Received January 6, 2021, accepted January 25, 2021, date of publication February 5, 2021, date of current version February 17, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3057366

An Immune Genetic Algorithm for Solving NPV-Based Resource Constrained Project Scheduling Problem

MD. ASADUJJAMAN[1,2], (Graduate Student Member, IEEE), HUMYUN FUAD RAHMAN[1], RIPON K. CHAKRABORTTY[1], (Member, IEEE), AND MICHAEL J. RYAN[1], (Senior Member, IEEE)

[1]Capability Systems Centre, School of Engineering and Information Technology, University of New South Wales, Canberra, ACT 2612, Australia
[2]Department of Industrial & Production Engineering, Rajshahi University of Engineering & Technology, Rajshahi 6204, Bangladesh

Corresponding author: Md. Asadujjaman (jonikhan007@yahoo.com)

ABSTRACT The net present value (NPV)-based resource constrained project scheduling problem (RCPSP) is a well-known scheduling problem in many industries, such as construction, software development, and manufacturing. Over the last five decades, although different approaches have been proposed to solve the problem, no single approach has been shown to achieve satisfactory performances with quality solutions for a wide range of problems. This study presents a hybrid immune genetic algorithm (IGA) to solve NPV-based RCPSPs. Hybridizing a genetic algorithm (GA) with an immune algorithm (IA) enhances the overall performance of their standalone components (i.e., only GA or IA). Performance of the proposed IGA is further improved by applying a variable insertion based local search (VINS) and forward-backward improvement (FBI). A restart mechanism is presented to the algorithm which induces diversity and helps to avoid becoming trapped in local optima. Moreover, an activity move rule (AMR) is implemented to shift the negative cash flow associated activities to further improve the NPV. Taguchi Design of Experiment (DOE) is conducted to investigate the impact of various parameters and to determine the appropriate set of parameters for the proposed IGA. The performances of the proposed algorithms are tested on 17,280 standard benchmark instances ranging from 25 to 100 activities. Comparison with the state-of-art algorithms through extensive numerical experiments reveal the effectiveness of the proposed algorithms. Overall, the proposed algorithm outperforms existing algorithms, particularly the projects with 0% and 100% negative cash flow associated activities, the 75-activity instances, and the projects with two resources usage in terms of a lower value of average percentage deviation.

INDEX TERMS Net present value, resource constrained project scheduling, RCPSPDC, immune genetic algorithm.

I. INTRODUCTION

A project is a unique endeavor that is constrained by the completion time, budget and resources. The primary objective of any project is to complete on time, within financial and resource constraints. Project scheduling is a major part of project management that typically involves the selection of the start and end time of any work [1]. Critical path method (CPM) is a widely used managerial tool for planning and scheduling projects for making decisions. However, CPM may not obtain optimal solution as it ignores the resource constraints and deadline of the project [2]. In practice, projects are depending on the limited resources and a completion time. Therefore, the duration determined by the CPM is practically unrealistic. The RCPSP determines the timetable of any task whilst satisfying the precedence relationships of project activities as well as the project's resource constraints [3], [4]. The classical RCPSP has been a widely studied issue over the past few decades, where the objective function is to minimize the length of the schedule, that is the project's makespan. Nonetheless, by focusing solely on makespan, the classical RCPSP does not address a project's economy as it neglects

The associate editor coordinating the review of this manuscript and approving it for publication was Arif Ur Rahman.

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

cash flows of the project. Many larger scale capital intensive projects such as construction engineering, power plants or infrastructure projects run over a long time horizon which involve huge cash flows [5], [6]. Therefore, owing to the time value of money (TVoM), a small variation in cash flow timing may have a substantial impact on the overall project's profitability [7]. As a result, from financial point of view, project scheduling focusing solely on the minimization of makespan is not always desirable [7]–[9]. Consequently, in RCPSP, financial perspectives have become an important point in decision-making by the project managers.

The classical RCPSP has therefore been extended to take into account of discounted cash flows (RCPSPDC), and to concentrate on the project profitability by maximizing the net present value (NPV) of the cash flows of the activities, which is also known as the NPV-based RCPSP. Cash flows are either positive or negative. In projects, on one hand, negative cash flows occurs due to the expenditure of resource procurement and activity execution; elseway, the positive cash flows come only from sales or payments made by the clients. The project's NPV depends on the TVoM of the cash flows. Therefore, the TVoM is considered by discounting the net cash flows, that is the difference between cash inflows and outflows. As the RCPSPDC is based on maximizing the NPV instead of the makespan, therefore, the project is bounded to its completion period, which is the project's deadline.

The RCPSPDC is an NP-hard problem [10]–[12], which means that exact methods are limited to solve small-sized problems within reasonable computational efforts [13]. To overcome this challenge, numerous heuristic and meta-heuristic algorithms are proposed for solving the RCPSPDC; example includes, Lagrangian relaxation (LR) procedure [11], [12], simulated annealing (SA) [5], [14], [15], tabu search (TS) [5], [14]–[16], scatter search (SS) [17], memetic algorithm (MA) [18], ant colony optimization (ACO) [19], [20] and genetic algorithms (GA) [10]. However, based on the problem's NP-hard characteristics and the no free lunch theorem, for both a relatively good solution and reasonable computing time, no single heuristic or meta-heuristic algorithm is suitable for solving the RCPSPDC [21]. Moreover, the literature shows that the hybridization of meta-heuristics overcomes the limitations of a single meta-heuristic algorithm [22]–[27].

Motivated by this fact, this study proposes a hybrid meta-heuristic algorithm to solve the RCPSPDC by the hybridization of the GA and the IA, creating an immune genetic algorithm (IGA). The GA is a powerful biological mechanism and natural selection theory-based meta-heuristic algorithm [28], [29]. Among the population-based meta-heuristics, GA is the most widely applied not only in RCPSP [30], [31] and RCPSPDC [10], but also in others sort of optimization domains [27], [32]–[34]. GA generates good quality solutions with a reasonable time when the problem domain is large [10], [33]. On the other hand, IA is also a biologically dependent heuristics that designs to obtain the best solution by modifying the genes [35]. While IA maintains a fine diversity, as a single meta-heuristic, it has some drawbacks such as premature convergence and poor search capability [36]. On the other hand, GA, as a single meta-heuristic has limited local search capability and abortive convergence [29]. In GA, the infeasible solutions generated by the worst individuals are eliminated, although the optimal solution may be very near to those infeasible solutions. Past studies show that in evolutionary algorithms the immune property enables the prevention of immature convergence [29], [37] and enhances local search [29], [38]. Therefore, the hybridization of GA and IA enhances the search capability to find the best solutions beyond that of the algorithms applied individually. This hybrid approach has already showed effectiveness in related complex planning and scheduling problems, such as assembly line balancing problems [39]–[41], job-shop scheduling [42], [43], flow-shop scheduling [29] and layout design [44]. However, to the best of our knowledge, the utilization of IGA to RCPSPDC has not been studied in the research. Moreover, none of the existing IGAs in the literature has considered the VINS, FBI, restart scheme, and the AMR to further enhance the overall performance. In this study, the concept of IGA is therefore utilized to examine the RCPSPDC. Furthermore, different components such as VINS, FBI, restart mechanism, and AMR have been utilized to improve algorithmic efficiency. Immunization in GA thus consequently improves the solution performance by enhancing the optimal or best solution(s) search capability [45]. Therefore, the following six key contributions have been made in this study:

1) Proposing a new IGA by hybridizing the GA and IA to solve the RCPSPDC.
2) Utilizing five different priority rules with the proposed IGA to initialize population to assist in searching better solutions.
3) Examining different types of crossover and mutation operators such as single-point-crossover, double-point-crossover, swap mutation and inverse mutation to utilize the best crossover and mutation operator.
4) Introducing a restart scheme to induce diversity and to avoid trapping in local solution.
5) Adopting the VINS and the FBI as a local search to enhance the exploitation of the solutions.
6) An AMR to delay the negative cash flow associated tasks that increases the overall profitability of the project due to the impact of the TVoM.

The performances of the proposed approaches are evaluated by solving the standard benchmark problem instances available for RCPSPDC [10], [12], [17].

The rest of the paper is organized as follows. The next section presents a comprehensive literature review on solving the RCPSPDC. The mathematical model utilized in this work is described in section 3. The proposed algorithm for solving the RCPSPDC is presented in section 4. Section 5 presents the computational results and discussion. The conclusions are drawn in section 6 with future research directions.

II. RELATED WORKS ON RCPSPDC

The NPV-based project scheduling was first introduced in 1970 by Russell [46] who presented it as a nonlinear model, assuming that the parameters are deterministic in nature and ignoring the deadline and resource constraints of the project. Later, Grinold [47] added a project deadline and transformed the nonlinear model of Russell [46] into an equivalent linear model. However, the model developed by Grinold [47] ignores the resource constraints in project scheduling. Next, Neumann and Zimmermann [48] modified the model of Grinold [47] for solving problems with ensuring precedence feasibility and resource constraints. The deterministic NPV-based project and its extensions has also been studied by Herroelen *et al.* [49], Schwindt and Zimmermann [50] and Mika *et al.* [14]. To solve those existing approaches, numerous exact and approximation methods have been utilized in the literature. Approximation methods can be devided into heuristic and meta-heuristic algorithms. A review on solving RCPSPDC using exact and approximation methods is found in the literature of Gu *et al.* [13]. A brief discussion of various algorithms to solve the RCPSPDC is presented next.

A. EXACT METHODS

Exact methods are those that ensure optimality by generating the optimal schedule. The most commonly used exact method to solve the RCPSPDC is the branch-and-bound procedure [51]–[53]. Vanhoucke, *et al.* [53] presented a branch-and-bound procedure for solving the projects with discounted cash flows, however ignores the resource constraints. Their approach showed its efficiency in generating optimum solutions for only small-sized problems. Doersch and Patterson [54] examined a linear programming (LP) model for solving the RCPSPDC, where the problem was solved by zero-one LP code. Their procedure solved small-sized problems optimally, however, that approach was computationally very expensive for larger-sized problems. Next, an another exact approach namely, lazy cause generation was introduced by Schutt, *et al.* [55] to solve the RCPSPDC. That study compared their results with Vanhoucke, *et al.* [53] and showed that their algorithm generated better deadline feasible solutions than existing approaches. The exact method presented by Schutt, *et al.* [55] provides a state-of-the-art method to slove the RCPSPDC. However, exact methods have drawbacks with convergence, particularly when the problem size increases [13], [23]. As the RCPSPDC is an NP-hard problem [10], it takes non-deterministic polynomial time in finding the optimal solution. In order to overcome this issue, numerous approximation methods have been developed to find the near optimal solution within a reasonable computational cost. The following section presents different approximation methods to solve the RCPSPDC.

B. HEURISTICS AND META-HEURISTICS ALGORITHMS

The first heuristic algorithm for RCPSPDC was developed by Russell [56] who revealed that the heuristics for makespan minimization is not necessarily suitable for NPV maximization. Later, Smith-Daniels and Aquilano [57] presented a backward scheduling concept for maximizing the project's NPV. Baroum and Patterson [58] developed cumulative cash flow weight based heuristic that resulted in increasing the NPV. Heuristic algorithms were also presented by Yang *et al.* [59], Pinder and Demeulemeester [60], Padman *et al.* [61], and Waligóra and Ró ycki [62] to solve the RCPSPDC. Among them, Pinder and Demeulemeester [60] assumed that no payments were made during the project life cycle, that means only one payment is done after completing the project. Moreover, they also ignored the concept of TVoM in their model. Furthermore, Yang *et al.* [59] presented nine heuristic algorithms however, their performance was limited to small activity projects.

With respect to the meta-heuristic algorithms, Icmeli and Erenguc [16] presented the TS method and tested their model on Patterson's dataset [63]. Chen and Chyu [18] apply MA to effectively solving small activity projects up to 20 tasks. Shou [19] also tested Patterson's dataset [63] applying ACO that improved the NPV of the projects. Chen *et al.* [20] present ACO that outperforms GA, SA and TS for projects up to 98 activities.

Patterson's dataset [63] contains instances between 7 and 50 activities in which the number of resources varies between 1 and 3. Later, Vanhoucke [17] introduced a new standard benchmark dataset for RCPSPDC consisting of 17,280 instances with four different sizes of activities (25, 50, 75 and 100) with the number of resources 2 or 4 which contains more complex instance problems. Vanhoucke [17] developed an SS heuristic to solve those datasets. Next, Gu *et al.* [11] applied LR heuristic for large project instances of the Vanhoucke's dataset [17]. Gu, *et al.* [12] further improved the results of Schutt *et al.* [55] using FBI method in LR procedure to meet the tight deadline of the projects. They also hybridized LR with constraint programming and showed the effectiveness of their hybrid approach by comparing with the study of Vanhoucke [17]. Thituvady *et al.* [64] hybridized LR with ACO (LR-ACO) and compared their findings with the study of Vanhoucke [17] for 100 activities. The LR-ACO showed better results, particularly for projects with positive cash flow activities. However, when the number of negative cash flow activities were larger and the project deadline was tight, SS [17] algorithm was more effective than the LR-ACO. Later, Leyman and Vanhoucke [10] solved those benchmark datasets by examining the GA and compared performance with the results of Gu, *et al.* [12] and Vanhoucke [17]. Their study reported that their proposed GA outperformed a few existing algorithms.

