relationship of set A.
3. A set R which narrates the renewable resources and the constraint of resource availability is Q_r, $\forall r \in R$.
4. Each activity 'a' has a set of:
   (a) Direct predecessors = $p_a \subseteq A$
   (b) Indirect predecessors = $p'_a \subseteq A$
   (c) Direct successors = $s_a \subseteq A$
   (d) Indirect successors = $s'_a \subseteq A$.
5. Each activity 'a' has a constant duration of time, d_a and a constant resource allocation.
6. An activity requires q_{ar} units of r types of resources during each time period in its total duration.
7. The activities a_0 and a_{n+1} denote the starting and ending of the project which are dummy activities.

The dummy start and end activities have duration $d_0 = d_{n+1} = 0$ and resource unit $q_{0r} = q_{(n+1)r} = 0$. The values of d_a and q_{ar} are assumed non negative. The precedence relationships between these activities are two types. One is precedence constraint that prevents an activity to start when its parent activities are yet to finish. Another one is the sum of all required resources at any time period could not exceed the given amount of resources.An example of RCPSP with seven activities is given in Fig. 1. One resource type and three available resource units are used here. The feasible schedule of this example with a makespan of 8 time periods is given in Fig. 2 where X-axis and Y-axis represent the time period and the resource instances respectively. According to the complexity theory, if decision version of an optimization problem is NP-complete, the problem will be called NP-hard. The decision version of RCPSP is NP-complete. NP-hardness of RCPSP has been presented by Blazewicz [4]. In the past, different types of scheduling algorithms were introduced to solve RCPSP, but they had some disadvantages that they failed to provide effective results for large instances (more than 60 instances) in a reasonable computational time. Researchers proposed a number of heuristic and metaheuristic algorithms to solve RCPSP. The majority of heuristic algorithms can be categorized into two types: first one priority rule based heuristics and second one neighborhood based heuristics. To obtain a feasible schedule, various heuristic algorithms used neighborhood search methods. Researchers developed many metaheuristic algorithms like Simulated annealing Boctor [6] Bouleimen and Lecocq [7], Tabu search Baar et al. [2] Nonobe and Ibaraki [25], Ant colony optimization (ACO) Merkle et al. [23] Herbots et al. [12], Artificial bee colony Akbari et al. [1], Particle swarm optimization (PSO) algorithm [31] Zhang et al. [30] Jarboui et al. [15] for solving RCPSP. There are two scheduling schemes like serial and parallel are mostly applied in most heuristic techniques published in the literature Kelley [18] Brooks [8] Boctor [5] Kolisch and Sprecher [19] Drexl and Gruenewald [10]. Recently, an adaptive PSO (A-PSO) algorithm is proposed by K. Neetesh and D. Prakash to solve RCPSP which is simple and also effective in producing the better results than previous works Kumar and Vidyarthi [20]. An adaptive inertia weight tuning process is used for the noticeable convergence of the PSO. All of the above mentioned algorithms can produce better results in some restricted inputs criteria but all of them have some limitations for large instances.

Chemical reaction optimization and genetic algorithm are two population based metaheuristic algorithms and they were used to solve many optimization problems. By

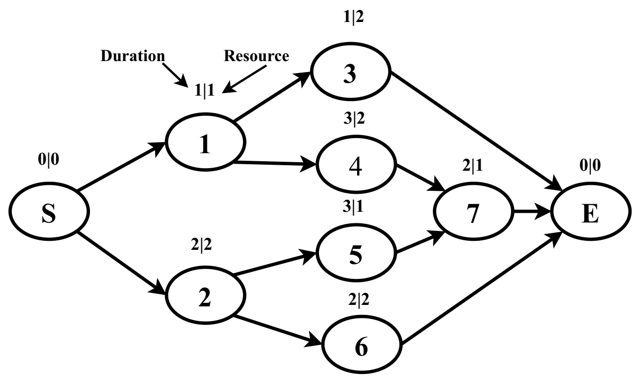


Fig. 1 A RCPSP example

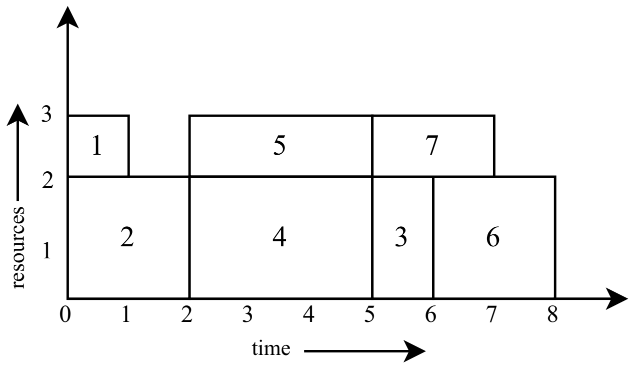


Fig. 2 A feasible schedule of the RCPSP example

exploring the efficiencies and effectiveness of CRO and GA, this paper proposes a new hybrid metaheuristic method named CRO-GA for solving the resource constrained project scheduling problem. A new priority selection operator has been introduced in this method and embedded with CRO. The originality and contribution of the proposed work are as follows:

1. We have redesigned four basic operators such as on-wall ineffective collision, decomposition, intermolecular ineffective collision, synthesis of CRO to find the job sequence which ensures minimum project duration.
2. A priority based selection operator has been proposed here for CRO.
3. Three evolutionary operators of GA such as selection, crossover and mutation have been redesigned for RCPSP.
4. The combination of CRO with GA to solve RCPSP is one of our other novel works.
5. The proposed CRO-GA model follows both serial and parallel scheduling for estimating project duration. This model is easy to implement and takes minimum time to find the expected job sequence. The outcomes of the proposed model are compared with the previous related works such as A-PSO Kumar and Vidyarthi [20], MAOA Zheng and Wang [32], ABC Jia and Seo [16], GA Montoya-Torres et al. [24] to show the performance of our work.
6. We have done statistical tests to show the significance of our proposed method.

The paper is organized as follows. The motivation is presented in section 2 after the introduction. The literature review is given in section 3, whereas different related algorithms are explained. Section 4 represents the proposed CRO-GA hybrid model. The information about the dataset and how we have used it in our system have been depicted in section 5. In section 6, the experimental results are given, whereas the performances of the proposed method are compared with other related algorithms. The conclusions are depicted in section 7.

2 Motivation

RCPSP has several important application areas. It is widely used in large projects which consist of a huge number of jobs. Production and management largely depend on intelligent scheduling which is controlled by RCPSP. Besides, RCPSP has become more popular in the field of computing, wireless communication, operations research and transport planning. Many algorithms are available to solve RCPSP. But in case of large instances they showed poor performance in solving RCPSP. Our target is to design an algorithm which solves this problem efficiently and effectively than the others.

Chemical reaction optimization is an efficient population based metaheuristic algorithm. The CRO algorithm represents the molecular interaction during a chemical reaction Mahmud et al. [22]. It is an optimization technique inspired by the nature of the chemical reaction. According to the studies by a number of researchers, the CRO algorithm has proved its improved performance than the other metaheuristic approaches by solving various problems with efficiencies Saifullah and Islam [26]. CRO has been used to solve the 0–1 knapsack problem Truong et al. [27], task scheduling and grid computing problem Xu et al. [29], quadratic assignment problem Xu et al. [28], shortest common super sequence problem Saifullah and Islam [26], longest common subsequence problem for multiple string Islam et al. [14], the max flow problem Barham et al. [3], RNA structure prediction Kabir and Islam [17], generalized vertex covering Islam et al. [13] etc. with superior results than the other exact, heuristic and metaheuristic algorithms.

Genetic Algorithm is based on the principal of genetics and natural selection. It is frequently used to find optimal or near optimal solutions of difficult problems which otherwise would take extensive time to solve. So, we have proposed an algorithm based on CRO by redesigning four basic operators and the priority based selection as an additional operator and combined with GA to make efficient convergence for solving RCPSP problem than the other existing algorithms.

3 Related works

Several approaches were proposed by researchers for solving RCPSP. Only some of these approaches had shown efficiency in a huge number of activities. This section gives a literature review for solving RCPSP while focusing on heuristic and metaheuristic techniques such as genetic algorithm (GA), tabu search (TS), simulated annealing (SA), ant colony optimization (ACO), and artificial bee colony (ABC).

3.1 Simulated annealing (SA)

Simulated annealing (SA) is a searching approach in the large solution space for finding the comparatively global optimum for a given operation. Boctor proposed a new method for solving resource constrained project scheduling problems using simulated annealing algorithm in which resources are limited, renewable and non-preemptive Boctor [6]. This algorithm solves both single mode and multi mode problems and satisfies different optimizing objective functions. Any scheduling heuristic is manipulated to generate the initial solution in this algorithm. This adaptation has some special characteristics; it can be detailed as an annealing procedure with reheating and variable cooling rate. The procedure is composed of a number of heating cycles and at the beginning of each of them the cooling temperature T is reinitialized and set to its initial value TMAX. The proposed way shows better performance compared with some tabu search heuristics. The main drawback of the designed work was to deal with a large amount of dataset which produced unexpected results.

Single mode and multiple mode version of resource constrained project scheduling problem were solved by Bouleimen and Lecocq using a simulated annealing (SA) algorithm Bouleimen and Lecocq [7]. The objective function is the minimization of the makespan. A new design replaces the conventional SA search scheme that takes into account the specificity of the solution space of scheduling problems. For RCPSP, all parameters were set after preliminary statistical experiments completed on test instances and the search was dependent on an alternated activity and time incrementing process. For MRCPSP, they used an original approach which consisted of two embedded search loops. The algorithm proved the effectiveness of both adaptation by experimenting benchmark instances available in the literature among the recent published methods.

