**ORIGINAL RESEARCH**

# An ABC-Genetic method to solve resource constrained project scheduling problem

**Vahid Zeighami**[1], **Reza Akbari**[2], **Ismail Akbari**[3], **Yevgen Biletskiy**[4]

1. Department of Mathematics, Shiraz University, Shiraz, Iran. 2. Department of Computer Engineering and Information Technology, Shiraz University of Technology, Shiraz, Iran. 3. Department of Electrical and Computer Engineering, University of New Brunswick, Fredericton, Canada. 4. Department of Electrical and Computer Engineering, University of New Brunswick, Fredericton, Canada.

**Correspondence:** Vahid Zeighami. Address: Department of Mathematics, Shiraz University, Shiraz, Iran. Email: vahid.zeighami@gmail.com.

**Received:** May 13, 2012 **Accepted:** August 20, 2012 **Published:** December 1, 2012
**DOI:** 10.5430/air.v1n2p185 **URL:** http://dx.doi.org/10.5430/air.v1n2p185

## Abstract
The aim of this work is to study the effect of hybridization on the performance of the Artificial Bee Colony (ABC) as a recently introduced metaheuristic for solving Resource Constrained Project Scheduling Problem (RCPSP). For this purposes, the ABC is combined with a Genetic Algorithm (GA). At the initial time, the algorithm generates a set of schedules randomly. The initial solution has been evaluated against constraints, and the infeasible solutions have been resolved to feasible ones. Then, the initial schedules are to be improved iteratively using a hybrid method until termination condition is met. The proposed method works by integrating the ABC and GA search processes. The GA method updates schedules by including the best solutions found by the ABC approach. Next, the ABC method picks the solutions found by GA search. The new methodological approach is used by the algorithm to maintain the priorities of the activities in feasible ranges. The performance of the proposed algorithm has been compared against a set of state-of-art algorithms. The simulation results have demonstrated that the proposed algorithm provides an efficient way for solving RCPSP and produce competitive results compared to other algorithms investigated in this work.

## Keywords
Artificial bee colony, Genetic algorithm, Hybrid method, Resource constrained project scheduling problem

## 1 Introduction
The single mode Resource Constrained Project Scheduling Problem (RCPSP) problem has been studied in this work. The RCPSP is known as a NP-hard problem [1]. Different types of algorithms ranging from exact to meta-heuristics have been proposed by researchers, and practitioners to tackle the complexities of the RCPSP problems. The exact methods have difficulties in solving large-scale RCPSP problems [2]. Hence, the use heuristic and meta-heuristic methods are needed. There are different classes of heuristics and meta-heuristics such as; Simulated Annealing (SA), Ant Colony Optimization (ACO), Particle Swarm Optimization (PSO), Genetic Algorithms (GA), Bee Algorithms (BA), etc which have been successfully used to solve RCPSP problems. The meta-heuristic methods have the ability to solve large-scale RCPSP because they provide at least one solution from the start of the algorithm. However, they may trap in local optima that result sub-optimal solutions.

The previous studies on the heuristics and meta-heuristics demonstrated that these techniques have substantive efficiency in solving RCPSP problems. However, it seems that more efficiency may be obtained by hybridization of these techniques. This assumption encourages authors to design a hybrid algorithm to solve the RCPSP problems. The success of a hybrid methods have roots in the fact that they utilize the advantages of two algorithms. Usually, heuristics and meta-heuristics have some positive effects, as well as some drawbacks. However, the use of two or more techniques provides the ability to use the advantages to mitigate the drawbacks. Nowadays, different types of hybrid methods have been proposed to solve the RCPSP problems. The ANGEL [3], ACOSS [4], Neurogenetic [5], and GA-Hybrid [6] are known as some of representative hybrid methods presented in literature.