C. SUMMARY

From the above literature review on RCPSPDC, it can be seen that, many exact methods, heuristics, and meta-heuristic algorithms have been proposed over the years. Among them, GA is a simple and effective meta-heuristic algorithm, which has a good track record to solve different complex problems with combinatorial optimization [28], as well as for solving RCPSPDC [10]. However, GA has two main disadvantages: lack of a local search capability and premature convergence [29], [65]. Therefore, to overcome these drawbacks, GA combines with other meta-heuristics [29]. The main reasons of the hybridizing GA and IA are: to improve the search area and to avoid local optima [40], [43], [44]. To take the advantages of hybridization, an IGA has been proposed in this research for solving the RCPSPDC. A brief discussion on the proposed IGA with its different components is presented in Section IV.

III. PROBLEM DESCRIPTION

The RCPSPDC network is represented as an activity-on-node *(A-o-N)* network *G(N, Pred)*, where N represents the project nodes or activities, and *Pred* represents the precedence relationship between the activities. Activity 1 and N are dummies. The mathematical formulation for the classical RCPSPDC [10], [12], [17] is as follows:

$$MaxZ = \sum_{i=1}^{N} C_i e^{-\alpha F_i} \tag{1}$$

Subject to:

$$F_i \leq F_j - D_j, \quad \forall (i,j) \in Pred \tag{2}$$

$$\sum_{i \in S(t)} R_{ik} \leq A_k, \quad k = 1, \ldots, K; \; t = 1, \ldots, D \tag{3}$$

$$F_N \leq D \tag{4}$$

$$F_i \; integer, \quad \forall i \in N \tag{5}$$

where,

N: the number of activities

K: the number of renewable resources

C_i: the cash flow of activity i

F_i: the finish time of activity i

α: the discount rate

D_j: the known duration of activity j

$S(t)$: the set of in-process activities at time t

R_{ik}: the K type renewable resource demand for activity i

A_k: the constant availability of K type renewable resource

D: the project's due date.

The objective function (1) seeks to maximize the project's NPV, whereas constraint (2) ensures precedence relationship of the project activities. Inequality (3) ensures the renewable resource availability constraint at time t, i.e., the demand of the renewable resources (R_{ik}) must satisfy its availability (A_k). Inequality constraint (4) presents the project's deadline constraint. The finish time of the activities will be a non-negative integer value presented in constraint (5), which is the decision variable. When a project fails to meet the due date, the following penalty function is imposed to the project's NPV [10].

$$NPV = -Y_1 + NPV_{inf}.Y_2^{F_n - D}; \quad \text{if } NPV_{inf} \geq 0 \tag{6}$$

$$NPV = -Y_1 + \frac{NPV_{inf}}{Y_2^{F_n - D}}; \quad \text{if } NPV_{inf} \leq 0 \tag{7}$$

where, NPV_{inf} is the NPV of the deadline infeasible schedule. Y_1 is a large fixed value (=20,000) [10]. Y_2 aims to penalize the NPV depending on the difference of the project's makespan from the due date. A detailed discussion on the optimum value of Y_1 and Y_2 for projects with different cash flows is explained by Leyman and Vanhoucke [10].

IV. PROPOSED IMMUNE GENETIC ALGORITHM

This section presents the IGA for the RCPSPDC. GA is a biological evolution-based meta-heuristic for solving combinatorial optimization problems [28], [66], [67]. In GA, an individual is characterized by genes (activities) which are joined to form a chromosome (solution). These solutions are further evaluated by selection, crossover and mutation operations to find the best solution. On the other hand, IA is another biologically based evolutionary algorithm that purposes modifying the genes of the individuals in order to generate a better fitness [68]. In IA, the objective function of the problem is known as the antigen, whereas antibody is the required candidate solution. The biological immunization system generates antibodies and excludes antigens by altering the genes to deal with the invading antigens. [69]. Thus, IA in GA overcomes the blindness of crossover and mutation operations [29], [70] and improves global search ability. Firstly, the immunization in the IGA is affinity based which ensures the better generation as well as the best solution. To increase the local search capability of IGA, a VINS scheme is also introduced in the proposed IGA. Moreover, for better exploitation and to ensure the diversity, a restart mechanism is used. The forward-backward-improvement (FBI) first shifts the activities of a feasible schedule from left to right towards the deadline. Thus, the resulted schedule is considered to shift from right to the left which helps to meet the project's due date. Furthermore, the AMR delays the activities with negative cash flows as far as possible by pushing right towards the due date, to increase the overall profitability. A pseudo code presented in Algorithm 1 shows the generalized IGA for the RCPSPDC, and the flow chart for the proposed IGA is shown in Fig 1. The following sections are described based on the flowchart presented in Fig 1.

A. SOLUTION REPRESENTATION AND NON-RANDOM INITIALIZATION

In IGA, a solution is usually represented by binary, integer, or real number [71], [72]. A solution is also known as a chromosome. In RCPSPDC, each chromosome is represented as a sequence of the project activities. Fig. 2 shows the solution representation of the IGA for the RCPSPDC, where a set of individuals is called the population. An individual (sequence) is characterized by a set of genes (activities). The genes together form a chromosome (sequence) which is the solution of the problem. The population is evaluated based on the individual's fitness value, that is the project's NPV. The greater the NPV value, the better an individual is. The population passes through a number of operations till meeting the termination condition. A complete generation includes

Algorithm 1 IGA for RCPSPDC

```
1: Set population size ($PS$), crossover probability ($P_c$), mutation probability ($P_m$), immunization probability ($P_i$), $R\_counter$, $FBI\_counter$, max generation ($Gen$);
Non-random
2: initialization with five-priority rules $P_0$;
3: for each $i \in P_0$ do
4:     Generate schedule using the parallel schedule generation;
5:     Apply FBI;
6: end for;
7: $P \leftarrow P_0$;
8: Set $Gen$ =1;
9: while $Gen$ <= max $Gen$ do
10:     for $n$= 1 to ncross (=$PS \times P_c$) do
11:         Select two individuals as parents $i_a, i_b \in P$ by tournament selection;
12:         Generate two offsprings from $i_a$, $i_b$ by crossover presented in Algorithm 2 or Algorithm 3;
13:         if rand (0,1) < $P_m$ then
14:             Apply mutation presented in Algorithm 4 or Algorithm 5;
15:         end if;
16:     end for;
17:     for $k$= 1 to nimmune (=$PS \times P_i$) do
18:         Apply immunization presented in Algorithm 6;
19:     end for;
20:     for $i$ = worst individual of the population do
21:         Apply VINS;
22:     end for;
23:     if best fitness ($Gen$) = best fitness ($Gen$ + 1) for consecutive $R\_counter$ then
24:         Apply restart;
25:     end if;
26:     if $FBI\_counter / Gen == 0$ then
27:         Apply FBI;
28:     end if;
29:     if project activities are associated with negative cash flows then
30:         Apply AMR;
31:     end if;
32:     Apply Elitism strategy;
33:     $Gen \leftarrow Gen + 1$;
34: end while;
35: Save the best sequence and NPV;
```

a range of operations from the tournament selection to the elitism strategy presented in Fig 1.

A solution is usually generated at random, which, due to the complexity of the problem, does not guarantee the quality solution. In this study, a non-random initial population is thus generated applying a multi-priority rule. The process is as follows: the first five individuals or solutions of the initial population are generated by five priority rules namely, shortest operation first (SOF) [73], [74], minimum resource demand (MinRD) [73], [74], maximum resource demand (MaxRD) [73], [74], most total successor (MTS) [73], [75], [76], and discounted cash flow weight (CFW) [60], [77]. The rest individuals of the initial population are randomly generated. A description of the different priority rules is given below:

SOF: The activities are scheduled based on their duration; i.e., an activity with the shortest or minimum duration is being scheduled first [73], [74].

MinRD: The activities are scheduled based on the total necessary resource, which is the activity with the least resource usage being scheduled first [73], [74].

MaxRD: The activities are scheduled based on the total resource used, that is the activity with the most resource use scheduled first [73], [74].

MTS: Scheduling is done in such a way that the activity with maximum number of successors being scheduled first [73], [75], [76].

CFW: The activities are scheduled in such a way that an activity with highest value of discounted cash flow based on activity duration being scheduled first [60].

Among the priority rules, cash flow weight [77] has already proven to be very effective rule, while other priority rules are also competitive [73] to solve the RCPSPDC. The schedule is generated by two different generation schemes [78]: serial schedule generation scheme (SSGS) that works based on activity-incremetation, and parallel schedule generation scheme (PSGS) which performs with the time-incrementation. Among them, the PSGS is better for hard problems, whereas the SSGS is better for the easy problems [79]. The PSGS generates a non-delay schedule [79]. Therefore, the schedules are generated by the renowned PSGS, in this study. After PSGS, the well-known FBI approach implemented by Li and Willis [80] is applied to ensure the deadline feasibility of the solution.

B. SELECTION MECHANISM AND GENERATION SCHEME

In this proposed IGA, the tournament selection mechanism [75], [81] is used as the selection operator, where three individuals (schedules) are chosen randomly from the population. Then, the best two individuals based on their NPV value are selected as parents. After the reproduction operation, an offspring replaces the worst individual of the current population (P_{worst}) if it generates a better NPV value than that individual. In GA, this process is well-known as the steady state [82]. The elitism strategy [21] is preserved in this study where best individual or sequence of one generation is saved and passed without alteration to its subsequent generation.

C. CROSSOVER

The crossover is a process that combines the chromosomes of the parents for creating the new offspring with a probability of P_c. Two different types of crossover operations name single-point and double-point crossover are examined in this

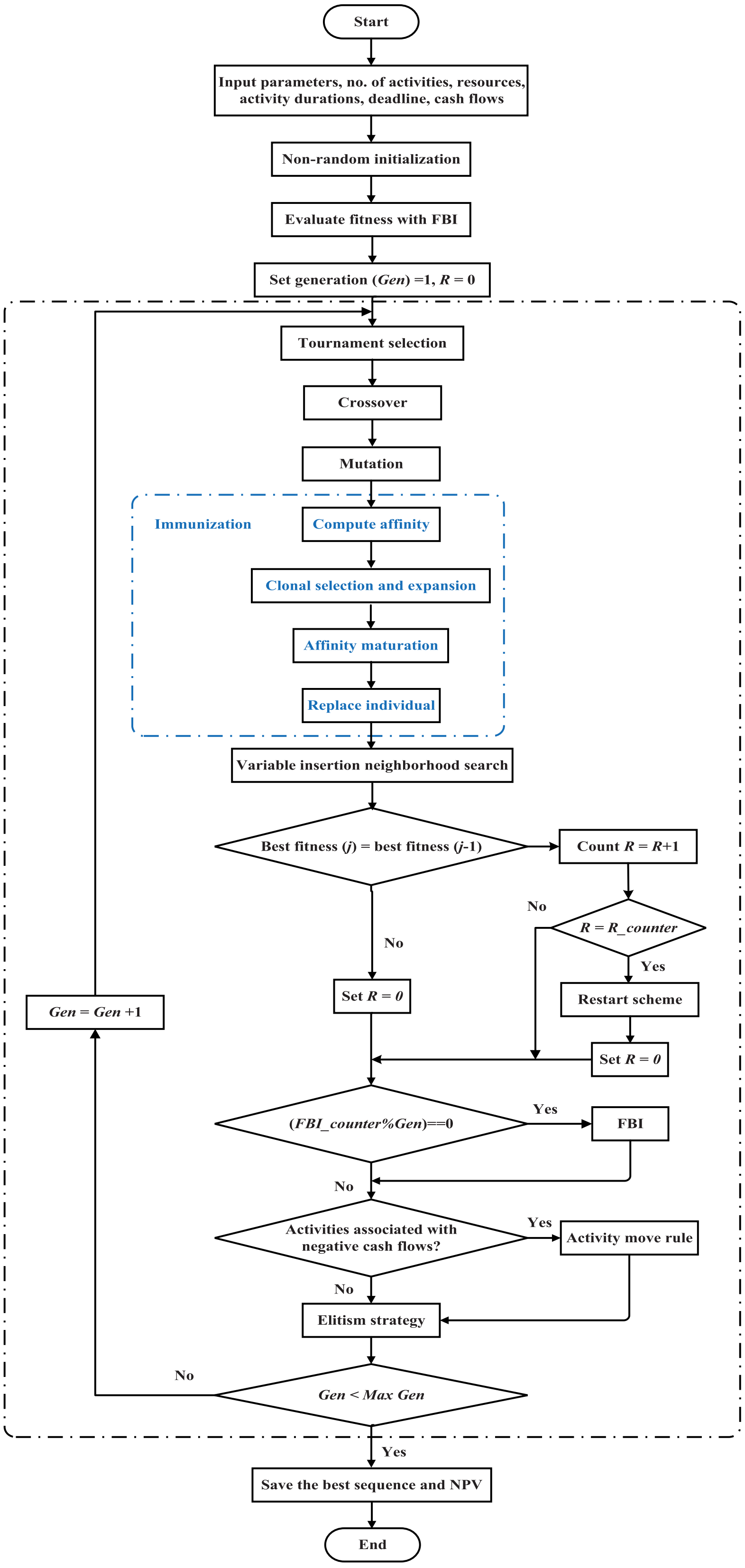

Start
Input parameters, no. of activities, resources, activity durations, deadline, cash flows
Non-random initialization
Evaluate fitness with FBI
Set generation (Gen) =1, R = 0
Tournament selection
Crossover
Mutation
Immunization
Compute affinity
Clonal selection and expansion
Affinity maturation
Replace individual
Variable insertion neighborhood search
Best fitness (j) = best fitness (j-1)
Count R = R+1
No
R = R_counter
Yes
Restart scheme
Set R = 0
No
Set R = 0
Gen = Gen +1
(FBI_counter%Gen)==0
Yes
FBI
No
Activities associated with negative cash flows?
Yes
Activity move rule
No
Elitism strategy
No
Gen < Max Gen
Yes
Save the best sequence and NPV
End

FIGURE 1. Flowchart of IGA.