3.2 Ant colony optimization (ACO)

The general idea of ACO metaheuristic is to avail the ant algorithm for finding the best solution. Merkle used ant colony optimization algorithm for solving non-preemptive resource constrained project scheduling problems Merkle et al. [23]. They proposed a combination of summation evaluation and direct method for solving RCPSP. Within this approach, the initial solution is generated by using serial or parallel scheduling algorithm. In the case of the summation evaluation method, it is important to schedule a job not too late. But at the same time, a group of activities should be scheduled which satisfies the resource requirements. In the activity list, there might be some places from which the activity list used by a serial or parallel schedule that is good while other places in the activity list might be worse. This kind of behavior can be modeled using the direct evaluation method. For handling this behavior they proposed a combination of direct and summation evaluation method. Summation evaluation strategy enforces a job to be scheduled according to its tardiness value.

The main parameters of an ACO algorithm are α, β and ρ. During the whole run of the algorithm, these parameters are assigned to fixed values. In the previous work, no variation in the parameters value was considered but in the proposed work they considered a variation in the values of β and ρ. Here, β controls the contingent influence of the heuristic values and parameter ρ determines the convergence speed of the algorithm.

They tested the ACO algorithm on a set of large benchmark problems from the project scheduling library Merkle et al. [23]. From this library, they used the test set j120.sm which contains the largest problem instances in the PSPLIB. They compared their results with the results of other heuristics for the RCPSP including genetic algorithm (GA), simulated annealing (SA), tabu search and other sampling algorithms but this algorithm performed best on average. The fact that the algorithm behaves very well (compared to several other heuristics) in both cases with and without restrictions to the number of evaluated schedules and shows the flexibility of the approach.

3.3 Multi agent optimization algorithm (MAOA)

There are mainly three types of characteristics in a multi agent system and these are environments, behaviors of agents and interaction among agents. An agent holds three types of behavior like autonomous, social and self learning behavior. A multi agent based method was developed by Xiao and Wang to solve RCPSP Zheng and Wang [32]. In this paper, a group structure of N groups with S agents is employed where the best agent of a group is selected as a group leader. In the MAOA, solutions are represented by agents. The relationship between individual agents are described by an organized architecture. In the MAOA, an environment is constructed by dividing agents into several groups. Then the global exploration is performed through the social behavior of an agent and the local searching is performed through the autonomous and self learning behavior of an agent. In this algorithm, agents are moved among the groups to adjust the environment and information are shared using agent based search behaviors for further well evolution.

For solving RCPSP, a particular activity list called extended activity list is used to represent an agent, which is comprised of three lists. First one is an activity list $A\{a_0, a_1, a_2, \dots, a_{n+1}\}$, the second one is the start time of each activity and the last is the end time of each activity. For finding a better initial agent in MAOA, a regret based biased random sample method with LFT priority rule is used to initialize each activity. To select an eligible activity a selection strategy based on roulette wheel with LFT priority rule is used. In this paper, they designed a new operator named magnet based crossover to solve RCPSP. The magnet based crossover operator (MBCO) can provide offspring which inherit some characterstics from both parents. The MBCO is used to obtain a better exploration. After obtaining a better offspring, the worst agent in the population is replaced by the new offspring. The resource based crossover operator (RBCO) is used to inherit an activity which utilizes resources Zheng and Wang [32]. The RBCO can improve the local behavior of an agent. The agents and the leader of a group are modified by performing RBCO.

To compare the results of this algorithm with other existing algorithms, they used the three well known datasets from PSPLIB Kolisch and Sprecher [19]. The benchmark dataset contains 480 J30 instances, 460 J60 instances and 600 J120 instances. In this work, they used average percentage deviation from lower bound for comparison.

3.4 Genetic algorithm (GA)

A genetic algorithm is usually used to produce a standard result by applying different operators. A genetic algorithm approach for resource constrained project scheduling problem was proposed by Montoya [24]. The algorithm was evaluated with benchmark instances available at the PSPLIB website. In terms of computational time and effective solution quality, their solution procedure is enough efficient. In this paper, for the implementation of the genetic algorithm they developed an object oriented conceptual model. In OOP implementation, they represented each chromosome as an object having its own features. There are two types of schedule generation scheme in the case for solving RCPSP. In this paper, they used a series schedule generation scheme. In this work for implementing the operators of GA, they also used two types of crossovers: one point crossover and two point crossover. The evolution strategy is implemented in this algorithm using the basis of elitist list. The strategy is consisted of selection of the best individual and utilization of it to produce a number of important individuals for the next generation. The population size was fixed to 150. This OOP model showed an efficient convergency.

4 Proposed CRO-GA hybrid model

Chemical reaction optimization (CRO) is inspired by the nature of chemical reactions and a hugely used population based metaheuristic for optimization. It was proposed by Albert Lam and Li [21]. It is an approach that couples chemical reactions with an optimization technique. There are several essential attributes of each molecule that takes part in the basic operation of CRO. These are molecular structure (ω), potential energy (PE) and kinetic energy (KE). Molecular structure (ω) captures a solution of the problem. There are four elementary reactions of CRO which are on-wall ineffective collision, decomposition, synthesis and intermolecular ineffective collision. A genetic algorithm is a search based heuristic which is inspired from the biological process. The fundamental concept of the GA design is based on the principles posed by Garg to find the survival of the fittest Garg [11]. GA was inaugurated as a computational metaphor of adaptive systems. GA is modeled by combining the principles of the evolution through natural selection, employing a population of individuals that undergo selection in the presence of variability inducing operators such as mutation and recombination (crossover). For the evaluation of individuals, a fitness function is used and reproductive success depends on fitness value.

In the proposed CRO-GA based hybrid model, CRO uses or shares GA′s selection operator. Ordinarily, CRO chooses molecules randomly from the population for executing any of the four reactions. There remains a possibility of not adopting good molecules, which may not produce better molecules comparing with reactant molecules. We have combined the theory of thermodynamics and evolution in our hybrid CRO-GA model. We have designed a priority based selection operator for CRO which selects supreme molecules for a collision.

In the proposed model, CRO searches for the optimal solution both locally and globally by adopting the priority based selection operator and operating its basic four operators in the solution space at first then it will transfer its solution space to GA for a better outcome. GA will use all CRO modified molecules as chromosomes to form its own population. Then GA executes its operation using native selection, crossover and mutation operators. There exists both good and cheap mean random quality molecules in the initial CRO population. The priority based selection operator is designed based on priority rule. Hence, favorable molecules get further opportunity to react. As poor molecules are discarded during each step of CRO execution, the final solution space of CRO becomes many times better than the primary population. If CRO cannot ensure optimal or near optimal solutions even in the final stage, GA will try. GA uses CRO's final solution space but no priority selection is needed as the solution space is already good enough. GA uses random selection operator.

In the proposed model, CRO selects prioritized molecules from the population where GA′s selection emphasizes random molecules which ensures more robustness than single CRO or GA as there is a combination of randomness and priority. Sometimes CRO gets stuck at local minima, in those cases GA assists to escape it and reach to global minima. If CRO perceives expected solution, GA will never be triggered and the execution will be terminated. Two termination conditions have been used in the proposed CRO-GA model: first the specified number of iterations passed; second expected solution is found. For GA we have always used 20 generations and it also follows the aforementioned termination conditions. The flowchart of our proposed CRO-GA hybrid model is shown in Fig. 3.

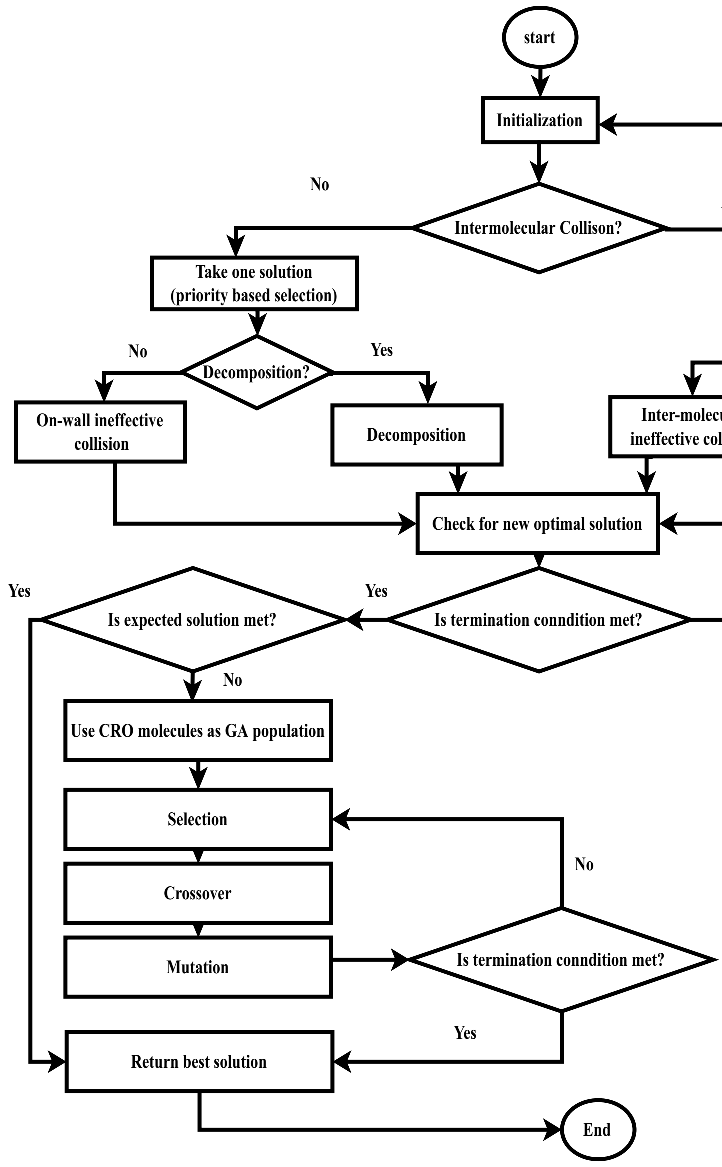

start
Initialization
Intermolecular Collison?
No
Yes
Take one solution (priority based selection)
Take two solutions (priority based selection)
Decomposition?
No
Yes
Synthesis?
No
Yes
On-wall ineffective collision
Decomposition
Inter-molecular ineffective collision
Synthesis
Check for new optimal solution
Is termination conndition met?
No
Yes
Is expected solution met?
Yes
No
Use CRO molecules as GA population
Selection
Crossover
Mutation
Is termination conndition met?
No
Yes
Return best solution
End