The ANGEL method combines ACO, GA, and a local search as a hybrid method [3]. In this method, the ACO generates the initial population for the GA. The GA is applied on the population, and the pheromones are updated when better solutions obtained by GA. If the GA is terminated, the ACO starts a new search. This process is iterated by the ANGEL until termination condition is met. The ACCOS is a hybrid method which works by combining a local search strategy, the ACO and scatter search[4]. In this method, the ACO generates the first population. The pheromone trails of the ACO is used by the scatter method to improve the solutions. After that, ACO uses the scatter results to update the pheromone set and performs a new search. The Neurogenetic was designed as a hybrid of genetic algorithm and neural networks [5]. In this method, the search process is based on the genetic iterations for global search and on the neural network for local search. The method works by interleaving genetic and neural networks where the best result is generated by the genetic algorithm. This result is used by neural network for local search, and the solutions obtained by the neural network are added to the genetic population for further search using genetic iterations. GA-Hybrid is another hybrid method that uses peak crossover operator for RCPSP [6]. It also uses a local improvement operator which is applied to the generated schedules.

The study of hybrid methods shows that in most cases, a hybrid approach works by employing two or more search methods. Hence, it is possible to provide a framework for hybridizing search methods as shown in Figure 1. The search methods pass the solution found by them to each other. The search method $S_x(S_y)$ first process the knowledge provided by the cooperative search method $S_y(S_x)$. The knowledge may be used for updating the solutions of the $S_x$ $(S_y)$ search methods, or it may be used for tuning parameters of the $S_x$ $(S_y)$ search methods. Usually, a hybrid method such as ACCOS and Neurogenetic tries to employ one of their techniques to provide global search while the other(s) act as local search.

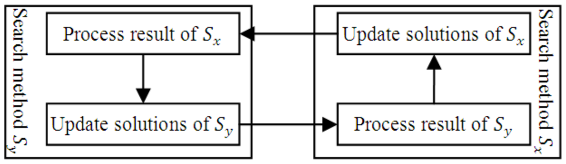


**Figure 1.**A general framework for combination two or more search methods to solve RCPSP problem.

From Figure 1, it seems that successful methods for RCPSP can be designed by combining two or more consistent search methods. This fact encourages designing a new hybrid method. Hence, this work is aimed to propose a new method to solve the RCPSP problem. The proposed method called ABC-GA hybridizes; the ABC meta-heuristic with the genetic algorithm. The method works by integrating these two techniques. Each technique receives the results of the other one, improves them, and returns them. The ABC-GA method has been tested on the single mode RCPSP case studies and compared with the other methods proposed in literature. The efficiency of the proposed method is evaluated as the outcome of the experiments.

The remaining of this paper is organized as follows. Section 2 introduces basic concepts of the RCPSP problems. Section 3 describes the specification of the proposed method. Section 4 presents the benchmarks, experiments, and comparative studies. Finally, Section 5 concludes this work.

## 2 Basic Concepts

The mathematical representation of the single mode resource constrained project scheduling problem is presented here. In single mode-RCPSP, it is assumed that there is a project containing $n$ main activities $\{A_1, A_2, \ldots, A_n\}$, so called non-dummy activities and two dummy activities $\{A_0, A_{n+1}\}$, which represent the beginning and the end of the project. Hence the project has $A = \{A_0, A_{n+1}\} \cup \{A_1, A_2, \ldots, A_n\}$ activities. Also, the project uses $K$ renewable resource types $R = \{R_1, R_2, \ldots, R_K\}$. The duration of an activity $A_j$ is denoted as $d_j$. Each activity $A_j$ needs $r_{jk}$ units of resource $R_k$ during each period of its duration. For the dummy activities $A_0$ and $A_{n+1}$ we have $d_0 = d_{n+1} = 0$ and $r_{0k} = r_{n+1k} = 0$. Using these information, the project can be modeled as a precedence graph. For the sake of simplicity, an example of the single mode RCPSP problem is given in Figure 2. The beginning activity $A_0$ and the end activity $A_9$ are dummy activities while the other ones represent the non-dummy activities. The RCPSP problem can be modeled as a directed graph, so called precedence graph. The nodes of the graph represent the activities and the links between nodes represent the precedence constraints. Also, each node has the duration and the required units of each resource type is given.