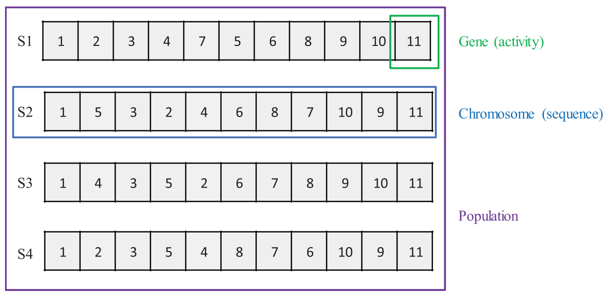

S1 1 2 3 4 7 5 6 8 9 10 11
S2 1 5 3 2 4 6 8 7 10 9 11
S3 1 4 3 5 2 6 7 8 9 10 11
S4 1 2 3 5 4 8 7 6 10 9 11
Gene (activity)
Chromosome (sequence)
Population

FIGURE 2. Solution representation of IGA for RCPSPDC.

study to solve the RCPSPDC. The following section describes different crossover operations.

1) SINGLE-POINT CROSSOVER

This study adopts the single-point crossover introduced by Hartmann [75]. In a single-point crossover operation, each position of the activity sequence is examined individually from both parents. After that, up to a random crossover point, from one of the parents each child inserts all activities. Particularly, parent 1 inserts activities directly into child 1 and parent 2 inserts activities directly into child 2. Lastly, each child's missing activities are inserted directly from the other parent following the same relative order of activities, with ensuring the precedence relationship of activities. Fig. 3 shows a project network example, while the single-point crossover for this network is shown in Fig. 4. Assume that, the value of the discount factor is 0.01 and the deadline of the project is 15 units. After crossover, the fitness of child 1 is inferior to its parent, therefore parent 1 will participate the next step. On the other hand, child 2 produces better NPV than its parent, thus it will participate the next operation. The pseudo code for crossover is presented in Algorithm 2.

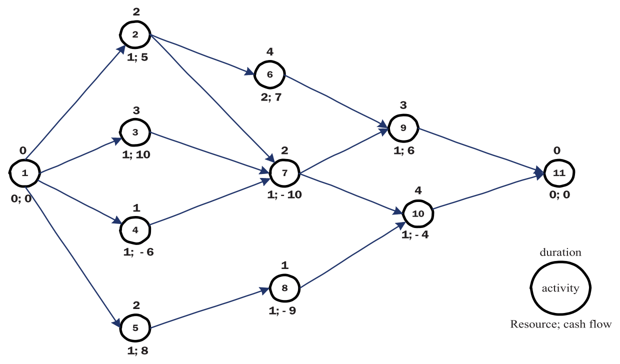


FIGURE 3. An example of project network.

2) DOUBLE-POINT CROSSOVER

In double-point crossover operation [31], [75], [83], [84], at first two cut points are selected randomly. Outside of these two points, all the activities of the first parent are introduced directly into the first child. Then, the first child's missing activities are taken in relative order from the other parent with satisfying the precedence relationship. Similarly, for the second child, the activities outside the two cut points are inserted directly from the second parent and the remaining activities are taken in relative order from the first parent with satisfying the precedence relationship. Fig. 5 shows the double-point crossover operation for the project network shown in Fig. 3. After crossover, the fitness of the children are inferior to their parent, therefore the parents will participate the next step. The pseudo code for crossover is presented in Algorithm 3.

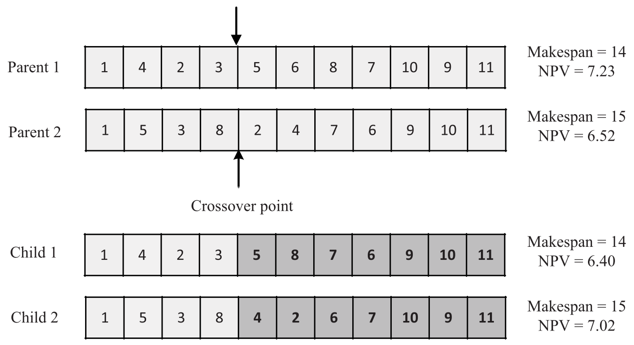

Parent 1 1 4 2 3 5 6 8 7 10 9 11 Makespan = 14 NPV = 7.23
Parent 2 1 5 3 8 2 4 7 6 9 10 11 Makespan = 15 NPV = 6.52
Crossover point
Child 1 1 4 2 3 5 8 7 6 9 10 11 Makespan = 14 NPV = 6.40
Child 2 1 5 3 8 4 2 6 7 10 9 11 Makespan = 15 NPV = 7.02

FIGURE 4. Single-point crossover.

```
Algorithm 2 Single-Point Crossover
Input: The selected parents using the tournament selection
1: Generate a random crossover point ($C_p$) between 2 and ($n - 1$);
2: Up to $C_p$, insert the activities from parent 1 to child 1 and parent 2 to child 2;
3: Insert each child's missing activities from the other parent in that parent's same relative order of activities with satisfying activity precedence;
4: Save and return child 1 and child 2;
```

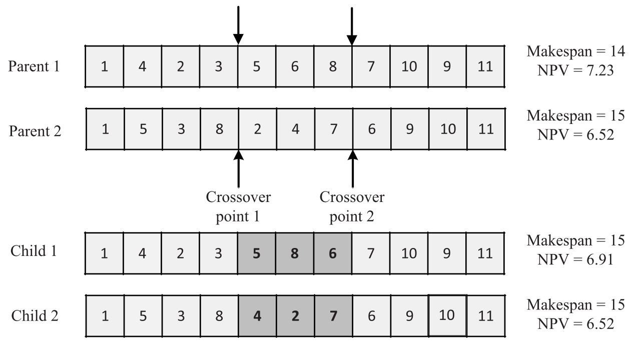

Parent 1 1 4 2 3 5 6 8 7 10 9 11 Makespan = 14 NPV = 7.23
Parent 2 1 5 3 8 2 4 7 6 9 10 11 Makespan = 15 NPV = 6.52
Crossover point 1
Crossover point 2
Child 1 1 4 2 3 5 8 6 7 10 9 11 Makespan = 15 NPV = 6.91
Child 2 1 5 3 8 4 2 7 6 9 10 11 Makespan = 15 NPV = 6.52

FIGURE 5. Double-point crossover.

D. MUTATION

To maintain diversity, after crossover, the mutation operation modifies the position of the genes (activities) for making

Algorithm 3 Double-Point Crossover

```
Input: The selected parents using the tournament selection
1: Generate two random crossover points ($C_{p1}$) and ($C_{p2}$), where $1 < C_{p1} < C_{p2} < n$;
2: Insert the activities directly from parent 1 to child 1, and parent 2 to child 2, except the activities between ($C_{p1}$) and ($C_{p2}$);
3: Insert each child's missing activities from the other parent in that parent's same relative order of activities between ($C_{p1}$) and ($C_{p2}$) while satisfying their precedence relationships;
4: Save and return child 1 and child 2;
```

some development of the child [34], [85]. In this study, two different types of mutation operations are utilized, namely, swap and inverse mutations, to solve the RCPSPDC.

1) SWAP MUTATION

In swap mutation [84], with satiafying the precedence relationship, the position of two activities are swapped randomly with a probability of P_m. Assume that, the schedule (1-4-2-3-5-6-8-7-10-9-11) is selected for the swap mutation operation. For the project network shown in Fig. 3, activity 4 may swap its position with activity 2, 3, 5, 6 and 8 after satisfying the precedence relationship. Therefore, the swap mutation between activity 4 and 5 is shown in Fig. 6. The NPV of the schedule is improved after mutation, which is also deadline feasible. Therefore, the schedule after swap mutation will compete the next operation. Algorithm 4 provides the pseudo code for the swap mutation.

2) INVERSE MUTATION

In inverse mutation [19], [84], [86], a new chromosome is generated by randomly selected two activity positions

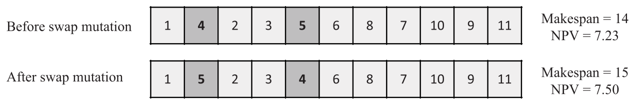


FIGURE 6. **Swap mutation.**

Algorithm 4 Swap Mutation

```
Input: The selected individuals using $P_m$
1: Generate the first random activity number ($v$) between 2 and ($n - 1$);
2: Generate the second random activity number ($w$) between 2 and ($n - 1$) where $v \neq w$;
3: Interchange the position of $v$ and $w$ with satisfying their precedence relations;
4: if $NPV(i) > NPV(P_{worst})$ then
5:     Replace the individual;
6: end if;
```

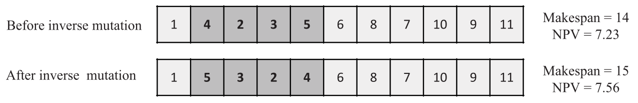


FIGURE 7. **Inverse mutation.**

and then inverting the activities between them by satisfying their precedence relationships. Assume that, the chromosome (1-4-2-3-5-6-8-7-10-9-11) is selected for the inverse mutation operation with the probability of P_m. For the project network shown in Fig. 3, if the randomly selected activity positions are 2 and 5, then after inversing the activities between them the new chromosome is (1-5-3-2-4-6-8-7-10-9-11) as shown in Fig. 7. The NPV of the schedule is improved after inverse mutation, which is also deadline feasible. Therefore, the schedule after inverse mutation will compete the next operation. The pseudo code for the swap mutation is presented in Algorithm 5.

Algorithm 5 Inverse Mutation

```
Input: The selected individuals using $P_m$
1: Randomly generate two numbers: ($v$) within $2 : n - 3$, and ($w$) within $v + 1 : n - 1$;
2: Inverse the position of the activities between $v$ and $w$, and arrange the sequence with satisfying their precedence relations;
3: if $NPV(i) > NPV(P_{worst})$ then
4:     Replace the individual;
5: end if;
```

E. IMMUNIZATION

The purpose of immunization in GA is to overcome the blindness of crossover and mutation in operation [69]. The IA in GA also gets the benefit of local information for accomplishing the best solution [69] with the immunization probability, P_i. The immunization used in this study is typically based on four steps proposed by Zandieh and Gholami [69]: affinity evaluation, clonal selection and expansion, affinity maturity, and replacement of antibodies. Algorithm 6 represents the pseudo code for immunization. The steps are described below:

1) Evaluate Affinity: Each antibody (schedule) has a certain NPV which corresponds to the antibody affinity. Drawing the concept of affinity [69], a higher affinity means a better fitness value of the antibody, the affinity function is defined as:

$$Affinity(i) = \frac{NPV(i)}{\sum_{i=1}^{nimmune} NPV(i)} \tag{8}$$

The higher value of NPV is equivalent to the higher value affinity. A higher value of antibody fitness indicates that the feasible solution is more closer to the optimum solution.

Algorithm 6 Immunization

```
Input: Individuals after crossover and mutation
1:  for $i = 1$ to $nimmune(= PS \times P_i)$ do
2:      Calculate $Affinity(i) = \frac{NPV(i)}{\sum_{i=1}^{nimmune} NPV(i)}$;
3:      Apply clonal selection and expansion where, $clone \propto affinity$;
4:      Perform affinity maturation by swap mutation;
5:      if $NPV(i) > NPV(P_{worst})$ then
6:          Replace the antibody;
7:      end if;
8:      if The individual is selected for the replacement then
9:          Save and return the solution;
10:     end if;
11: end for;
```

2) Clonal selection and expansion: In this step, the antibodies with the higher affinity from the population is selected first. After that, clones (copies) are generated in proportional to the affinity values; a higher affinity results in more clones. That means that more clones are produced from a schedule with a higher NPV value than a schedule with a lower NPV value.
3) Affinity maturation: For each clone, the affinity maturation is carried out by swap mutation where higher affinity results in lower mutation, and vice versa.
4) Antibody replacement: Finally, if it generates a better NPV than the P_{worst}, then replace the antibody.

F. LOCAL SEARCH

The local search is utilized to increase the performance of the algorithm. In this study, two different types of local search are employed: VINS and FBI. The generated solution is first go to the VINS in each generation, and after restart scheme the FBI is applied after each FBI_counter. A detail discussion on the VINS and FBI is presented if the following subsections.