Fig. 3 Flowchart of CRO-GA

4.1 Population generation

The population is generated and the values of different parameters such as PopSize, MoleColl, buffer, initial KE, KELossRate, thresholds (α, β) are initialized in the initialization phase. The population is made up with a number of solutions or molecules. Every solution is a sequence of all activities which satisfies the precedence relation. An activity is said to be eligible, if its all predecessors have already been added to the solution sequence. A list S will be a partial feasible solution of our problem, if we add some activities randomly from the eligible set. When an activity is added to S, the eligible set and S are updated. By iteratively doing this, when the eligible set is empty, we get one feasible solution. The pseudo-code for our proposed population generation is shown in Algorithm 1.

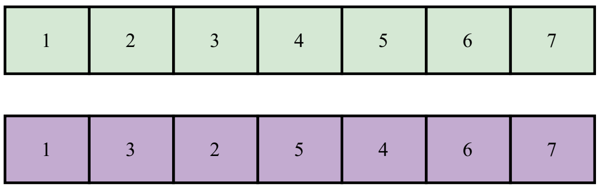


Fig. 4 Population generation

Algorithm 1: Population Generation

```
Input: A set of activities J with precedence relationship.
Output: A sequence, S of activities following the precedence
        relationship.
1 S = {}, be a feasible solution
2 Compute eligible set of activities, E from S, J
3 while $E \neq \phi$ do
4     Add one activity j from E to S randomly
5     Update E
6 end
7 return S
```

By applying the aforementioned population generation algorithm to the example in Fig. 1 for two times, two feasible solutions are generated shown in Fig. 4.

4.2 Operators used in CRO

In this section we have designed some operators which are used as primary operators of CRO. The operators are described below.

4.2.1 Priority based selection

We have designed a new selection operator named priority based selection operator. Priority based selection has been used for molecule selection. A molecule with lower fitness value (makespan value) has a higher probability to select for CRO operation. Priority calculation of CRO follows Eq. 1.

$$priority = \frac{1}{fitness} \tag{1}$$

This operator helps the method to select molecules with lower fitness (makespan) value. Since the objective of this method is to find the minimum makespan, this selection increases the efficiency of this method. The whole population is sorted according to priority in descending order. After sorting the population in descending order, the molecules with the minimum fitness values come to the front of the sorted population list that leads to select molecules randomly from the first half of the sorted population for CRO's elementary reactions.

4.2.2 On-wall ineffective collision

This elementary reaction is used for local search in CRO. We randomly pick a molecule (solution) from the population and select one random position, i from the selected solution S. Then the elements of i and $i+1$ positions are swapped which create a new molecule S'. If this breaks the precedence relation, we have to abandon this operation and choose a new position i and do the same thing. If the new molecule's potential energy (PE) is less than the original, then the original molecule S is replaced by the new one S'. This operation can be written as $S \rightarrow S'$. The on-wall ineffective collision for a feasible molecule of RCPSP is shown in Fig. 5. The pseudo-code of on-wall ineffective collision is given in Algorithm 2.

Algorithm 2: On-wall ineffective collision

```
Input: Molecule $S$.
Output: Molecule $S'$.
1  Copy the values from $S$ to $S'$
2  while True do
3      Get $pos_1$ randomly from 1, 2, 3,..., m-1
4      $pos_2 \leftarrow pos_1+1$
5      if job in $pos_1$ is not predecessor of job in $pos_2$ then
6          break
7      end
8  end
9  $S'[pos_1] \leftarrow S[pos_2]$
10 $S'[pos_2] \leftarrow S[pos_1]$
11 return $S'$
```

4.2.3 Decomposition

For exploring another area of the search space decomposition is used. A newly generated molecule after performing decomposition gets an enormous change in its molecular structure. The output of this operator is two molecules S_1 and S_2. Molecule S_1 gets the sequence of the first half of its original molecule S and molecule S_2 gets the sequence of the second half of its original molecule S. The rest parts of the solutions are shuffled according to their own patents following the precedence relationship. The operation of this operator can be represented as $S \rightarrow S_1 + S_2$.

A solution with the minimum PE is kept in the population. Figure 6 represents the decomposition reaction.

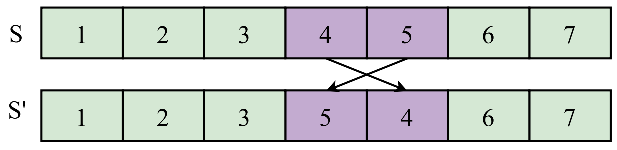


Fig. 5 On-wall ineffective collision

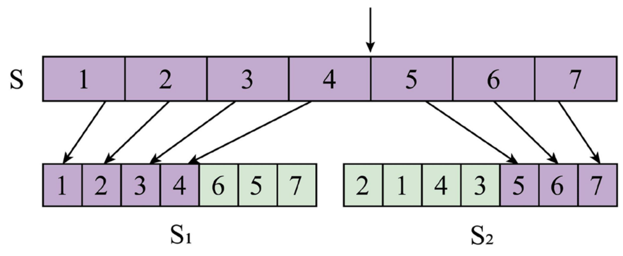


Fig. 6 Decomposition

The pseudo-code of decomposition is shown in Algorithm 3.

Algorithm 3: Decomposition

```
Input: Molecule $S$.
Output: Molecule $S_1$ and $S_2$.
1 Copy the values from $S$ to $S_1$ and $S_2$.
2 Take $pos$ randomly from 1, 2, 3,..., m
3 for $i \leftarrow 0$ to $pos$ do
4     Shuffle jobs of $S_2$ by following precedence relationship
5 end
6 for $i \leftarrow pos$ to $m$ do
7     Shuffle jobs of $S_1$ by following precedence relationship
8 end
9 return $S_1$ and $S_2$
```

4.2.4 Inter molecular ineffective collision

This reaction affects the solution structure but the number of solutions remains unchanged. Two solutions, say S_1 and S_2 are taken randomly and two random points are also taken within the solution length. As a result, both the solutions are divided into three parts. The odd parts of S_1 are rearranged following the solution's sequence of S_2 and the even part remains unchanged to form new solution S'_1. The same thing is done in S_2 to form new solution S'_2. The reaction can be represented as $S_1 + S_2 \rightarrow S'_1 + S'_2$.

A solution with the minimum PE is kept in the population. Figure 7 shows this reaction for RCPSP. The pseudo-code of inter molecular ineffective collision is given in Algorithm 4.

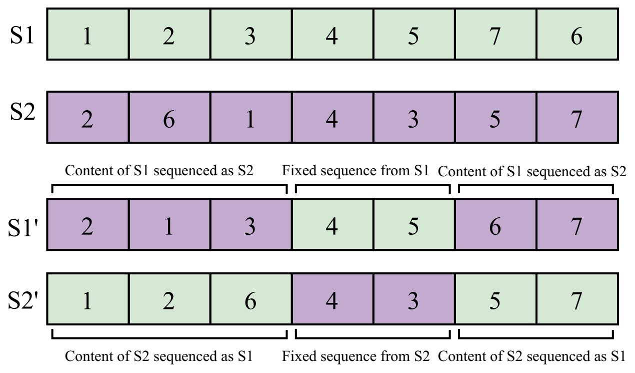

S1
1
2
3
4
5
7
6
S2
2
6
1
4
3
5
7
Content of S1 sequenced as S2
Fixed sequence from S1
Content of S1 sequenced as S2
S1'
2
1
3
4
5
6
7
S2'
1
2
6
4
3
5
7
Content of S2 sequenced as S1
Fixed sequence from S2
Content of S2 sequenced as S1

Fig. 7 Inter molecular ineffective collision

Algorithm 4: Intermolecular ineffective collision

```
Input: Molecule $S_1$ and $S_2$.
Output: Molecule $S'_1$ and $S'_2$.
1 Copy the values from $S_1$ to $S'_1$ and $S_2$ to $S'_2$
2 Take $pos_1$ and $pos_2$ randomly from 1, 2, 3,..., m (where
    $pos_1 < pos_2$)
3 for $i \leftarrow 0$ to $pos_1$ do
4     Resequence jobs of $S'_1$ maintaining the job sequence of
        $S_2$
5     Resequence jobs of $S'_2$ maintaining the job sequence of
        $S_1$
6 end
7 for $i \leftarrow pos_2$ to $m$ do
8     Resequence jobs of $S'_1$ maintaining the job sequence of
        $S_2$
9     Resequence jobs of $S'_2$ maintaining the job sequence of
        $S_1$
10 end
11 return $S'_1$ and $S'_2$
```