Based on the assumptions, the main objective of the RCPSP is to minimize the makespan of the project $P$ with the schedule $S_n$. The minimization process is formulated as follows:

$$\text{Minimize the makespan of schedule } S_n \tag{1}$$

Subject to:

$$S_l \leq S_j - d_j, \;\; j = 1, \ldots, n+1; \; l \in M_j, \tag{2}$$

where $M_j$ presents a set of preceding activities of the activity $A_j$ and $S_j$is the finish time of the activity $A_j$. The RCPSP is a constraint handling problem. Resource and precedence constraint are known as two main constraints that should be satisfied in solving RCPSP problem. Equation 2 shows the precedence constraints between activities, and the Equation 3 shows the resource constraints.

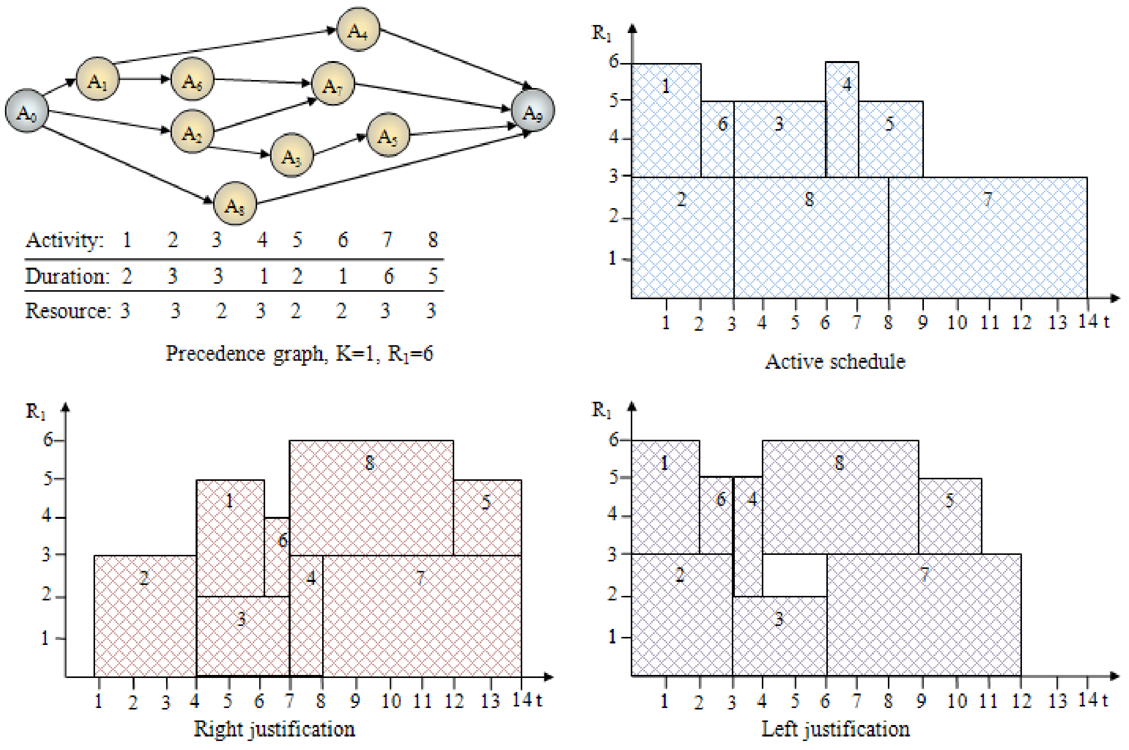
$$\textstyle\sum_{j \in \vartheta(t)} r_{jk} \leq R_k, \quad k \in \{1, 2, \ldots, K\}, \tag{3}$$

Where $\vartheta(t)$ represents the resource limitation constraint, and $S_j \geq 0, \; j \in \{1, 2, \ldots, n+1\}$ describes the constraints of decision variables.