1) VARIABLE INSERTION NEIGHBORHOOD SEARCH (VINS)

The VINS [21], [27] is employed to increase the performance of these proposed algorithms as a local search technique. In VINS, with satisfying the activity precedence, each activity in a sequence is inserted from position x^{th} to y^{th}, where $x \neq y$. Firstly, consider the position of the dummy activities is fixed. Then, by satisfying the precedence constraints, an activity is inserted to all feasible positions. Assume that, a schedule (1-4-2-3-5-7-6-8-9-10-11) is selected for the local search operation. Fig. 8 shows the insertion operation for the activity 4 of the sequence. With satisfying the precedence relationship, the activity 4 can only be inserted into the 3^{th}, 4^{th} and 5^{th} position of the sequence. Therefore, three new schedules are generated with the NPV of 6.85 units, 6.89 units, and 7.12 units, respectively. All generated schedules are deadline feasible. Thus, the schedule with NPV 7.12 units will replace

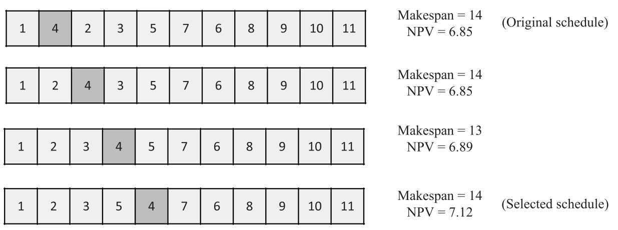
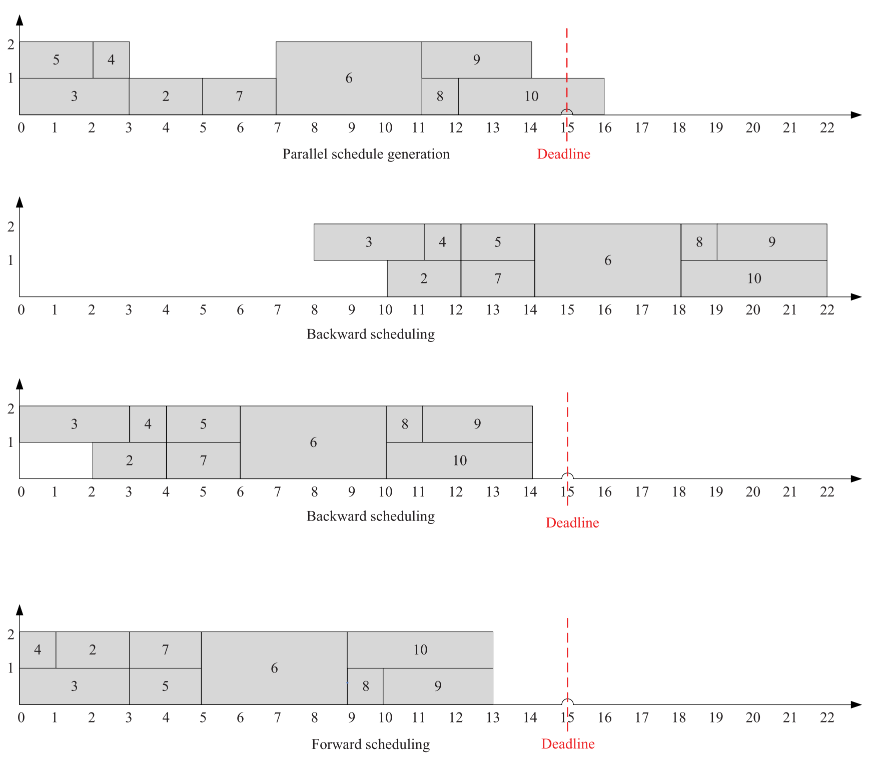


FIGURE 8. Variable insertion neighborhood search.

the original schedule. Note that, the VINS is utilized to the P_{worst} in each generation.

2) FORWARD BACKWARD IMPROVEMENT (FBI)

The FBI mechanism is a useful tool for minimizing completion time of the project [80], [87], [88]. This procedure is typically applied at the fitness evaluation stage of the current generation, after the PSGS, to meet the project's due date. Noted that, applying the FBI mechanism in every generation makes the algorithm computationally expensive. Hence, this mechanism is applied after a certain number of generations (FBI_counter). The value of the FBI_counter is determined using the Taguchi's DOE described in the subsection V-B. In FBI, the finish time of activities of a forward schedule determines the priority for the backward scheduling. Furthermore, after backward scheduling, the activity start times determine the activity priority for the next forward schedule. The pseudo code for FBI is presented in Algorithm 7. Fig. 9 also shows the FBI operation for a given schedule (1-3-5-4-2-7-6-8-9-10-11). After PSGS, the makespan of the sequence is 16 units which is deadline infeasible and the NPV is 7.25 units, as shown in Fig. 9. After that, backward schedule is generated by shifting the activities to the right, in the descending order of the finish time. Conventionally, in FBI, the backward scheduling starts from the point of makespan [80], [87], [88]. However, starting the backward schedule from the makespan may create a problem to shift the activities towards the left due to the starting time of the first activity. To avoid this problem, in this study, the backward schedule is started from the point of summation of all activity durations, (i.e. $\sum_{n=1}^{n} D_i$). After shifting all the activities towards the deadline, it is seen that there is no activity for the first eight-unit times. Thus, after shifting all the activities 8 units left, the makespan is 14 units which is deadline feasible and the NPV is 6.78 units. Similarly, for forward schedule, towards the left, the activities are moved as much as possible depending on the backwards schedule's activity start time. Then the sequence generated by the forward schedule has a makespan of 13 units with an NPV of 6.72 units. Therefore, after FBI, three different schedules are generated: schedule form PSGS, backward schedule and forward schedule. Thus, the schedule which is deadline feasible and has a better NPV value will preform to the next step. Therefore, the schedule (1-3-4-2-5-7-6-8-9-10-11) will compete the next step.



S	1	3	5	4	2	7	6	8	9	10	11
ST	0	0	0	2	3	5	7	11	11	12	22
FT	0	3	2	3	5	7	11	12	14	16	22

NPV= 7.25; Makespan = 16

S	1	3	4	2	5	7	6	8	9	10	11
ST	0	0	3	2	4	4	6	10	11	10	14
FT	0	3	4	4	6	6	10	11	14	14	14

NPV = 6.78; Makespan = 14

S	1	3	4	2	5	7	6	8	9	10	11
ST	0	0	0	1	3	3	5	9	10	9	13
FT	0	3	1	3	5	5	9	10	13	13	13

NPV = 6.72; Makespan = 13

FIGURE 9. Forward backward improvement.

G. RESTART SCHEME

A restart mechanism is utilized to avoid trapping in local optima and to maintain diversity [82], [89]. Note that, the restart scheme is utilized if the IGA has the equal best-fitness value (NPV) for consecutive *R_counter* generations. There are four steps of the restart scheme which is described as follows:

1) Firstly, sort the individuals of the population in descending order of their fitness value (NPV);
2) Secondly, the first 20% individuals are skipped as the best individuals;
3) Thirdly, 50% of the remaining 80% individuals are generated from the best individuals utilizing the swap mutation;
4) Finally, the rest 40% individuals are generated at random.

H. ACTIVITY MOVE RULE (AMR)

The AMR delays the activities with negative cash flow based on the precedence relations [10]. In AMR, the process starts with regards to the last activity of the sequence. The last dummy activity is scheduled at the time of project's deadline. After that, with satisfying the precedence relationship the negative cash flow associated activities are delayed as possible. The pseudo code for AMR is presented in Algorithm 8. Fig. 10 illustrates the AMR for the schedule, which is selected after FBI. With satisfying the precedence relationship, a delay is possible for the negative cash flow associated activity 10. Therefore, after applying the AMR, the sequence is unchanged, however the start time of the activities has changed. Thus, after applying the AMR, the makespan is 15 units and the NPV is 6.90. Note that, the AMR is only implemented when negative cash flows are associated with the activities.

I. TERMINATION CONDITION

The termination criteria is a predefined generation number. When it generates 5,000 schedules, the algorithm stops. The termination condition is related to the PS. For example, if the PS is 20, then 250 generations is the termination criteria.

J. 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 non-random initialization has a complexity of $O(n^2)$. The PSGS and FBI have the complexity of $O(n^2K)$. The selection, swap mutation, inverse mutation, and immunization have a complexity of $O(n)$. Both the single and double-point crossover have a complexity of $O(n^2)$. The computational complexity of VINS is $O(n^4K)$ in the worst case. The complexity of restart mechanism and AMR are $O(n^3n_pK)$ and $O(n^3K)$, respectively. The overall complexity is $O(n^2)+O(n^2K)+O(n_pn_gn^2K(n+n^2+n+n))+n_pn_gn^4K+n_pn_gn^3K+n_pn_gn^4K+n_pn_gn^3K$. Hence, the overall complexity of the proposed IGA is $O(n_pn_gn^4K)$.

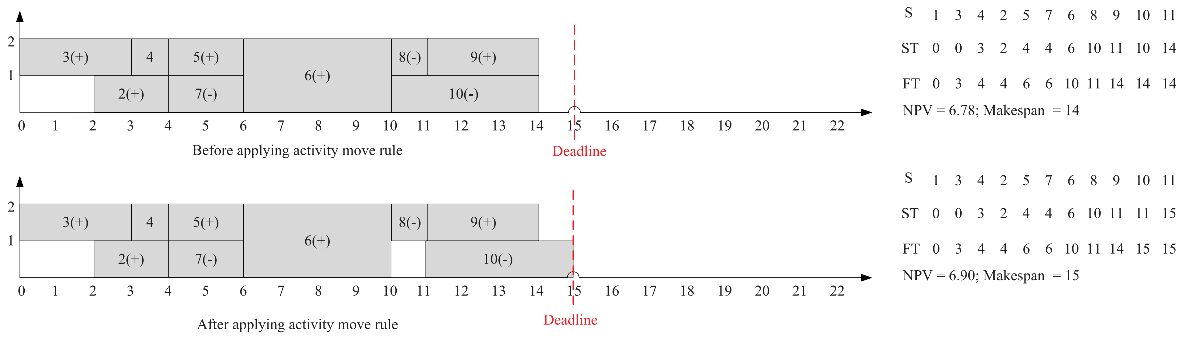


FIGURE 10. Activity move rule.

Algorithm 7 Forward Backward Improvement

```
Input: Schedule after PSGS and it's fitness, $NPV_{PSGS}$
1: for $\forall i \in P$ do
2:   for activity $n : -1 : 1$ do
3:     Set $F_n = \sum_{n=1}^{n} D_i$;
4:     if The activity ready for scheduling has all of its successors scheduled then
5:       Schedule the activity with satisfying the resources;
6:     else
7:       Select the next activity on list;
8:     end if;
9:   end for;
10:  Makespan of the backward schedule, $M_{BS} = \sum_{n=1}^{n} D_i - F_1$;
11:  NPV of the backward schedule, $NPV_{BS}$
12:  for activity $1 : n$ do
13:    Set $F_1 = 0$;
14:    if The activity ready for scheduling has all of its predecessors scheduled then
15:      Schedule the activity with satisfying the resources;
16:    else
17:      Select the next activity on list;
18:    end if;
19:  end for;
20:  Makespan of forward schedule, $M_{FS} = F_n$;
21:  NPV of the forward schedule, $NPV_{FS}$;
22:  $NPV(i) = max(NPV_{PSGS}, NPV_{BS}, NPV_{FS})$ and deadline feasible;
23:  if $NPV(i) > NPV(P_{worst})$ then
24:    Replace the individual;
25:  end if;
26: end for;
```

Algorithm 8 Activity Move Rule

```
1: for $\forall i \in P$ do
2:   for activity $n : -1 : 1$ do
3:     Set $F_n = D$;
4:     if The activity is associated with a negative cash flow and has all of its successors scheduled then
5:       Schedule the activity with satisfying the resources;
6:     else
7:       Select the next activity on list;
8:     end if;
9:   end for;
10:  Determine $NPV(i)$;
11:  Save and replace the individual;
12: end for;
```

V. COMPUTATIONAL RESULTS

The results of the computational experiments are presented in this section. All the experiments of this study were performed on an Intel(R) Core i7-3770 processor with a 16 GB RAM and 3.40 GHz CPU. The algorithms were experimented in MATLAB programming language (2018b). This research examines the standard benchmark data employed by Leyman and Vanhoucke [10], Gu *et al.* [12], and Vanhoucke [17], that are available the following link: www.projectmanagement.ugent.be. The data consists of 720 project networks with four different types of activities: 25, 50, 75 and 100. Every dataset is executed with six different negative cash flow percentage (% Neg), 0%, 20%, 40%, 60%, 80% and 100%. Moreover, as the project's due date, four different deadline increments (D-Incr), 5%, 10%, 15% and 20%, are imposed to the lower bound of RCPSP project duration. Thus, the dataset consists of (720*6*4) = 17,280 problem instances. Each project is also characterized by the order strength (OS) [90] and resource constrainedness (RC) [91]–[93]. The higher the OS, the more precedence relations among the activities. On the other hand, RC measures the average quantity needed by all project activities of each renewable resource, divided by its availability. The OS and RC of the project instances are both either 0.25, 0.50 or 0.75. The number of resource usage (RU) by an activity in

TABLE 1. Comparison among different heuristic combinations using Friedman test.