4.2.5 Synthesis

Synthesis is reverse to decomposition. Here, two molecules collide with each other and produce one molecule. Let S_1 and S_2 are two molecules. After a collision, if S is fused molecule, then it can be represented as $S_1 + S_2 \rightarrow S$. Here, we randomly pick two solutions such as S_1, S_2 from the population. Firstly, a contiguous block of activities is selected from S_1 by randomly drawing two positions n_1, n_2. Secondly, it

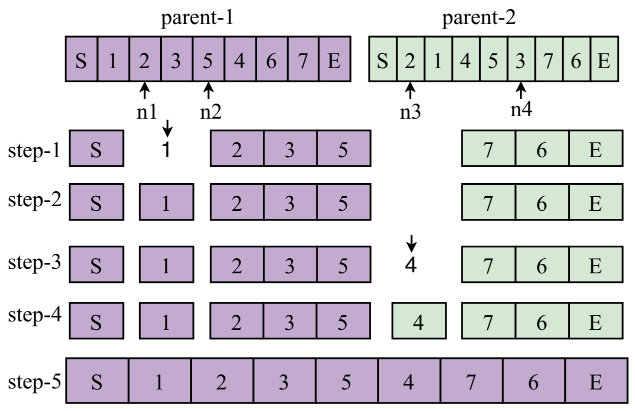


Fig. 8 Synthesis

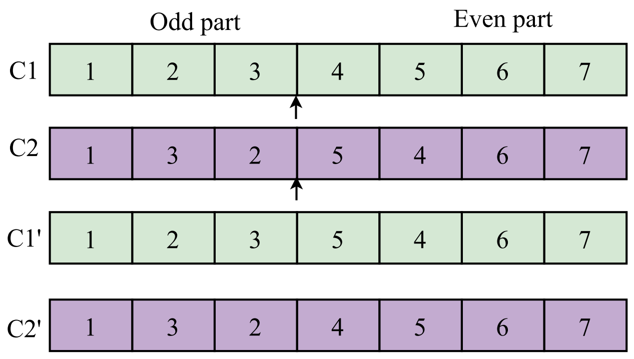


Fig. 9 Crossover

determines the smallest position of the contiguous block in S_2 denoted as n_3 and largest position of the block in solution S_2 denoted as n_4. The new solution S is then built by concatenating:

1. The activities in positions 1 to n_3 - 1 of S_2.
2. The activities in positions n_3 to n_4 of S_2 which are the predecessors to any activity of the contiguous block.
3. The activities of the contiguous block.
4. The activities in positions n_3 to n_4 of S_2 which are the are successors to any activity of the contiguous block.
5. The activities in positions n_4 + 1 to last of S_2.

The activities of n_3 to n_4 of S_2 which are not predecessors or successors of the block are called free activities. Free activities are added before the block or after the block randomly without violating any precedence relationship. The scenario is given in Fig. 8. The pseudo-code of synthesis is shown in Algorithm 5.

Algorithm 5: Synthesis

```
Input: Molecule $S_1$ and $S_2$.
Output: Molecule $S$.
Take $n_1$ and $n_2$ randomly from 1, 2, 3,..., m
Initialize $n_3$ and $n_4$ as 0
for $i \leftarrow 0$ to $m$ do
    if $S_2[i]$ exists in $S_1[n_1$ to $n_2]$ then
        $n_3 \leftarrow i$
        break
    end
end
for $i \leftarrow m$ to $0$ do
    if $S_2[i]$ exists in $S_1[n_1$ to $n_2]$ then
        $n_4 \leftarrow i$
        break
    end
end
first $\leftarrow S_2[0$ to $n_3]$
middle $\leftarrow S_1$ $[n_1$ to $n_2]$
last $\leftarrow S_2[n_4$ to m]
for $i \leftarrow n_3$ to $n_4$ do
    if $S_2[i]$ is predecessor of $S_1[n_1$ to $n_2]$ then
        Append $S_2$[i] to first
    end
    else if $S_2[i]$ is successor of $S_1[n_1$ to $n_2]$ then
        Append $S_2$[i] to middle
    end
    else
        Randomly append $S_2[i]$ to first or middle
    end
end
S $\leftarrow$ first + middle + last
return $S$
```

4.3 Operators used in GA

We have used three operators of GA to solve RCPSP. The operators are described below.

4.3.1 Selection

The random rule based selection has been used for chromosome selection. GA selects a chromosome randomly from its population.

4.3.2 Crossover

Crossover searches the solution space globally. Two chromosomes are selected from population randomly and one point crossover is performed on them. We take one random point within the chromosome length. As a result, both of the chromosomes are divided into two parts. The odd part of each chromosome remains unchanged and the even part of each chromosome is rearranged following the sequence of the other chromosome. We can represent the operation as $C_1 + C_2 \rightarrow C'_1 + C'_2$. Figure 9 shows this operation for RCPSP. The pseudo-code of crossover is given in Algorithm 6.

Algorithm 6: Crossover

```
Input: Chromosome $C_1$ and $C_2$.
Output: Chromosome $C'_1$ and $C'_2$.
1 Copy the values from $C_1$ to $C'_1$ and $C_2$ to $C'_2$
2 Take $pos$ randomly from 1, 2, 3,..., m
3 for $i \leftarrow pos_2$ to $m$ do
4     Resequence jobs of $C'_1$ maintaining the job sequence of $C_2$
5     Resequence jobs of $C'_2$ maintaining the job sequence of $C_1$
6 end
7 return $C'_1$ and $C'_2$
```

4.3.3 Mutation

Mutation performs a local search for GA. Randomly one chromosome is chosen and mutation is executed by following a threshold. A random number is taken from a uniform distribution from 0 to 1. If the number is greater than the threshold value, a mutation occurs otherwise not. We have used 0.4 as a threshold value. Mutation follows CRO′s on-wall ineffective operator remaining a slight difference. The first position is taken randomly but the second position is taken by adding a specific number with the first position. The scenario is depicted in Fig. 10. The pseudo-code of mutation is shown in Algorithm 7.

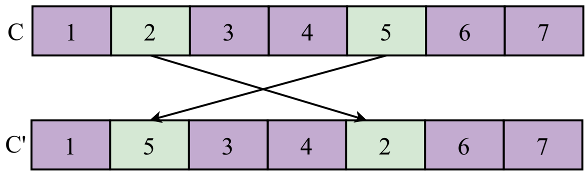


Fig. 10 Mutation

Algorithm 7: Mutation

```
Input: Chromosome $C$.
Output: Chromosome $C'$.
1 Copy the values from $C$ to $C'$
2 x ← 3
3 while True do
4     Get $pos_1$ randomly from 1, 2, 3,..., m-4
5     $pos_2 \leftarrow pos_1$ + x
6     if jobs in $pos_1$ and $pos_2$ of M do not contain any precedence relationship then
7         break
8     end
9 end
10 $C'[pos_1] \leftarrow C[pos_2]$
11 $C'[pos_2] \leftarrow C[pos_1]$
12 return $C'$
```

4.3.4 Parameter setting

We implemented the proposed CRO-GA in Python 3.6 and tested on a personal computer with Intel Core i5 (2.50 GHz) and 4 GB RAM under Windows 10 operating system (64 bit). Among the state of the art algorithms for RCPSP, MAOA showed better performence. MAOA was implemented using C++ Zheng and Wang [32]. For fair comparison, we also implemented MAOA in Python 3.6 and tested on the same environment. There are three key parameters in MAOA they are the number of group (N), the group size (S) and the acceptance probability (ρ). The best combination of parameters for MAOA is given in Table 1.

There are eight main parameters of CRO-GA. They are PopSize, Alpha, Iteration, KELossRate, MolColl, InitialKE, Beta, Mutation Threshold. For the experiment we set different values to these parameters. To achieve the best results, values of the parameters are tuned. Parameter tunings are showed in Fig. 11(a) to 11(d). The parameters used in the proposed CRO-GA are shown with their values in Table 2. From the Figure 11(a) to 11(d), we can observe that for α = 3, KElossRate = 0.4 and β = 0.6, we get best solutions.

Table 1 Parameters used in MAOA

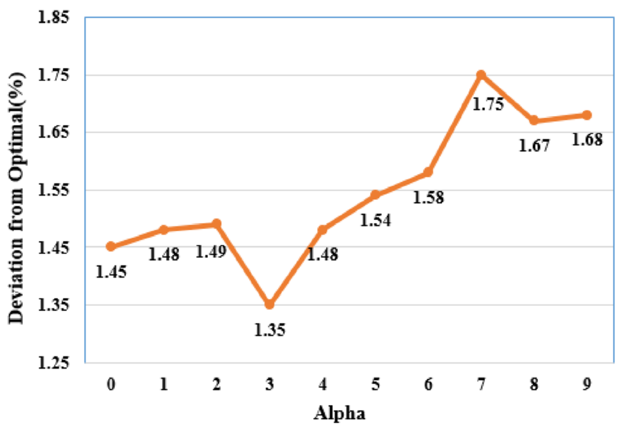
Dataset	N	S	ρ
J30	7	5	0.9
J60	5	10	0.9
J90	5	10	0.9
J120	5	10	0.9