**Serial and parallel SGS**: As described before, the main objective of the RCPSP is to minimize the makespan of the project. For this purpose, we need to construct schedules using the activities of the project. Hence, we need to use a schedule generation scheme (SGS)[7]. Serial-SGS and Parallel-SGS are two approaches used to generate schedules [7]. The Serial-SGS and Parallel-SGS use respectively time-incrementation and activity-incrementation approaches to schedule activities of a project. In serial-SGS, a schedule is generated from the list of activities in $n$ stages. One activity is selected at each stage and scheduled at the earliest precedence and resources feasible time. Parallel-SGS schedules the activities with the feasible precedence and resource at each point of time. Each of the serial-SGS and parallel-SGS has its own advantage as well as drawbacks. The parallel SGS may fail in constructing an optimal schedule. However, the parallel SGS has the ability to produce schedules of better average quality because it utilizes resources as early as possible. Under this way, a more compact schedules may be obtained [4]. Hence, one can select one of them for schedule generation. However, it is possible to use both of them.

**Double justification:** In recent years, different approaches have been suggested by researchers to obtain better schedules. Double justification (DJ) is known as an efficient way for generating better schedules [8]. Double justification is based on the right and left justifications. The justification tries to adjust the start time of each activity in scheduling in order to obtain better makespan. Usually, the makespan of the justified schedules is shorter than that before justification [9]. The Right

(Left) justification is aimed to start the activity as late (early) as possible while the start times of the other activities are remained unchanged. In Right justification, all the activities are justified in decreasing order of their finishing times, and in the Left justification, all the activities are justified in ascending order of their starting times. In the Right (Left) justification, the set of activities $A$ are scheduled in decreasing (increasing) order according to their finish time (start time). At each step of the right (left) justification the start time $s_j$ of the activity $A_j$ is adjusted in sequence such that the adjusted start time $s_j^{'} \geq s_j (s_j^{'} \leq s_j)$ and make the $s_j^{'}$ as large (small) as possible. The justification is finished after all the activities have been justified [8][8].



| Activity: | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| Duration: | 2 | 3 | 3 | 1 | 2 | 1 | 6 | 5 |
| Resource: | 3 | 3 | 2 | 3 | 2 | 2 | 3 | 3 |

**Figure 2**.A precedence graph and its corresponding double justifications.

We can use the right and left justification in order to perform double justification. In double justification, first the right justification is performed on the schedule $S$ and a new schedule called $S^R$ is obtained. After that the left justification is performed on the $S^R$ and a new schedule $(S^R)^L$ is obtained. Based on these justification approaches, the double justification can be modeled as $DJ(S) = (S^R)^L$. The double justification provide the ability to shorten the makespan of schedule $S$ using the right and left justifications. Figure2 shows a double justification which is created based on right and left justifications. As can be seen from this figure, the makespan of the project is decreased using double justification.

## 3 Hybrid RCPSP Solver

The details of the proposed hybrid algorithm are presented in this section. The hybrid method called ABC-GA works by interleaving the ABC [10] and GA [11] phases. The ABC phase acts as a global search while the GA phase is used as a local search. The ABC phase tries to find more profitable regions in the search space. These regions may contain schedules with smaller makespans. Finding these regions provide the ability for the GA phase to perform local search in a more efficient way.

The schematic diagram of the ABC-GA algorithm is given in Figure 3. The ABC-GA method receives population size, Max_Trial, crossover probability, mutation rate and the project $Prj$ to be scheduled as the input parameters. The algorithm starts by initializing the individuals (schedules) of the population. The initialization phase usually returns a set of feasible and infeasible schedules. Hence the initial schedules should be checked against the constraints and resolved to feasible solutions. After initialization, the main body of the algorithm starts.