Combination Number	Different heuristic combinations	Mean NPV	Mean rank
1	GA with random initialization, single-point crossover and swap mutation	621.13	6.63
2	GA with random initialization, single-point crossover and inverse mutation	563.14	5.00
3	GA with random initialization, double-point crossover and swap mutation	623.78	6.21
4	GA with random initialization, double-point crossover and inverse mutation	585.21	5.67
5	GA with non-random initialization, single-point crossover and swap mutation	653.90	6.90
6	GA with non-random initialization, single-point crossover and inverse mutation	623.25	4.92
7	GA with non-random initialization, double-point crossover and swap mutation	666.68	6.92
8	GA with non-random initialization, double-point crossover and inverse mutation	653.88	6.50
9	IA with random initialization	563.05	4.21
10	IA with non-random initialization	610.11	5.81
11	IGA with random initialization, single-point crossover and swap mutation	997.56	13.02
12	IGA with random initialization, single-point crossover and inverse mutation	996.75	13.23
13	IGA with random initialization, double-point crossover and swap mutation	997.56	13.46
14	IGA with random initialization, double-point crossover and inverse mutation	1004.40	14.10
15	IGA with non-random initialization, single-point crossover and swap mutation	1006.44	14.65
16	IGA with non-random initialization, single-point crossover and inverse mutation	1002.99	13.94
17	IGA with non-random initialization, double-point crossover and swap mutation	**1007.59**	**14.77**
18	IGA with non-random initialization, double-point crossover and inverse mutation	**1009.89**	**15.08**

each processing period is either 2 or 4. The discount rate is employed as 1% for calculating the NPV.

A. DIFFERENT HEURISTIC COMBINATIONS

This section describes different combinations of GA, IA and IGA utilizing random or non-random initialization and different crossover and mutation operators. Before solving all the instance problems, the performance of different heuristic combinations are tested. Table 1 shows different combinations of GA and IGA. Considering OS, RC, and RU, each combination is tested with 24 instance problems; which are: rcpspdc55, rcpspdc301, rcpspdc411 and rcpspdc665 with D-Incr 5% and 20%, and % Neg 0%, 60% and 100% (i.e. 6 instances from each 25, 50, 75 and 100 activity problems). All the heuristic run with the following parameter values: $PS = 20$; $P_c = 0.6$; $P_m = 0.15$; $P_i = 0.7$, R_counter = 5 and FBI_counter = 10. The overall performance is evaluated using the non-parametric Friedman test [94], [95]. The relative ranks are calculated based on the mean NPV of the instance problems. The rank measurement using the Friedman test (Table 1) revealed that, the IGA with non-random initialization, double-point crossover and inverse mutation (IGA-IM) and the IGA with non-random initialization, double-point crossover and swap mutation (IGA-SM) are the best two heuristic combinations to solve the RCPSPDC. Therefore, all the instance problems are solved using the IGA-SM and IGA-IM.

B. PARAMETER SETTINGS

In this section, the parameter analyses for the proposed IGA is presented. Different combination of parameters may change the output of the proposed IGA. Thus, the well-known Taguchi's DOE is used to test the behaviors of the combination of different parameters for determining the best parameters set. Six different factors with three different levels for the proposed IGAs presented in Table 2 are tested. The

TABLE 2. Combination of parameter values.

Parameters	Parameter level		
	1	2	3
PS	20	40	50
P_c	0.6	0.7	0.8
P_m	0.1	0.15	0.2
P_i	0.7	0.8	0.9
$R\_counter$	5	10	15
$FBI\_counter$	10	15	20

orthogonal array employed in this study is $L_{27}(3^6)$; thus there are 27 combinations in the DOE. Each parameter combination was tested by 45 instance problems, 15 instances form small-sized problems (25 activity), 15 instances form medium-sized problems (50 activity) and the remainder instances are from the larger-sized problems (100 activity). This experiment has conducted on the most complex problem instances considering OS, RC, and RU. The orthogonal array and the responsive mean NPV values for each operation are presented in Table 3.

The factor level trend of each parameter for IGA-SM and IGA-IM are presented in Fig. 11 and Fig. 12, respectively. Table 4 presents the best combination of parameters for the proposed algorithms. After DOE, for IGA-SM, the optimum values of these factors are (based on the maximum values of mean NPV): $PS = 20$; $P_c = 0.7$; $P_m = 0.15$; $P_i = 0.8$, $R\_counter = 10$ and $FBI\_counter = 10$. On the other hand, for IGA-IM, the best combination of the parameters is: $PS = 20$; $P_c = 0.8$; $P_m = 0.1$; $P_i = 0.8$, $R\_counter = 5$ and $FBI\_counter = 20$.

The responses for the mean values of different parameters for IGA-SM and IGA-IM are presented in Table 5 and Table 6, respectively. Here, for each parameter, delta is the difference between the maximum and minimum mean response values. A larger delta value indicates a higher influence of the factor on the solution. Thus, the factor with the

TABLE 3. Orthogonal table and responsive mean NPV.

Experiment No.	Parameters						Mean NPV	
	PS	*Pc*	*Pm*	*Pi*	*R_counter*	*FBI_counter*	IGA-SM	IGA-IM
1	20	0.6	0.1	0.7	5	10	629.16	634.60
2	20	0.6	0.1	0.7	10	15	636.43	623.40
3	20	0.6	0.1	0.7	15	20	612.16	625.72
4	20	07	0.15	0.8	5	10	636.17	634.51
5	20	0.7	0.15	0.8	10	15	648.24	627.05
6	20	0.7	0.15	0.8	15	20	622.91	628.32
7	20	0.8	0.2	0.9	5	10	633.63	629.23
8	20	0.8	0.2	0.9	10	15	626.43	603.11
9	20	0.8	0.2	0.9	15	20	622.75	610.87
10	40	0.6	0.15	0.9	5	15	616.67	607.74
11	40	0.6	0.15	0.9	10	20	626.55	626.04
12	40	0.6	0.15	0.9	15	10	609.46	599.84
13	40	07	0.2	0.7	5	15	616.37	613.51
14	40	0.7	0.2	0.7	10	20	622.58	615.80
15	40	0.7	0.2	0.7	15	10	629.96	619.48
16	40	0.8	0.1	0.8	5	15	620.85	621.42
17	40	0.8	0.1	0.8	10	20	626.36	635.12
18	40	0.8	0.1	0.8	15	10	617.69	625.07
19	50	0.6	0.2	0.8	5	20	614.01	619.28
20	50	0.6	0.2	0.8	10	10	613.26	603.42
21	50	0.6	0.2	0.8	15	15	603.61	609.32
22	50	07	0.1	0.9	5	20	617.63	605.41
23	50	0.7	0.1	0.9	10	10	604.76	619.74
24	50	0.7	0.1	0.9	15	15	604.97	615.24
25	50	0.8	0.15	0.7	5	20	620.12	629.34
26	50	0.8	0.15	0.7	10	10	615.35	621.83
27	50	0.8	0.15	0.7	15	15	609.67	611.18

TABLE 4. The best combination of parameters.

Algorithm	*PS*	*Pc*	*Pm*	*Pi*	*R_counter*	*FBI_counter*
IGA-SM	20	0.7	0.15	0.8	10	10
IGA-IM	20	0.8	0.1	0.8	5	20

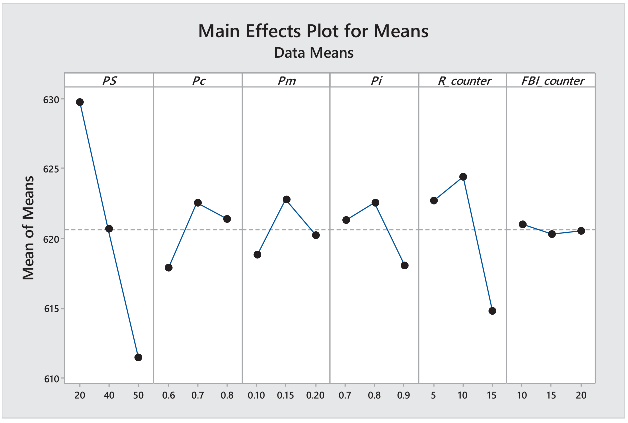


FIGURE 11. Factor level trend for IGA-SM.

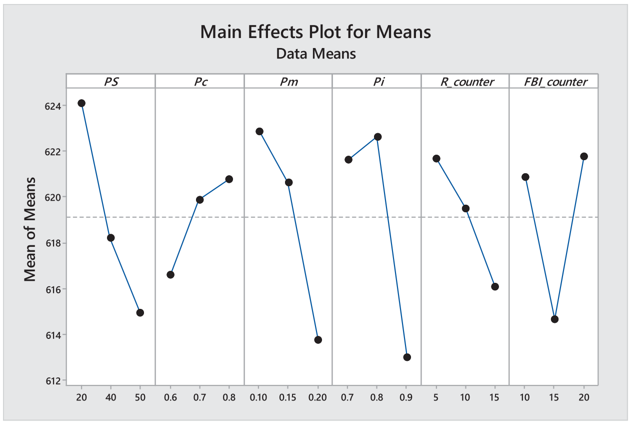


FIGURE 12. Factor level trend for IGA-IM.

largest delta value is the most influential factor. The most significant factor for IGA-SM is PS; followed by $R\_counter$, P_c, P_i, P_m and $FBI\_counter$. For IGA-IM, on the other hand, P_i is the most influential factor; followed by PS, P_m, $FBI\_counter$, $R\_counter$ and P_c.

C. COMPARISON OF RESULTS WITH STATE-OF-ART ALGORITHMS

To make a fair comparison with state-of-the-art algorithms, our proposed algorithms terminate after 5,000 schedule generations. The results are analyzed based on the relative average deviation (%AvDev) and the percentage of deadline feasible solutions (%Feas). The %AvDev is defined as the deviation from the best-known resource unconstrained NPV-based solutions (NPV_{opt}), described in Vanhoucke [96]. Therefore, the %AvDev for the result obtained form the IGA (NPV_{IGA}) is calculated as:

$$\%AvDev = |\frac{NPV_{opt} - NPV_{IGA}}{NPV_{opt}}| * 100\% \tag{9}$$

To calculate the %AvDev, instances with deadline feasible solutions are only considered here. A solution with the lower %AvDev value is a better solution. Note that, the value of %AvDev may be greater than 100%. For instance,

TABLE 5. Response table for mean values of parameters for IGA-SM.

Level	*PS*	*Pc*	*Pm*	*Pi*	*R_counter*	*FBI_counter*
1	629.80	617.90	618.90	621.30	622.70	621.00
2	620.70	622.60	622.80	622.60	624.40	620.40
3	611.50	621.40	620.30	618.10	614.80	620.60
Delta	18.30	4.70	3.90	4.50	9.60	0.70
Rank	1	3	5	4	2	6

TABLE 6. Response table for mean values of parameters for IGA-IM.

Level	*PS*	*Pc*	*Pm*	*Pi*	*R_counter*	*FBI_counter*
1	624.10	616.60	622.90	621.70	621.70	620.90
2	618.20	619.90	620.70	622.60	619.50	614.70
3	615.00	620.80	613.80	613.00	616.10	621.80
Delta	9.10	4.20	9.10	9.60	5.60	7.10
Rank	2	6	3	1	5	4

TABLE 7. Comparative result based on the %Neg.

Benchmark parameter	5000 Schedule								12500 Schedule	
%Neg	IGA-SM		IGA-IM		Leyman and Vanhoucke [10]		Vanhoucke [17]		Gu et al. [12]	
	%AvDev	%Feas	%AvDev	%Feas	%AvDev	%Feas	%AvDev	%Feas	%AvDev	%Feas
0%	**30.30**	98.82	**30.27**	98.89	30.84	98.85	31.16	99.79	30.64	99.90
20%	**27.57**	98.89	**27.57**	98.68	28.59	99.34	29.17	99.79	28.42	99.90
40%	**68.26**	98.26	**68.33**	98.16	71.21	98.85	69.78	99.79	67.03	99.90
60%	**476.25**	98.13	**478.72**	98.13	487.85	97.40	482.66	99.79	460.68	99.93
80%	442.16	98.06	440.42	98.06	415.22	98.33	442.50	99.79	422.21	99.93
100%	**237.40**	98.30	**235.81**	98.23	240.45	98.44	252.93	99.79	247.07	99.93

TABLE 8. Comparative result based the D-Incr.

Benchmark parameter	5000 Schedule								12500 Schedule	
D-Incr	IGA-SM		IGA-IM		Leyman and Vanhoucke [10]		Vanhoucke [17]		Gu et al. [12]	
	%AvDev	%Feas	%AvDev	%Feas	%AvDev	%Feas	%AvDev	%Feas	%AvDev	%Feas
5%	235.27	93.63	234.67	93.43	228.94	94.21	234.86	99.17	222.64	99.65
10%	146.16	100.00	146.19	100.00	133.91	99.93	152.41	100.00	142.94	100.00
15%	**324.63**	100.00	**325.48**	100.00	337.93	100.00	332.34	100.00	323.67	100.00
20%	147.93	100.00	147.39	100.00	145.83	100.00	149.71	100.00	146.57	100.00

TABLE 9. Comparative result based on the number of activities.