Table 2 Parameters used in CRO-GA

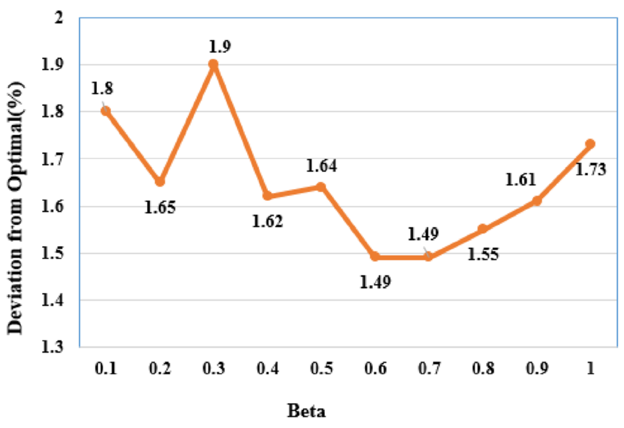
Symbol	Description	Value
PopSize	Population Size	100
Alpha	On-Wall and Decomposition Criteria	3
Iteration	No. of Execution	2000
KELossRate	Kinetic Energy Loss Rate	0.4
MolColl	Uni and Inter Molecular Criteria	0.6
InitialKE	Initial Kinetic Energy	0.51
Beta	Synthesis Criteria	0.6
Threshold	Mutation Criteria	0.4

5 Experimental dataset

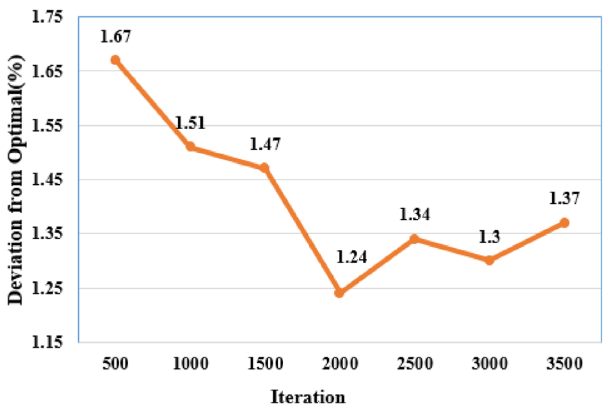
For analyzing the performance of our proposed CRO-GA algorithm, the project duration was determined using serial or parallel scheduling. We used the PSLIB datasets given in the literature Kumar and Vidyarthi [20]. This benchmark dataset contains four different sets named J30, J60, J90, and J120 with project of 30, 60, 90, and 120 activities respectively. Here, J30, J60, and J90 sets contain 480 instances for each of them but J120 set contains 600 instances. The optimal solutions are given only for J30 instances and the upper bounds and lower bounds are provided for J60, J90 and J120 in the literature and these can be found in psplib. In this experimental work, two lower bounds were considered one was best lower bound and another one was critical path method lower bound. A critical path method (CPM) lower bound was estimated by assuming endless resources (relaxation of resource constraints) and solving the simplified problem. To evaluate our proposed model we generated 1000, 5000 and 50000 schedules. The efficiency of the proposed model was measured by considering the value of the average deviation and the value of the number of optimal found. The project makespan has been computed by randomly choosing serial or parallel scheduling as the



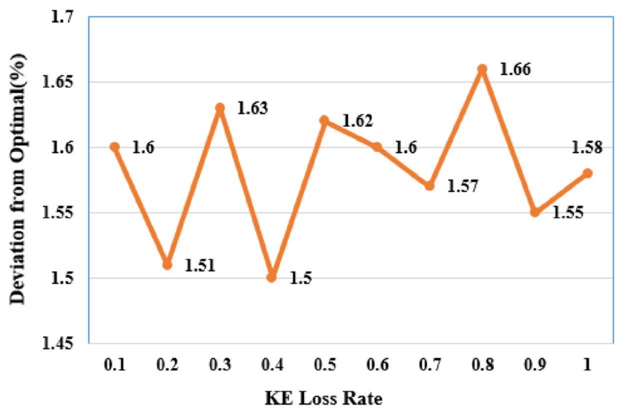
(a) Parameter tuning for alpha.



(b) Parameter tuning for beta.



(c) Parameter tuning for iteration



(d) Parameter tuning for KE Loss Rate.

Fig. 11 Parameter Tuning for some parameters

fitness function. We measured the average deviation of project duration for every dataset and also calculated the number of times optimal found.

The average deviation from optimal and average deviation from critical path method lower bound are represented by Avg.Dev.opt and Avg.Dev.Cpm.lb respectively. Eq. 2 is used to calculate Avg.Dev.opt and Avg.Dev.Cpm.lb.

$$Avg.Dev.opt \textbf{ or } Avg.Dev.Cpm.lb = \frac{1}{N}\sum_{i=1}^{N}\frac{x_i - x_i'}{x_i'} \times 100 \tag{2}$$

Here, x_i is the calculated makespan and x_i' is optimal or critical makespan for instance i.

The number of optimal solutions is calculated using the Eq. 3.

$$N_{optimal} = \frac{n}{N} \times 100 \tag{3}$$

Here, n is the number of optimal instances and N is the total number of instances.

It is clear that the smaller average deviation and greater number of optimal solutions indicate better algorithm.

6 Experimental results

First of all, we show the effect of genetic algorithm after combining it with chemical reaction optimization algorithm. The result of the proposed hybrid algorithm are compared with other existing state of art works where it can be observed that CRO-GA performs better than other metaheuristic algorithms in terms of average deviation, number of optimal solution and execution time.

6.1 Effect of GA in hybrid CRO-GA

To show the influence of GA in CRO-GA, we implemented GA, CRO and MAOA algorithms in our system and compared the performance with our proposed hybrid algorithm. The performance of the proposed model was evaluated using different numbers of schedules e.g., 1000, 5000 and 50000 schedules. We executed GA, CRO, MAOA and CRO-GA five times with the mentioned dataset and the best results are reported in Table 3. Besides this, for fair comparison MAOA is combined with GA and the results are shown in Table 3. Since the optimal solutions are known for J30 instances, there is no need of calculating critical path method (CPM) lower bound deviation for J30 and the corresponding cells of

Table 3 Comparison between CRO-GA and other methaheuristics

Dataset	Algorithm	Avg.Dev.opt(%) Or Avg. Dev.lb(%)			Avg.Dev.Cpm.lb(%)			No of Optimal Found(%)		
		1000	5000	50000	1000	5000	50000	1000	5000	50000
J30	GA	0.18	0.071	0.02	N/A	N/A	N/A	88.95	96.88	99.38
	CRO	0.169	0.0693	0.02	N/A	N/A	N/A	89.16	97.08	99.58
	MAOA	0.17	0.06	0.01	N/A	N/A	N/A	89.87	97.13	99.58
	MAOA-GA	0.17	0.058	0.01	N/A	N/A	N/A	89.87	97.29	99.58
	CRO-GA	0.158	0.052	0.00	N/A	N/A	N/A	90.00	97.70	100
J60	GA	1.74	1.13	0.86	11.74	10.97	10.65	72.70	74.00	76.25
	CRO	1.70	1.10	0.84	11.73	10.97	10.649	73.00	74.16	76.25
	MAOA	1.75	1.32	0.84	11.67	10.84	10.64	73.00	74.20	76.25
	MAOA-GA	1.63	1.09	0.84	11.66	10.84	10.64	73.00	74.20	76.46
	CRO-GA	1.60	1.02	0.82	11.64	10.80	10.627	73.13	74.38	76.70
J90	GA	2.64	2.18	1.49	15.15	14.60	12.23	56.67	57.70	59.20
	CRO	2.56	2.15	1.47	15.11	14.56	12.23	57.00	57.92	59.20
	MAOA	3.25	2.10	1.25	15.72	15.03	12.47	56.67	57.50	58.96
	MAOA-GA	3.00	2.00	1.25	15.65	15.00	12.47	57.08	57.70	58.96
	CRO-GA	2.03	1.83	1.22	14.83	14.21	12.15	57.35	58.62	60.21
J120	GA	4.30	3.08	2.50	33.92	32.50	31.50	31.67	33.15	34.17
	CRO	4.26	3.06	2.51	33.88	32.47	31.50	32.50	33.15	34.17
	MAOA	5.23	3.57	2.94	33.87	32.64	31.02	32.15	32.92	34.17
	MAOA-GA	4.85	3.50	2.90	33.87	32.60	31.02	32.15	32.92	34.17
	CRO-GA	4.08	3.03	2.42	33.85	32.42	30.95	32.70	33.15	35.00

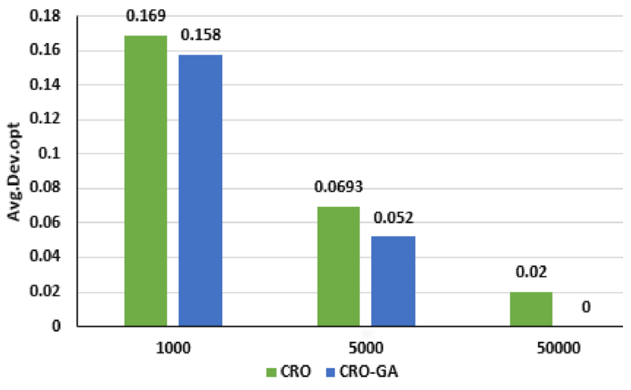
Table 3 for J30 Avg.Dev.Cpm.lb(%) are denoted by N/A (not applicable). Only average deviation from optimal has been estimated in case of J30. Deviation from best and critical path method lower bound are measured in case of J60, J90, and J120 since the optimal deviation for them are not given. The CRO and MAOA separately performed nearly same for our dataset where GA showed poor performance in case of both average deviation and number of optimal solutions. We also compared the result of MAOA-GA with the result of our proposed model where CRO-GA calculates the average deviation 0.15%, 1.60% 2.03% and 4.08% which are lowest comparing with other algorithms for 1000 number of schedules. The proposed algorithm also shows better performance than others for J60. J90 and J120 datasets.