Benchmark parameter	5000 Schedule								12500 Schedule	
Activity	IGA-SM		IGA-IM		Leyman and Vanhoucke [10]		Vanhoucke [17]		Gu et al. [12]	
	%AvDev	%Feas	%AvDev	%Feas	%AvDev	%Feas	%AvDev	%Feas	%AvDev	%Feas
25	200.52	100.00	200.23	100.00	192.59	99.51	192.46	100.00	190.05	100.00
50	**287.30**	99.61	**288.31**	99.63	294.05	99.03	291.75	99.86	285.36	100.00
75	**153.57**	97.62	**152.80**	97.62	153.97	98.06	160.30	99.72	153.51	99.98
100	209.95	96.42	209.68	96.18	204.38	97.55	223.42	99.58	208.08	99.68

suppose the NPV_{opt} of a project with 60% negative cash flow associated activities is 2 and the NPV_{IGA} is -1, thus the %AvDev is 150%. The results of this study are compared with the state-of-art algorithms. To the best of our knowledge, the GA [10], the SS [17] and the LR-based algorithm [12] are the existing state-of-the-art algorithms for the RCPSPDC, focusing on the available benchmark dataset introduced by Vanhoucke [17].

Tables 7–12 presents the comparative computational results in terms of %Neg, D-Incr, activity number, OS, RC and RU, respectively. Here, the blue color indicates that the proposed algorithms outperforms the state-of-the-art algorithms in comparison with Leyman and Vanhoucke [10], Vanhoucke [17] based on 5,000 schedule generations. On the other hand, the red color indicates that the results outperform the existing algorithm based on 12,500 schedules generated by Gu *et al.* [12]. Interested readers may download the results of this study at the following link for further research: https://research.unsw.edu.au/projects/cross-disciplinary-optimisation-under-capability-context.

Based on the results shown in Tables 7-12, following findings can be summarized:

1) For both 0% and 100% cash flows, results obtained from both the IGA-SM and IGA-IM outperform all the state-of-the-art algorithms, even if the termination criteria is 5,000 or 12,500 schedules generation. Therefore, for 0% and 100% negative cash flow activity instances, both the IGA-SM and IGA-IM are better

TABLE 10. Comparative result based on OS.

Benchmark parameter	5000 Schedule								12500 Schedule	
OS	IGA-SM		IGA-IM		Leyman and Vanhoucke [10]		Vanhoucke [17]		Gu et al. [12]	
	%AvDev	%Feas	%AvDev	%Feas	%AvDev	%Feas	%AvDev	%Feas	%AvDev	%Feas
0.25	**163.36**	97.80	**162.88**	97.67	166.00	98.00	179.67	99.69	161.09	99.91
0.50	211.36	97.80	210.65	97.73	203.56	97.95	210.55	99.69	207.53	99.83
0.75	263.77	99.64	264.73	99.58	263.71	99.65	260.23	100.00	257.26	100.00

TABLE 11. Comparative result based on RC.

Benchmark parameter	5000 Schedule								12500 Schedule	
RC	IGA-SM		IGA-IM		Leyman and Vanhoucke [10]		Vanhoucke [17]		Gu et al. [12]	
	%AvDev	%Feas	%AvDev	%Feas	%AvDev	%Feas	%AvDev	%Feas	%AvDev	%Feas
0.25	**81.30**	97.38	**81.81**	97.27	342.07	98.13	342.11	99.58	65.50	100.00
0.50	331.78	97.86	331.78	97.81	141.25	98.75	149.67	99.90	338.33	99.74
0.75	225.44	99.98	224.66	99.98	151.81	98.73	160.33	99.90	224.44	100.00

TABLE 12. Comparative result based on RU.

Benchmark parameter	5000 Schedule								12500 Schedule	
RU	IGA-SM		IGA-IM		Leyman and Vanhoucke [10]		Vanhoucke [17]		Gu et al. [12]	
	%AvDev	%Feas	%AvDev	%Feas	%AvDev	%Feas	%AvDev	%Feas	%AvDev	%Feas
2	**246.23**	98.16	**246.34**	98.10	251.76	98.38	252.45	99.79	248.18	99.83
4	180.23	98.66	179.99	98.61	171.18	98.69	181.82	99.79	169.18	100.00

than the methods of Leyman and Vanhoucke [10], Vanhoucke [17], and Gu *et al.* [12].

2) For 20%, 40%, and 60% negative cash flows, IGA-SM and IGA-IM outperform the existing algorithms based on 5,000 schedule generations. Therefore, the proposed IGAs are more effective algorithms based on the %Neg except 80% in comparison to the results of Leyman and Vanhoucke [10], and Vanhoucke [17].
3) In terms of D-Incr, our proposed algorithms outperform for 15% D-Incr and 5,000 schedule generations, however for others D-Incr are also competitive.
4) Both the IGA-SM and IGA-IM outperforms the results of Leyman and Vanhoucke [10], and Vanhoucke [17] for 50 activity instances. For 75 activity instances the proposed IGA-IM outperforms existing all state-of-the-art algorithms, whereas IGA-SM outperforms algorithms based on 5,000 schedule generations. A similar trend can be observed as the activity number increases.
5) In case of both OS and RC value of 0.25, the proposed algorithms obtained outperforming results comparing results after 5,000 schedule generations.
6) Based on RU, both IGA-SM and IGA-IM are the best performing algorithm for 2-RU, as it generates the least %AvDev in comparison to both 5,000 and 12,500 schedule generations. Therefore, it competes favourably with all the results of Leyman and Vanhoucke [10], Vanhoucke [17], and Gu *et al.* [12].
7) In comparison based on the %Feas, the LR-based algorithm developed by Gu *et al.* [12] outperforms other algorithms, however the results of IGA-SM and IGA-IM are competitive.

The AMR is used to shift the activities with negative cash flow. Hence, for the 100% negative cash flow associated projects, all the activities is moved to the left. As a result, once the schedule starts there is no idle time in the schedule. Nonetheless, when the activities are associated with both positive and negative cash flows, only the activities with negative cash flow can be shift to the left of the schedule. Such movements may induce idle time(s) in the project schedule. As a consequence, AMR increases the project's NPV (i.e. fitness), however, not much in comparison to the projects associated with 100% negative cash flows. In contrast, AMR is not applied to 100% positive cash flow associated activity projects. The schedule is therefore generated in such a way that the activities are performed without delays with shifting left as early as possible. Thus, the 0% and 100% negative cash flow associated projects achieve higher NPV than the others.

In this study, IGA-SM generates 98.41% deadline feasible solution with a %AvDev of 213.15, whereas IGA-IM generates 98.36% deadline feasible solution with a %AvDev of 213.08. Thus, IGA-IM performs better in comparison with IGA-SM based on the lower value of %AvDev, however inferior in terms of generating the higher numbers of deadline feasible solutions.

Before analyzing the difference between IGA-SM and IGA-IM by any parametric or non-parametric test, the normality test of the data is important. If the data is normally distributed, then parametric test (e.g.; *t*-test) is conducted; otherwise, the non-parametric test like the Wilcoxon signed rank test [97] is examined. In this study, the normality of the measured NPV values of the test instances is assessed using the Kolmogorov-Smirnov test [98]. The Kolmogorov-Smirnov test (Table 13) showed that the NPV values of the instances for both IGA-SM and IGA-IM are not exhibited normal distributions ($p < 0.05$). Therefore, to analyze the difference between IGA-SM and IGA-IM,

TABLE 13. Normality test of the resulted NPV values of the test instances using Kolmogorov-Smirnov test.

Algorithm	Statistic	p-value	Decision
IGA-SM	0.064	0.000	NN
IGA-IM	0.064	0.000	NN

NN: Non-normal ($p < 0.05$)

TABLE 14. Wilcoxon signed rank test between IGA-SM and IGA-IM.

Activity set	Mean rank		p	Decision
	IGA-SM	IGA-IM		
25	319.89	333.79	0.328	NS
50	1046.15	1076.67	0.000	S
75	1365.83	1392.49	0.000	S
100	1467.59	1777.10	0.000	S

S: Significant ($p < 0.05$); NS: Not significant ($p > 0.05$)

the Wilcoxon signed rank test is carried out based on the NPV values. Table 14 shows the Wilcoxon signed rank test based on the NPV values of deadline feasible solutions. There is no statistically significant ($p < 0.05$) difference between the IGA-SM and IGA-IM for 25 activity instance set. However, the IGA-IM is significantly ($p > 0.05$) better than the IGA-SM for 50, 75 and 100 activity projects.

Figures 13–14 present box plot for 25, 50, 75, and 100 activities using the IGA-SM and IGA-IM, respectively based on the NPV values of deadline feasible solutions. The range of the data is presented by the end of the wishkers. The upper and lower edges of the box represent the first and third quartile respectively. The line crossing the box indicates the median, the central value of the data.

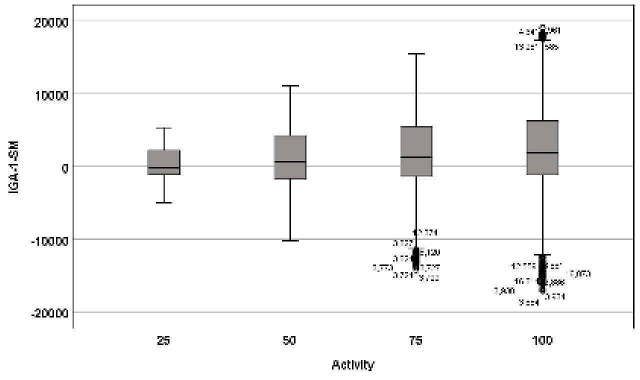


FIGURE 13. Box plot of different activity instances using IGA-SM.

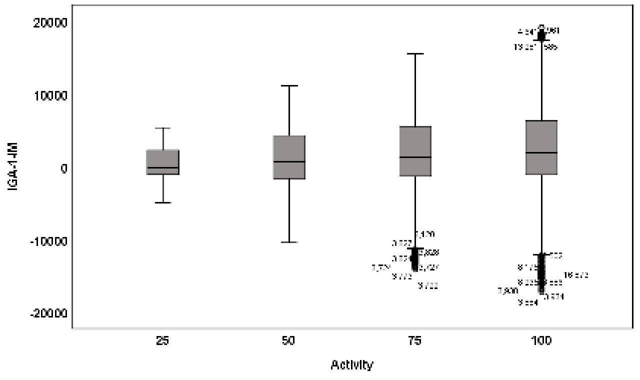


FIGURE 14. Box plot of different activity instances using IGA-IM.

Table 15 shows the average computational time (in seconds) to solve the different size instances. For each size instances, 25 to 100 activities, IGA-SM takes more time than the IGA-IM. This is because of the different combinations of the parameters, specially for the FBI_counter as it applies after 10 generations for the IGA-SM whereas after 20 generations for the IGA-IM. Therefore, in terms of the computational time IGA-IM is better than the IGA-SM. From Table 15, it is seen that, the average computational time increases exponentially with the increase of the activities number that represents the NP-hard characteristics of the RCPSPDC.

TABLE 15. Average computational time.

Activity	Time (seconds)	
	IGA-SM	IGA-IM
25	17.91	16.17
50	88.59	82.55
75	286.93	277.85
100	753.61	687.31

VI. CONCLUSION

Owing to the nature of the complexity of the RCPSPDC, over the last few decades, no single algorithm has been shown to perform better with a reasonably good solution for all the problem instances of the datasets. For this reason, in this study, two different types of IGAs have examined for solving the RCPSPDC. Firstly, the RCPSP with the target of maximizing the NPV is established and consequently the hybrid IGA is applied to solve the problem. Hybridizing the IA with GA increases the search ability to find the nearly optimal solution and avoids the issue of premature convergence. The VINS in IGA enhances the performance of the population's worst individual. In addition, the restart scheme induce diversity and prevents trapping from local solution. Moreover, the AMR delays the negative cash flow associated activities which further improves the NPV.

The parameters of the algorithms are set using the well known Taguchi's DOE. The performance of the proposed algorithms are verified by testing on a standard dataset problems from 25 to 100 activity projects. Consequently, the proposed algorithms have been compared with the state-of-the-art algorithms to examine their performance. The study revealed that, the proposed algorithms are consistently more effective than the existing algorithms with a lower value of average deviation. The proposed algorithm outperforms existing state-of-the-art algorithms, particularly for (i) the projects with 0% and 100% negative cash flow associated activities, (ii) the 75-activity instances, and (iii) the projects with two resources usage. The proposed techniques can be utilized in a range of financial sectors such as software, manufacturing, service, and construction industries. The algorithms developed in this study can aid practitioners and project managers in making the right decision at the right time in executing the project. These will help project managers to more carefully estimate the potential budget in order to ensure profitability.