We also compared the performance of CRO and CRO-GA by illustrating the graphical representation which can be seen in Fig. 12.The graphical representation shows that the hybrid algorithm always perform better while comparing with CRO algorithm alone. We also analyzed the percentage of GA triggering when CRO becomes unsuccessful to find the optimal solution as shown in Fig 13 for each dataset. Here, we did our experiment using 1000, 5000, 50000 schedules. It can be noticed from the Figure 13 that when the number of schedules increases, the percentage of GA triggering decreases.

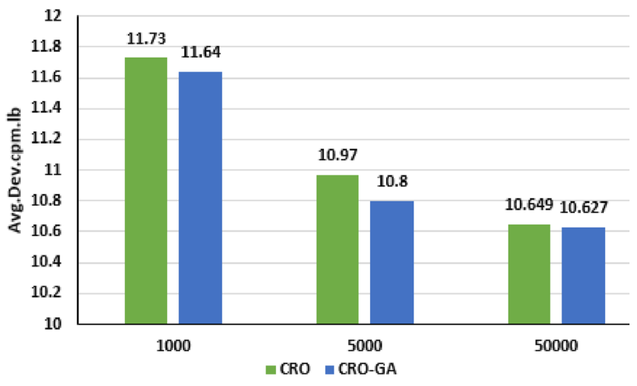
6.2 Comparisons with other algorithms

We compared the performance of our proposed hybrid algorithm with other four existing state of the art works e.g., A-PSO Kumar and Vidyarthi [20], MAOA Zheng and Wang [32], ABC Jia and Seo [16], GA Montoya-Torres et al. [24] that related to RCPSP. We analyzed the performance for each dataset separately. Although the A-PSO, MAOA, ABC have been experimented with similar number of schedules, GA was not applied for the same number of schedules. Hence, most of the comparisons for GA remain empty. Besides, all these existing algorithms worked with J30, J60, and J120 datasets but they did not experiment with J90 dataset.

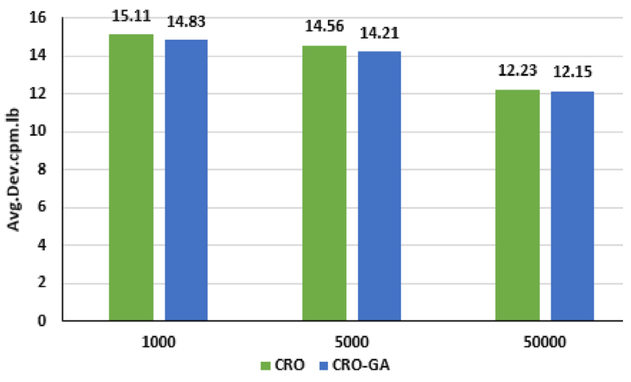
Tables 4, 5 and 6 show the comparison of experimental results for J30, J60, J120 respectively between CRO-GA and other mentioned algorithms. The algorithm that did not refer any output for a particular case is denoted by '-' (hyphen) in the tables. The percentage of average deviation



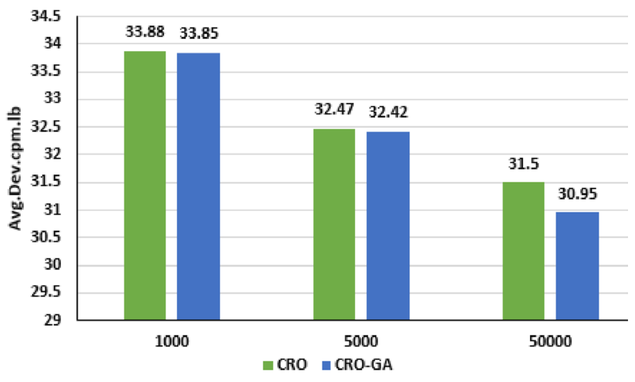
(a) Comparison of J30 dataset



(b) Comparison of J60 dataset

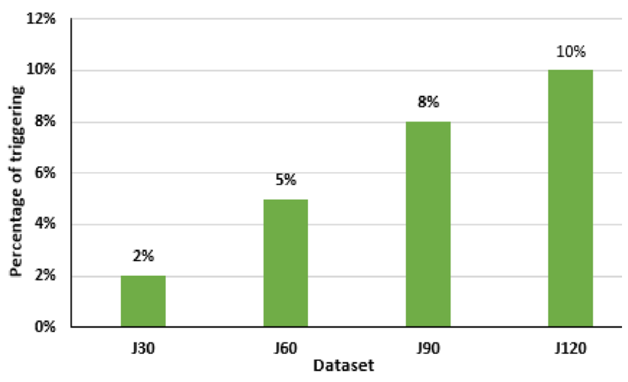


(c) Comparison of J90 dataset

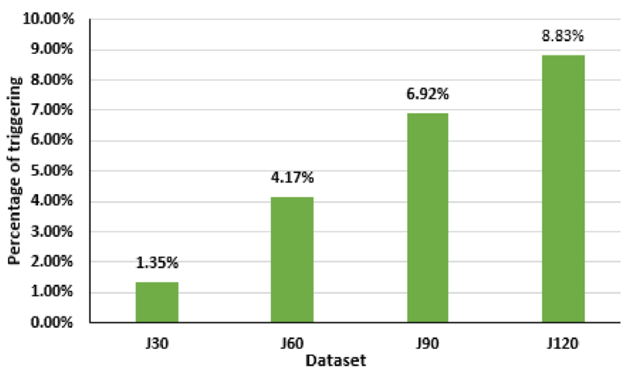


(d) Comparison of J120 dataset

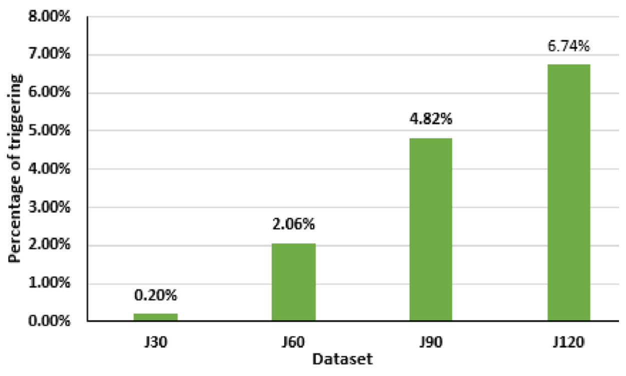
Fig. 12 Comparison of Avg.Dev between CRO and CRO-GA corresponding to Table 3 for all datasets



(a) Percentage of GA triggering for 1000 schedules



(b) Percentage of GA triggering for 5000 schedules



(c) Percentage of GA triggering for 50000 schedules

Fig. 13 Comparison of percentage of GA triggering for different number of schedules

Table 4 Comparison of CRO-GA with other algorithms for J30

Algorithm	References	Avg.Dev.opt(%)			No. of Optimal Found(%)	
		1000	5000	50000	1000	5000
CRO-GA	Present Work	0.158	0.052	0.00	90.0	97.70
A-PSO	Kumar and Vidyarthi [20]	0.28	0.06	–	88.92	97.48
MAOA	Zheng and Wang [32]	0.17	0.06	0.01	89.87	97.13
ABC	Jia and Seo [16]	0.49	0.17	–	86.60	91.74
GA	Montoya-Torres et al. [24]	2.0	–	–	–	–

Table 5 Comparison of CRO-GA with other algorithms for J60

Algorithm	References	Avg.Dev.lb(%)		Avg.Dev.cpm.lb(%)			No. of Optimal Found(%)	
		1000	5000	1000	5000	50000	1000	5000
CRO-GA	Present Work	1.60	1.02	11.64	10.80	10.627	73.13	74.38
A-PSO	Kumar and Vidyarthi [20]	3.02	2.76	11.94	11.12	–	73.02	75.58
MAOA	Zheng and Wang [32]	1.75	1.32	11.67	10.84	10.64	73.00	74.20
ABC	Jia and Seo [16]	3.35	3.02	12.35	11.96	–	72.50	74.03
GA	Montoya-Torres et al. [24]	5.0	–	–	–	–	–	–

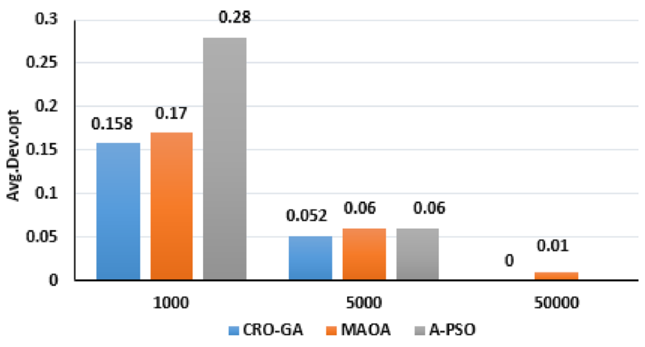
Table 6 Comparison of CRO-GA with other algorithms for J120

Algorithm	References	Avg.Dev.lb(%)		Avg.Dev.cpm.lb(%)			No. of Optimal Found(%)	
		1000	5000	1000	5000	50000	1000	5000
CRO-GA	Present Work	4.08	3.03	33.85	32.42	30.95	32.70	33.15
A-PSO	Kumar and Vidyarthi [20]	8.02	7.24	34.93	32.49	–	30.75	32.96
MAOA	Zheng and Wang [32]	5.23	3.57	33.87	32.64	31.02	32.15	32.92
ABC	Jia and Seo [16]	8.12	7.42	36.84	35.79	–	29.50	31.20

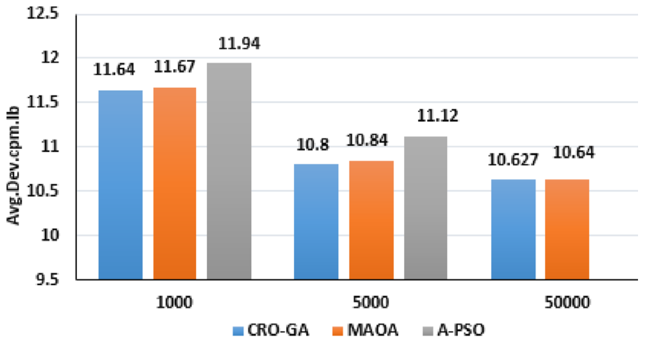
and percentage of the number of optimal solutions obtained can be seen for J30, J60 and J90 datasets in Tables 4, 5 and 6. The proposed CRO-GA algorithm showed 0.158%, 0.052% and 0.00% average deviation for 1000, 5000 and 50,000 schedules for J30 dataset and solved 90% and 97.70% instances optimally which are better than other algorithms. Since CRO-GA also performed better for 50,000 schedules, it can be stated that the hybrid algorithm can explore the search space effectively.