The proposed algorithms are also applicable to solve resource constrained portfolio optimization and revenue management problems. However, the proposed IGA is problem specific whose performance is dependent on different parameter combinations. Thus, for any further extension of the problem, the algorithm should be redesigned. In this study, the cash inflows comes from the payments at activity's completion time. However, cash flow may occur with the progress of the work at a regular time interval. In the future, a hybrid approach of the IGA will be tested on RCPSPDC with different payment methods. It is known that, owing to the no free lunch theorem, no single heuristic or meta-heuristic algorithm with a reasonably good solution and within a reasonable computational time is suitable for solving the RCPSPDC [21]. Therefore, in the future, the performance examination of other swarm intelligence algorithms such as animal migration optimization [99], water wave optimization [100], or their hybridization with other heuristics would be valuable to solve the RCPSPDC. Moreover, the study of multi-mode RCPSPDC with uncertainty and disruption would be more interesting. Furthermore, an integration of supply chain management with the RCPSPDC to ensure the right product at the right place at the right time with a reasonable cost would be helpful to assure the profitability of the project.

REFERENCES

[1] W. Herroelen, "Project scheduling-theory and practice," *Prod. Oper. Manage.*, vol. 14, no. 4, pp. 413–432, Jan. 2009.

[2] H.-W. Wang, J.-R. Lin, and J.-P. Zhang, "Work package-based information modeling for resource-constrained scheduling of construction projects," *Autom. Construct.*, vol. 109, Jan. 2020, Art. no. 102958.

[3] R. K. Chakrabortty, H. F. Rahman, and M. J. Ryan, "Efficient priority rules for project scheduling under dynamic environments: A heuristic approach," *Comput. Ind. Eng.*, vol. 140, Feb. 2020, Art. no. 106287.

[4] Y. Alipouri, M. H. Sebt, A. Ardeshir, and M. H. F. Zarandi, "A mixed-integer linear programming model for solving fuzzy stochastic resource constrained project scheduling problem," *Oper. Res.*, vol. 20, no. 1, pp. 197–217, 2020.

[5] Y. Liang, N. Cui, T. Wang, and E. Demeulemeester, "Robust resource-constrained max-NPV project scheduling with stochastic activity duration," *OR Spectr.*, vol. 41, no. 1, pp. 219–254, Mar. 2019.

[6] V. S. Bilolikar, K. Jain, and M. Sharma, "An adaptive crossover genetic algorithm with simulated annealing for multi mode resource constrained project scheduling with discounted cash flows," *Int. J. Oper. Res.*, vol. 25, no. 1, pp. 28–46, 2016.

[7] C. Zhao, H. Ke, and Z. Chen, "Uncertain resource-constrained project scheduling problem with net present value criterion," *J. Uncertainty Anal. Appl.*, vol. 4, no. 1, p. 12, Dec. 2016.

[8] G. Ulusoy and L. Özdamar, "A heuristic scheduling algorithm for improving the duration and net present value of a project," *Int. J. Oper. Prod. Manage.*, vol. 15, no. 1, pp. 89–98, Jan. 1995.

[9] A. Schnabel, C. Kellenbrink, and S. Helber, "Profit-oriented scheduling of resource-constrained projects with flexible capacity constraints," *Bus. Res.*, vol. 11, no. 2, pp. 329–356, Sep. 2018.

[10] P. Leyman and M. Vanhoucke, "A new scheduling technique for the resource–constrained project scheduling problem with discounted cash flows," *Int. J. Prod. Res.*, vol. 53, no. 9, pp. 2771–2786, 2015.

[11] H. Gu, P. J. Stuckey, and M. G. Wallace, "Maximising the net present value of large resource-constrained projects," in *Proc. Int. Conf. Princ. Pract. Constraint Program.* Berlin, Germany: Springer, 2012, pp. 767–781.

[12] H. Gu, A. Schutt, and P. J. Stuckey, "A Lagrangian relaxation based forward-backward improvement heuristic for maximising the net present value of resource-constrained projects," in *Proc. Int. Conf. AI OR Techn. Constriant Program. Combinat. Optim. Problems*. Berlin, Germany: Springer, 2013, pp. 340–346.

[13] H. Gu, A. Schutt, P. J. Stuckey, M. G. Wallace, and G. Chu, "Exact and heuristic methods for the resource-constrained net present value problem," in *Handbook on Project Management and Scheduling*, vol. 1. Cham, Switzerland: Springer, 2015, pp. 299–318.

[14] M. Mika, G. Waligóra, and J. Węglarz, "Simulated annealing and tabu search for multi-mode resource-constrained project scheduling with positive discounted cash flows and different payment models," *Eur. J. Oper. Res.*, vol. 164, no. 3, pp. 639–668, Aug. 2005.

[15] G. Waligóra, "Simulated annealing and tabu search for discrete-continuous project scheduling with discounted cash flows," *RAIRO Oper. Res.*, vol. 48, no. 1, pp. 1–24, Jan. 2014.

[16] O. Icmeli and S. S. Erenguc, "A tabu search procedure for the resource constrained project scheduling problem with discounted cash flows," *Comput. Oper. Res.*, vol. 21, no. 8, pp. 841–853, 1994.

[17] M. Vanhoucke, "A scatter search heuristic for maximising the net present value of a resource-constrained project with fixed activity cash flows," *Int. J. Prod. Res.*, vol. 48, no. 7, pp. 1983–2001, Apr. 2010.

[18] A. H. L. Chen and C.-C. Chyu, "A memetic algorithm for maximizing net present value in resource-constrained project scheduling problem," in *Proc. IEEE Congr. Evol. Comput. (IEEE World Congr. Comput. Intell.)*, Jun. 2008, pp. 2396–2403.

[19] Y.-Y. Shou, "Ant colony algorithm for scheduling resource constrained projects with discounted cash flows," in *Proc. Int. Conf. Mach. Learn. Cybern.*, 2006, pp. 176–180.

[20] W.-N. Chen, J. Zhang, H. S.-H. Chung, R.-Z. Huang, and O. Liu, "Optimizing discounted cash flows in project scheduling—An ant colony optimization approach," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 40, no. 1, pp. 64–77, Jan. 2010.

[21] H. F. Rahman, R. K. Chakrabortty, and M. J. Ryan, "Memetic algorithm for solving resource constrained project scheduling problems," *Autom. Construct.*, vol. 111, Mar. 2020, Art. no. 103052.

[22] X. Li and M. Yin, "A discrete artificial bee colony algorithm with composite mutation strategies for permutation flow shop scheduling problem," *Scientia Iranica*, vol. 19, no. 6, pp. 1921–1935, Dec. 2012.

[23] X. Li and M. Yin, "A hybrid cuckoo search via Lévy flights for the permutation flow shop scheduling problem," *Int. J. Prod. Res.*, vol. 51, no. 16, pp. 4732–4754, 2013.

[24] C.-W. Tsai and J. J. P. C. Rodrigues, "Metaheuristic scheduling for cloud: A survey," *IEEE Syst. J.*, vol. 8, no. 1, pp. 279–291, Mar. 2014.

[25] Y. Wang, X. Li, and Z. Ma, "A hybrid local search algorithm for the sequence dependent setup times flowshop scheduling problem with makespan criterion," *Sustainability*, vol. 9, no. 12, p. 2318, Dec. 2017.

[26] R. Sindhu, R. Ngadiran, Y. M. Yacob, N. A. H. Zahri, M. Hariharan, and K. Polat, "A hybrid SCA inspired BBO for feature selection problems," *Math. Problems Eng.*, vol. 2019, pp. 1–18, Apr. 2019.

[27] H. F. Rahman, M. N. Janardhanan, and I. E. Nielsen, "Real-time order acceptance and scheduling problems in a flow shop environment using hybrid GA-PSO algorithm," *IEEE Access*, vol. 7, pp. 112742–112755, 2019.

[28] X. Li, X. Zhang, M. Yin, and J. Wang, "A genetic algorithm for the distributed assembly permutation flowshop scheduling problem," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, May 2015, pp. 3096–3101.

[29] M. Bessedik, F. Benbouzid-Si Tayeb, H. Cheurfi, and A. Blizak, "An immunity-based hybrid genetic algorithms for permutation flowshop scheduling problems," *Int. J. Adv. Manuf. Technol.*, vol. 85, nos. 9–12, pp. 2459–2469, Aug. 2016.

[30] S. U. Kadam and N. S. Kadam, "Solving resource-constrained project scheduling problem by genetic algorithm," in *Proc. 2nd Int. Conf. Bus. Inf. Manage. (ICBIM)*, Jan. 2014, pp. 159–164.

[31] J. Liu, Y. Liu, Y. Shi, and J. Li, "Solving resource-constrained project scheduling problem via genetic algorithm," *J. Comput. Civil Eng.*, vol. 34, no. 2, Mar. 2020, Art. no. 04019055.