The performance analysis of the proposed algorithm for J60 and J120 datasets has been done by comparing the percentage of average deviation from the best lower bound and critical path lower bound since the optimal results for these datasets are not given. For J60, the proposed method was evaluated for 1000 and 5000 schedules in case of average deviation from best lower bound where it was executed using 1000, 5000 and 50000 schedules for average deviation from critical path lower bound. Although the proposed hybrid algorithm performs almost similar while comparing for critical path lower bound, it shows 1.60% and 1.02% average deviation from best lower bound which is almost double than A-PSO and ABC algorithm for both 1000 and 5000 schedules. The percentage of calculating number of optimal solutions using CRO-GA is nearly same with other algorithms.

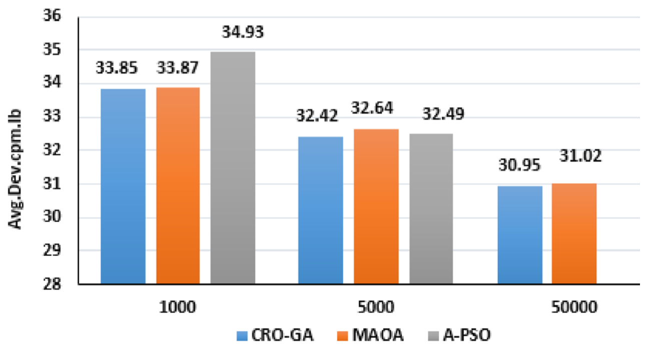
Further, the similar results have been reproduced for J120 dataset which have been reported in Table 6. The proposed method performs better for measuring the percentage



(a) Comparison of J30 dataset



(b) Comparison of J60 dataset



(c) Comparison of J120 dataset

Fig. 14 Comparison of CRO-GA and other algorithms with respect to Avg.Dev.cpm.lb for three datasets

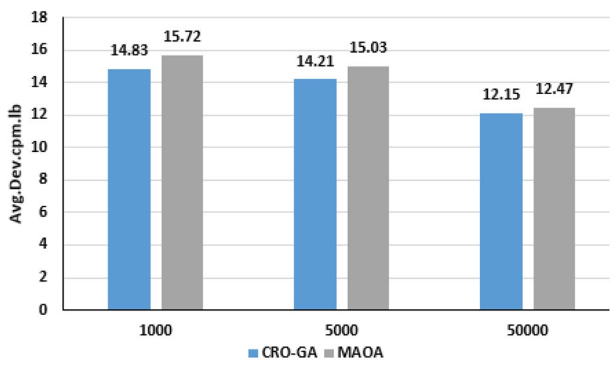


Fig. 15 Comparison of CRO-GA and MAOA for J90 dataset

of optimal solution for J120 dataset which are 32.70% and 33.15% respectively.

The graphical representations of performance analysis of our proposed algorithm with MAOA, and A-PSO has been shown in Fig. 14(a) to 14(c). Since the MAOA was not tested for J90 dataset Zheng and Wang [32] in the existing study, for fair comparison, we executed MAOA with J90 dataset in our environment and the graphical representation of comparison between CRO-GA and MAOA with respect to J90 dataset has been shown in Fig. 15.From the above performance analysis of the proposed algorithm, it can be seen the CRO-GA provides much better results than other heuristics and metaheuristic algorithms considering the same number of schedules. The reason behind the better performance of CRO-GA can be the hybridization of CRO with GA. While CRO failed to provide optimal results for some instances, the operators of GA are executed on these instances to calculate the minimum makespan.

6.3 Run time comparison

We compared the execution time of our proposed model with GA, CRO, MAOA and MAOA-GA which were implemented in our environment. The execution time of each algorithm was calculated by setting the best parameter values given in Tables 1, 2. We calculated the execution time for all the datasets with different number of schedules that has been reported in Table 7. Although the run time of GA and CRO algorithm is nearly similar, the experimental results for MAOA and MAOA-GA are different. Our proposed CRO-GA takes less time than both MAOA and MAOA-GA. The average running time of the proposed model is 6.65s, 43.56s and 463.24s for 1000, 5000 and 50000 schedules. We also illustrated graphical representations of execution time among MAOA, MAOA-GA and CRO-GA which are shown in Fig. 16(a) to 16(c).

After the performance analysis of our proposed hybrid algorithm, it can be concluded that CRO-GA is more efficient and effective in both cases of deviation and computational time in solving resource constrained project scheduling problem.

6.4 Comparison using Student's t-test

From the above discussion it can be seen that our proposed method has better performance over all other compared algorithms both in average deviation and execution time. Now we want to show a statistical significance test over the algorithms CRO-GA and MAOA. Here, 50,000 schedules were used for all datasets in the experiment. We have executed both CRO-GA and MAOA eight times for each dataset using same experimental setup. There are two hypothesis for the experiment that are stated as H_0: There is no statistical difference between CRO-GA and MAOA and alternative hypothesis, H_1 : CRO-GA is statistically significant than MAOA. We define significance level, $\alpha = 0.05$. Since the number of test cases is 8, the degree of freedom, dof = (8+8-2) = 14. The decision state value that means t_{value} is calculated using Eq. 4.

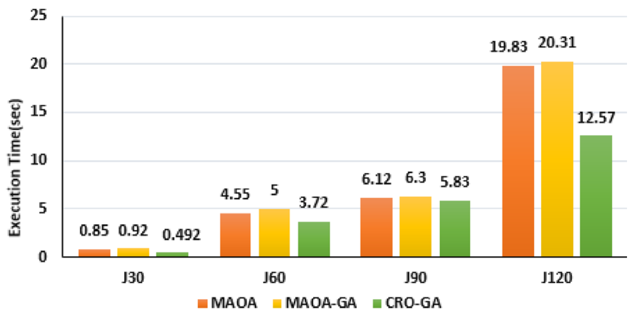
$$t_{value} = \frac{|v_1 - v_2|}{\sqrt{std_1 - std_2}} \tag{4}$$

Here v_1, v_2 are average of deviation values of two algorithms and std_1, std_2 are standard deviation of deviation values of two algorithms respectively.

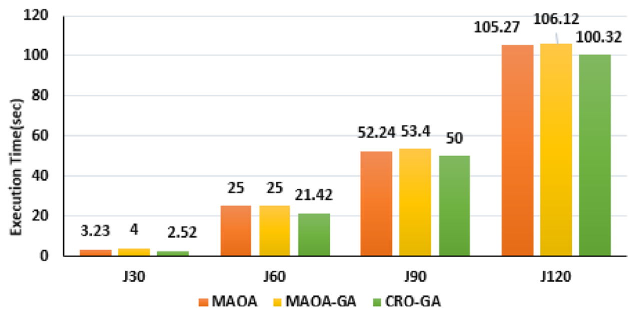
We get the critical value, $t_{crit} = 2.145$ from the t-distribution table at $\alpha = 0.05$ and dof=14 values. The null hypothesis is set that there is no significance difference between CRO-GA and MAOA. The null hypothesis can be rejected

Table 7 Execution time comparison between CRO-GA and MAOA

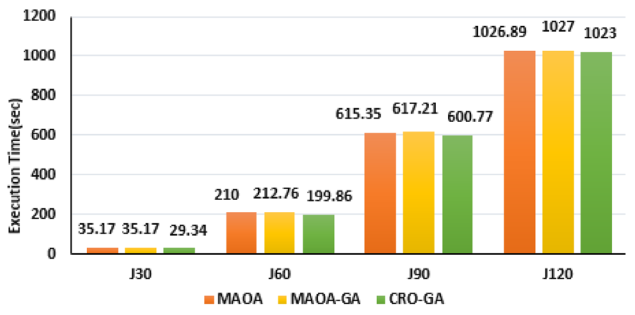
Algorithm	1000 schedules Avg. Time(s)				5000 schedules Avg.Time(s)				50000 schedules Avg.Time(s)			
	J30	J60	J90	J120	J30	J60	J90	J120	J30	J60	J90	J120
GA	0.41	3.60	5.50	11.20	2.00	20.30	49.00	99.23	29.26	195.00	595.26	1020.00
CRO	0.41	3.60	5.50	11.20	2.11	20.38	49.20	99.50	29.31	196.00	596.00	1020.00
MAOA	0.85	4.55	6.12	19.83	3.23	25.00	52.24	105.27	35.17	210.00	615.35	1026.89
MAOA-GA	0.92	5.00	6.30	20.31	4.00	25.00	53.40	106.12	35.17	212.76	617.21	1027.00
CRO-GA	0.492	3.72	5.83	12.57	2.52	21.42	50.00	100.32	29.34	199.86	600.77	1023.00



(a) Time comparison in case of 1000 schedules.