[32] H. F. Rahman, R. Sarker, and D. Essam, "A real-time order acceptance and scheduling approach for permutation flow shop problems," *Eur. J. Oper. Res.*, vol. 247, no. 2, pp. 488–503, Dec. 2015.
[33] M. A. Salido, J. Escamilla, A. Giret, and F. Barber, "A genetic algorithm for energy-efficiency in job-shop scheduling," *Int. J. Adv. Manuf. Technol.*, vol. 85, nos. 5–8, pp. 1303–1314, 2016.
[34] H. F. Rahman, R. Sarker, and D. Essam, "A genetic algorithm for permutation flow shop scheduling under make to stock production system," *Comput. Ind. Eng.*, vol. 90, pp. 12–24, Dec. 2015.
[35] L. Jiao and L. Wang, "A novel genetic algorithm based on immunity," *IEEE Trans. Syst., Man, Cybern. A, Syst. Humans*, vol. 30, no. 5, pp. 552–561, Sep. 2000.
[36] R. Zhang, T. Li, X. Xiao, and Y. Shi, "A danger-theory-based immune network optimization algorithm," *Sci. World J.*, vol. 2013, pp. 1–13, Jan. 2013.
[37] D. Dasgupta, "Parallel search for multi-modal function optimization with diversity and learning of immune algorithm," in *Artificial Immune Systems and Their Applications*. Berlin, Germany: Springer, 1999, pp. 210–220.
[38] H. Bersini, "The immune recruitment mechanism: A selective evolutionary strategy," in *Proc. 4th Int. Conf. Genetic Algorithms*. San Mateo, CA, USA: Morgan Kaufmann, 1991, pp. 520–526.
[39] H.-Y. Zhang, "An improved immune algorithm for simple assembly line balancing problem of type 1," *J. Algorithms Comput. Technol.*, vol. 11, no. 4, pp. 317–326, Dec. 2017.
[40] H. Zhang, "An immune genetic algorithm for simple assembly line balancing problem of type 1," *Assem. Automat.*, vol. 39, no. 1, pp. 113–123, Feb. 2019.
[41] H. F. Rahman, M. N. Janardhanan, and P. Nielsen, "An integrated approach for line balancing and AGV scheduling towards smart assembly systems," *Assem. Automat.*, vol. 40, no. 2, pp. 219–234, Jan. 2020.
[42] X.-D. Xu and C.-X. Li, "Research on immune genetic algorithm for solving the job-shop scheduling problem," *Int. J. Adv. Manuf. Technol.*, vol. 34, nos. 7–8, pp. 783–789, 2007.
[43] H. Ren, H. Xu, and S. Sun, "Immune genetic algorithm for multi-objective flexible job-shop scheduling problem," in *Proc. Chin. Control Decis. Conf. (CCDC)*, May 2016, pp. 2167–2171.
[44] T. Ghosh, B. Doloi, and P. K. Dan, "An immune genetic algorithm for inter-cell layout problem in cellular manufacturing system," *Prod. Eng.*, vol. 10, no. 2, pp. 157–174, Apr. 2016.
[45] J.-S. Chou, C.-F. Tsai, Z.-Y. Chen, and M.-H. Sun, "Biological-based genetic algorithms for optimized disaster response resource allocation," *Comput. Ind. Eng.*, vol. 74, pp. 52–67, Aug. 2014.
[46] A. Russell, "Cash flows in networks," *Manage. Sci.*, vol. 16, no. 5, pp. 357–373, 1970.
[47] R. C. Grinold, "The payment scheduling problem," *Nav. Res. Logistics Quart.*, vol. 19, no. 1, pp. 123–136, Mar. 1972.
[48] K. Neumann and J. Zimmermann, "Procedures for resource leveling and net present value problems in project scheduling with general temporal and resource constraints," *Eur. J. Oper. Res.*, vol. 127, no. 2, pp. 425–443, Dec. 2000.
[49] W. S. Herroelen, P. Van Dommelen, and E. L. Demeulemeester, "Project network models with discounted cash flows a guided tour through recent developments," *Eur. J. Oper. Res.*, vol. 100, no. 1, pp. 97–121, Jul. 1997.
[50] C. Schwindt and J. Zimmermann, "A steepest ascent approach to maximizing the net present value of projects," *Math. Methods Oper. Res.*, vol. 53, no. 3, pp. 435–450, Jul. 2001.
[51] E. Demeulemeester and W. Herroelen, "A branch-and-bound procedure for the multiple resource-constrained project scheduling problem," *Manage. Sci.*, vol. 38, no. 12, pp. 1803–1818, Dec. 1992.
[52] O. Icmeli and S. S. Erenguc, "A branch and bound procedure for the resource constrained project scheduling problem with discounted cash flows," *Manage. Sci.*, vol. 42, no. 10, pp. 1395–1408, Oct. 1996.
[53] M. Vanhoucke, E. Demeulemeester, and W. Herroelen, "On maximizing the net present value of a project under renewable resource constraints," *Manage. Sci.*, vol. 47, no. 8, pp. 1113–1121, Aug. 2001.
[54] R. H. Doersch and J. H. Patterson, "Scheduling a project to maximize its present value: A zero-one programming approach," *Manage. Sci.*, vol. 23, no. 8, pp. 882–889, Apr. 1977.
[55] A. Schutt, G. Chu, P. J. Stuckey, and M. G. Wallace, "Maximising the net present value for resource-constrained project scheduling," in *Proc. Int. Conf. Integr. Artif. Intell. (AI) Oper. Res. (OR) Techn. Constraint Program.* Berlin, Germany: Springer, 2012, pp. 362–378.
[56] R. A. Russell, "A comparison of heuristics for scheduling projects with cash flows and resource restrictions," *Manage. Sci.*, vol. 32, no. 10, pp. 1291–1300, Oct. 1986.
[57] D. E. Smith-Daniels and N. J. Aquilano, "Using a late-start resource-constrained project schedule to improve project net present value," *Decis. Sci.*, vol. 18, no. 4, pp. 617–630, 1987.
[58] S. M. Baroum and J. H. Patterson, "The development of cash flow weight procedures for maximizing the net present value of a project," *J. Oper. Manage.*, vol. 14, no. 3, pp. 209–227, Sep. 1996.
[59] K. K. Yang, L. C. Tay, and C. C. Sum, "A comparison of stochastic scheduling rules for maximizing project net present value," *Eur. J. Oper. Res.*, vol. 85, no. 2, pp. 327–339, Sep. 1995.
[60] J. P. Pinder and A. S. Marucheck, "Using discounted cash flow heuristics to improve project net present valve," *J. Operations Manage.*, vol. 14, no. 3, pp. 229–240, Sep. 1996.
[61] R. Padman, D. E. Smith-Daniels, and V. L. Smith-Daniels, "Heuristic scheduling of resource-constrained projects with cash flows," *Nav. Res. Logistics*, vol. 44, no. 4, pp. 365–381, Jun. 1997.
[62] G. Waligóra and R. Ró ycki, "Heuristic solving some discrete-continuous project scheduling problems with discounted cash flows," in *Proc. 21st Int. Conf. Methods Models Autom. Robot. (MMAR)*, Aug. 2016, pp. 971–974.
[63] J. H. Patterson, "A comparison of exact approaches for solving the multiple constrained resource, project scheduling problem," *Manage. Sci.*, vol. 30, no. 7, pp. 854–867, Jul. 1984.
[64] D. Thiruvady, M. Wallace, H. Gu, and A. Schutt, "A lagrangian relaxation and ACO hybrid for resource constrained project scheduling with discounted cash flows," *J. Heuristics*, vol. 20, no. 6, pp. 643–676, Dec. 2014.
[65] P.-C. Chang, W.-H. Huang, C.-J. Ting, L.-C. Wu, and C.-M. Lai, "A hybrid genetic-immune algorithm with improved offsprings and elitist antigen for flow-shop scheduling problems," in *Proc. 11th IEEE Int. Conf. High Perform. Comput. Commun.*, 2009, pp. 591–596.
[66] J. H. Holland, "Genetic algorithms and adaptation," in *Adaptive Control of Ill-Defined Systems*. Boston, MA, USA: Springer, 1984, pp. 317–333.
[67] D. E. Goldberg, *Genetic Algorithms*. New Delhi, India: Pearson, 2006.
[68] D. Alisantoso, L. P. Khoo, and P. Y. Jiang, "An immune algorithm approach to the scheduling of a flexible PCB flow shop," *Int. J. Adv. Manuf. Technol.*, vol. 22, nos. 11–12, pp. 819–827, Dec. 2003.
[69] M. Zandieh and M. Gholami, "An immune algorithm for scheduling a hybrid flow shop with sequence-dependent setup times and machines with random breakdowns," *Int. J. Prod. Res.*, vol. 47, no. 24, pp. 6999–7027, Dec. 2009.
[70] Y. Wang, X. Geng, F. Zhang, and J. Ruan, "An immune genetic algorithm for multi-echelon inventory cost control of IOT based supply chains," *IEEE Access*, vol. 6, pp. 8547–8555, 2018.
[71] C. A. C. Coello and N. C. Cortés, "Hybridizing a genetic algorithm with an artificial immune system for global optimization," *Eng. Optim.*, vol. 36, no. 5, pp. 607–634, Oct. 2004.
[72] D. Wang, R. Y. K. Fung, and W. H. Ip, "An immune-genetic algorithm for introduction planning of new products," *Comput. Ind. Eng.*, vol. 56, no. 3, pp. 902–917, Apr. 2009.
[73] M. Suresh, P. Dutta, and K. Jain, "Resource constrained multi-project scheduling problem with resource transfer times," *Asia–Pacific J. Oper. Res.*, vol. 32, no. 06, Dec. 2015, Art. no. 1550048.
[74] J. H. Patterson, "Alternate methods of project scheduling with limited resources," *Nav. Res. Logistics Quart.*, vol. 20, no. 4, pp. 767–784, Dec. 1973.
[75] S. Hartmann, "A competitive genetic algorithm for resource-constrained project scheduling," *Nav. Res. Logistics*, vol. 45, no. 7, pp. 733–750, Oct. 1998.
[76] R. Kolisch, "Efficient priority rules for the resource-constrained project scheduling problem," *J. Oper. Manage.*, vol. 14, no. 3, pp. 179–192, Sep. 1996.
[77] O. Icmeli and S. S. Erenguc, "The resource constrained time/cost tradeoff project scheduling problem with discounted cash flows," *J. Oper. Manage.*, vol. 14, no. 3, pp. 255–275, Sep. 1996.
[78] S. Hartmann and R. Kolisch, "Heuristic algorithms for the resource-constrained project scheduling problem: Classification and computational analysis," in *Project Scheduling*. Boston, MA, USA: Springer, 1999, pp. 147–178.
[79] R. Kolisch, "Serial and parallel resource-constrained project scheduling methods revisited: Theory and computation," *Eur. J. Oper. Res.*, vol. 90, no. 2, pp. 320–333, Apr. 1996.
[80] K. Y. Li and R. J. Willis, "An iterative scheduling technique for resource-constrained project scheduling," *Eur. J. Oper. Res.*, vol. 56, no. 3, pp. 370–379, Feb. 1992.

[81] H. F. Rahman and I. Nielsen, "Scheduling automated transport vehicles for material distribution systems," *Appl. Soft Comput.*, vol. 82, Sep. 2019, Art. no. 105552.

[82] R. Ruiz, C. Maroto, and J. Alcaraz, "Two new robust genetic algorithms for the flowshop scheduling problem," *Omega*, vol. 34, no. 5, pp. 461–476, Oct. 2006.

[83] T. Ladhari, M. K. Msakni, and A. Allahverdi, "Minimizing the total completion time in a two-machine flowshop with sequence-independent setup times," *J. Oper. Res. Soc.*, vol. 63, no. 4, pp. 445–459, Apr. 2012.

[84] S.-H. Chen and M.-C. Chen, "Operators of the two-part encoding genetic algorithm in solving the multiple traveling salesmen problem," in *Proc. Int. Conf. Technol. Appl. Artif. Intell.*, Nov. 2011, pp. 331–336.

[85] J. Alcaraz and C. Maroto, "A robust genetic algorithm for resource allocation in project scheduling," *Ann. operations Res.*, vol. 102, nos. 1–4, pp. 83–109, 2001.

[86] O. Enigin, G. Ceran, and M. Yilmaz, "An efficient ga for hybrid flow shop scheduling with multiprocessor task problem," *Appl. Soft Comput.*, vol. 11, no. 3, pp. 3056–3065, 2011.

[87] C. Fang and L. Wang, "An effective shuffled frog-leaping algorithm for resource-constrained project scheduling problem," *Comput. Oper. Res.*, vol. 39, no. 5, pp. 890–901, May 2012.

[88] L. Wang and C. Fang, "A hybrid estimation of distribution algorithm for solving the resource-constrained project scheduling problem," *Expert Syst. Appl.*, vol. 39, no. 3, pp. 2451–2460, Feb. 2012.

[89] X. Li and M. Li, "Multiobjective local search algorithm-based decomposition for multiobjective permutation flow shop scheduling problem," *IEEE Trans. Eng. Manag.*, vol. 62, no. 4, pp. 544–557, Nov. 2015.

[90] M. Tritschler, A. Naber, and R. Kolisch, "A hybrid Metaheuristic for resource-constrained project scheduling with flexible resource profiles," *Eur. J. Oper. Res.*, vol. 262, no. 1, pp. 262–273, Oct. 2017.

[91] J. H. Patterson, "Project scheduling: The effects of problem structure on heuristic performance," *Nav. Res. Logistics Quart.*, vol. 23, no. 1, pp. 95–123, Mar. 1976.

[92] W. Herroelen, B. De Reyck, and E. Demeulemeester, "Resource-constrained project scheduling: A survey of recent developments," *Comput. Oper. Res.*, vol. 25, no. 4, pp. 279–302, Apr. 1998.

[93] M. Vanhoucke, *Project Management With Dynamic Scheduling*. Berlin, Germany: Springer, 2012.

[94] D. W. Zimmerman and B. D. Zumbo, "Relative power of the Wilcoxon test, the Friedman test, and repeated-measures ANOVA on ranks," *J. Exp. Edu.*, vol. 62, no. 1, pp. 75–86, Jul. 1993.

[95] R. K. Chakrabortty, A. Abbasi, and M. J. Ryan, "Multi-mode resource-constrained project scheduling using modified variable neighborhood search heuristic," *Int. Trans. Oper. Res.*, vol. 27, no. 1, pp. 138–167, Jan. 2020.

[96] M. Vanhoucke, "An efficient hybrid search algorithm for various optimization problems," in *Proc. Eur. Conf. Evol. Comput. Combinat. Optim.* Berlin, Germany: Springer, 2006, pp. 272–283.

[97] E. A. Gehan, "A generalized wilcoxon test for comparing arbitrarily singly-censored samples," *Biometrika*, vol. 52, nos. 1–2, pp. 203–224, Jun. 1965.

[98] H. W. Lilliefors, "On the kolmogorov-smirnov test for normality with mean and variance unknown," *J. Amer. Stat. Assoc.*, vol. 62, no. 318, pp. 399–402, Jun. 1967.

[99] X. Li, J. Zhang, and M. Yin, "Animal migration optimization: An optimization algorithm inspired by animal migration behavior," *Neural Comput. Appl.*, vol. 24, nos. 7–8, pp. 1867–1877, Jun. 2014.

[100] Y.-J. Zheng, "Water wave optimization: A new nature-inspired metaheuristic," *Comput. Oper. Res.*, vol. 55, pp. 1–11, Mar. 2015.

MD. ASADUJJAMAN (Graduate Student Member, IEEE) is currently pursuing the Ph.D. degree in systems engineering with the University of New South Wales, Canberra, ACT, Australia. He is also an Assistant Professor (on leave) with the Department of Industrial and Production Engineering, Rajshahi University of Engineering and Technology. His research interests include a wide range of topics, including supply chain integrated project scheduling, operations research, financial engineering, ergonomics and human factors engineering. He is the author or coauthor more than 25 refereed journal and conference papers.

HUMYUN FUAD RAHMAN received the Ph.D. degree in computer science from the School of Engineering and IT, University of New South Wales, Canberra, ACT, Australia. He is currently a Research Associate (an Associate Lecturer) with the University of New South Wales. His research interests include the area of manufacturing system optimization considering uncertainty and disruptions, Industry 4.0, project, and supply chain management. His research interests include evolutionary computation, manufacturing system optimization, and supply chain management. From 2017 to 2018, he contributed to major research projects with leading defense industries in Europe.

RIPON K. CHAKRABORTTY (Member, IEEE) received the B.Sc., M.Sc., and Ph.D. degrees from the Bangladesh University of Engineering and Technology on Industrial and Production Engineering, in 2009, 2013, and 2017, respectively. He is currently a Lecturer on system engineering and project management and also the Program Coordinator of the Master of Decision Analytics and Master of Engineering Science, School of Engineering and Information Technology, the University of New South Wales (UNSW Australia), Canberra, ACT, Australia. He is currently the Group Leader of the Cross-Disciplinary Optimization Under Capability Context Research Team, and also the Deputy Director of the Capability Systems Centre, UNSW Canberra. He has written two book chapters and more than 95 technical journal and conference papers. His research interests include a wide range of topics in operations research, project management, supply chain management, artificial intelligence, cyber-physical systems, and information systems management. His research program has been funded by many organizations, such as Department of Defence- Commonwealth Government, Australia.

MICHAEL J. RYAN (Senior Member, IEEE) is currently the Director of the Capability Systems Centre, University of New South Wales, Canberra, ACT, Australia. He also lectures and regularly consults in a range of subjects including communications systems, systems engineering, requirements engineering, and project management. He is also the Co-Chair of the Requirements Working Group, International Council on Systems Engineering (INCOSE). He is a Fellow of Engineers Australia, a Fellow of the International Council on Systems Engineering, and a Fellow of the Institute of Managers and Leaders. He is the author or coauthor of 12 books, three book chapters, and more than 250 technical articles and reports.