(b) Time comparison in case of 5000 schedules.



(c) Time comparison in case of 50000 schedules.

Fig. 16 Comparison of Execution time among MAOA, MAOA-GA and CRO-GA for different number of schedules

Table 8 Average deviation value for J30 dataset

No	MAOA	CRO-GA
1	0.0142	0.012
2	0.015	0.01
3	0.013	0.0143
4	0.02	0.0127
5	0.0125	0.011
6	0.018	0.0134
7	0.0173	0.015
8	0.011	0.01

Table 9 Average deviation value for J60 dataset

No	MAOA	CRO-GA
1	10.69	10.65
2	10.72	10.68
3	10.69	10.64
4	10.80	10.76
5	10.64	10.62
6	10.77	10.66
7	10.74	10.69
8	10.69	10.62

and it can be said that CRO-GA and MAOA are significantly different if $t_{value} \quad t_{crit}$ or $t_{value} < -t_{crit}$.

Using the values from Table 8 we calculated $t_{value} = 2.22$. Since t_{value} is greater than $t_{crit} = 2.145$, we can reject the null hypothesis and it can be said that there is a statistical difference between CRO-GA and MAOA algorithms in case of J30 dataset. In the above case, $v_1 = 0.0151$, $v_2 = 0.0123$ and $std_1 = 0.00372$, $std_2 = 0.00189$.Similarly using the values from Table 9 we calculated $t_{value} = 2.21$. Since t_{value} is greater than $t_{crit} = 2.145$, we can reject the null hypothesis and it can be said that there is a statistical difference between CRO-GA and

Table 10 Average deviation value for J120 dataset

No	MAOA	CRO-GA
1	31.23	31.13
2	31.08	31.04
3	31.04	31.02
4	31.17	31.09
5	31.13	31.10
6	31.07	31.03
7	31.09	31.02
8	31.13	31.04

MAOA algorithms in case of J60 dataset. In the above case, $v_1 = 10.72$, $v_2 = 10.67$ and $std_1 = 0.051$, $std_2 = 0.045$. Similarly using the values from Table 10 we calculated $t_{value} = 2.24$. Since t_{value} is greater than $t_{crit} = 2.145$, we can reject the null hypothesis and it can be said that there is a statistical difference between CRO-GA and MAOA algorithms in case of J120 dataset. In the above case, $v_1 = 31.12$, $v_2 = 31.06$ and $std_1 = 0.06$, $std_2 = 0.04$. After the above three t-tests, it indicates that CRO-GA is statistically significant than MAOA for all datasets.

7 Conclusions

Over the last few years, RCPSP has become more vital as it has a fundamental impact on production and manufacturing. This work presents the well known optimization problem of project scheduling which is implemented by combining two population based metaheuristics called Chemical Reaction Optimization (CRO) and Genetic Algorithm (GA). Priority based selection operator has been integrated with CRO framework. This selection operator selects most fit solutions, good solutions have a higher probability of regenerating more fit solutions to reach the global optimal. As a result, our proposed algorithm has received a higher convergence rate. The operators of CRO and GA have been redesigned to solve RCPSP. GA has been combined with CRO for getting advantages from both thermodynamics and biological evolution concepts. The hard tasks during the implementation of RCPSP were estimating project duration using serial or parallel rules where all resource restrictions and precedence restrictions had to be maintained. The results of the CRO-GA algorithm are compared with pertinent algorithms such as A-PSO, MAOA, ABC, GA and the results show that CRO-GA outperforms the other algorithms. The additional priority based selection operator plays a significant role in the quick convergence of CRO. GA recoups CRO in case of non optimality. The integration of CRO, priority based selection operator and GA constructs hybrid CRO-GA method which ensures less average deviation and requires minimum computation time.

Although the experimental results show that CRO-GA works very good but converging rate is little bit slow for relatively long projects. Moreover, in a future complete study on population generation, more efficient parameter generation and more efficient hybrid technique for CRO may provide higher excellence in this field.

References

1. Akbari R, Zeighami V, Ziarati K (2011) Artificial bee colony for resource constrained project scheduling problem. Int J Ind Eng Comput 2(1):45–60
2. Baar T, Brucker P, Knust S (1999) Tabu search algorithms and lower bounds for the resource-constrained project scheduling problem. Meta-Heuristics. Springer, pp 1–18
3. Barham R, Sharieh A, Sliet A (2016) Chemical reaction optimization for max flow problem. IJACSA). Int J Adv Comput Sci Appl 7(8)
4. Blazewicz J, Lenstra JK, Kan AR (1983) Scheduling subject to resource constraints: classification and complexity.Discrete Appl Math 5(1):11–24
5. Boctor FF (1990) Some efficient multi-heuristic procedures for resource-constrained project scheduling. Eur J Operational Res 49(1):3–13
6. Boctor FF (1996) Resource-constrained project scheduling by simulated annealing.Int J Prod Res 34(8):2335–2351
7. Bouleimen K, Lecocq H (2003) A new efficient simulated annealing algorithm for the resource-constrained project scheduling problem and its multiple mode version. Eur J Operational Res 149(2):268–281
8. Brooks GH (1969) An algorithm for finding optimal or near optimal solutions to the production scheduling problem. J Indl Eng 16(1):34–40
9. Demeulemeester EL, Herroelen WS (1996) An efficient optimal solution procedure for the preemptive resource-constrained project scheduling problem. Eur J Operational Res 90(2):334–348
10. Drexl A, Gruenewald J (1993) Nonpreemptive multi-mode resource-constrained project scheduling. IIE Trans 25(5):74–81
11. Garg H (2016) A hybrid pso-ga algorithm for constrained optimization problems. Appl Math Comput 274:292–305
12. Herbots J, Herroelen W, Leus R (2004) Experimental investigation of the applicability of ant colony optimization algorithms for project scheduling
13. Islam MR, Arif IH, Shuvo RH (2019) Generalized vertex cover using chemical reaction optimization. Appl Intell 1–21
14. Islam MR, Saifullah CK, Asha ZT, Ahamed R (2018) Chemical reaction optimization for solving longest common subsequence problem for multiple string. Soft Comput 1–25
15. Jarboui B, Damak N, Siarry P, Rebai A (2008) A combinatorial particle swarm optimization for solving multi-mode resource-constrained project scheduling problems. Appl Math Comput 195(1):299–308
16. Jia Q, Seo Y (2013) Solving resource-constrained project scheduling problems: conceptual validation of flp formulation and efficient permutation-based abc computation. Comput Operations Res 40(8):2037–2050
17. Kabir R, Islam R (2018) Chemical reaction optimization for rna structure prediction. Appl Intell 1–24
18. Kelley JE (1963) The critical-path method: resource planning and scheduling. Ind Scheduling
19. Kolisch R, Sprecher A (1997) Psplib-a project scheduling problem library: Or software-orsep operations research software exchange program. Eur J Operational Res 96(1):205–216
20. Kumar N, Vidyarthi DP (2016) A model for resource-constrained project scheduling using adaptive pso. Soft Comput 20(4):1565–1580
21. Lam AY, Li VO (2010) Chemical-reaction-inspired metaheuristic for optimization. IEEE Trans Evolutionary Comput 14(3):381–399
22. Mahmud MR, Pritom RM, Islam MR (2017) Optimization of collaborative transportation scheduling in supply chain management with tpl using chemical reaction optimization. In Computer and Information Technology (ICCIT), 2017 20th International Conference of, pages 1–6. IEEE
23. Merkle D, Middendorf M, Schmeck H (2002) Ant colony optimization for resource-constrained project scheduling. IEEE Trans Evolutionary Comput 6(4):333–346

24. Montoya-Torres JR, Gutierrez-Franco E, Pirachicán-Mayorga C (2010) Project scheduling with limited resources using a genetic algorithm. Int J Project Managt 28(6):619–628
25. Nonobe K, Ibaraki T (2002) Formulation and tabu search algorithm for the resource constrained project scheduling problem. Essays and surveys in metaheuristics. Springer, pp 557–588
26. Saifullah CK, Islam MR (2016) Chemical reaction optimization for solving shortest common supersequence problem. Comput Biolo Chem 64:82–93
27. Truong TK, Li K, Xu Y (2013) Chemical reaction optimization with greedy strategy for the 0–1 knapsack problem. Appl Soft Comput 13(4):1774–1780
28. Xu J, Lam AY, Li VO (2010) Parallel chemical reaction optimization for the quadratic assignment problem. In World Congress in Computer Science, Comput Eng Appl Comput, Worldcomp 2010
29. Xu J, Lam AY, Li VO (2011) Chemical reaction optimization for task scheduling in grid computing. IEEE Trans Parallel Distributed Syst 22(10):1624–1631
30. Zhang H, Li H, Tam C (2006) Particle swarm optimization for resource-constrained project scheduling. Int J Project Managt 24(1):83–92
31. Zhang H, Li X, Li H, Huang F (2005) Particle swarm optimization-based schemes for resource-constrained project scheduling. Automation in Construc 14(3):393–404
32. Zheng X-L, Wang L (2015) A multi-agent optimization algorithm for resource constrained project scheduling problem. Exp Syst Appl 42(15):6039–6049