The main body is constituted from ABC and GA phases. The ABC phase updates the solutions using artificial bee colony method which was proposed by the Karaboga [10]. In the first step of the ABC phase the previous feasible solutions are used in order to compute the feasible boundaries for the priorities of the activities of each schedule. After that the employed and onlooker bees are used to generate new schedules using the scenario given in Section 3.2. After sending employed or onlooker bees, the boundary checking module is called in order to maintain the priorities of the activities in feasible ranges. At each cycle of the algorithm, the stagnant solutions are determined and their solutions are replaced with the new random solutions. For this purpose, the scout bees are used. The randomly generated solutions are passed to the constraint handling module in order to resolve the infeasible solutions to feasible ones.

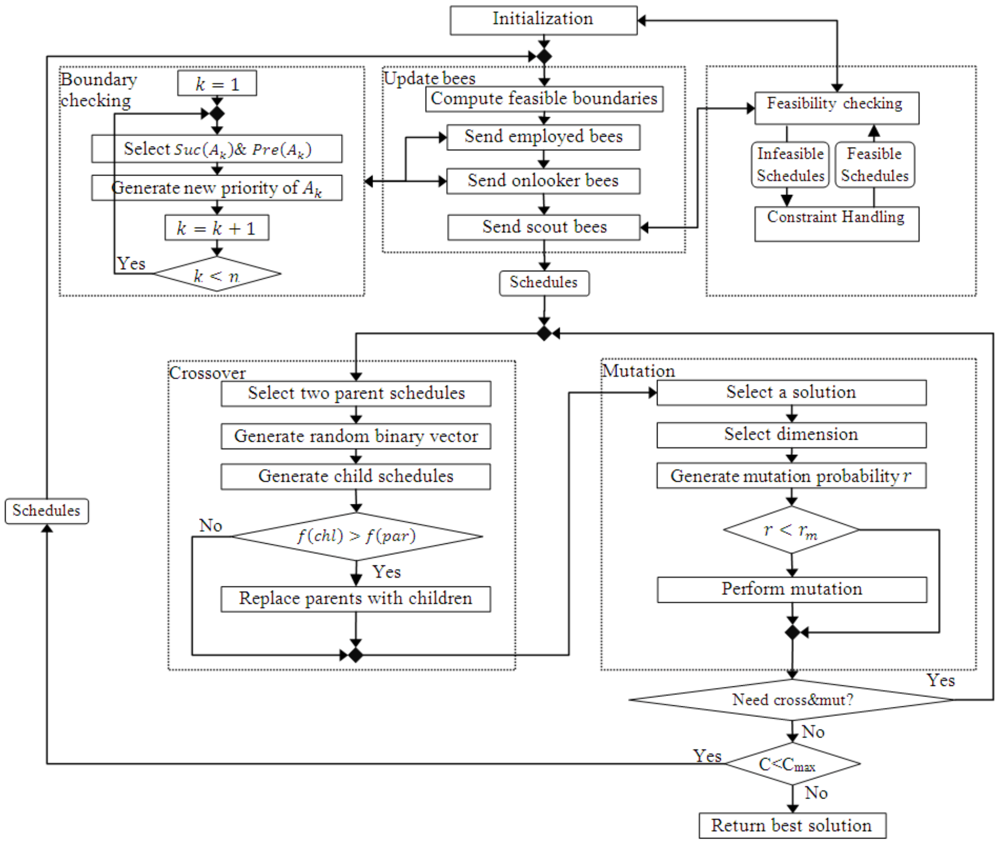


**Figure 3.**Schematic diagram of the proposed ABC-GA algorithm.

The resultant schedules are passed to the GA phase. The GA phase tries to improve the output of the ABC phase by employing the genetic operators such as crossover and mutation. In GA phase, a predefined percentage of the schedules are selected as parents and the crossover operation is performed on them. The generated children are compared against their parents and if they have better performance, their parents are removed and they are added to the population. The crossover is succeeded by the mutation. The updated solutions by the GA phase are returned to the ABC phase and this interleaving is iterated cycle by cycle until the termination condition is met. After termination, the best solution found by the algorithm is returned as the final result. The details of the main modules of the ABC-GA algorithm are given in the following sub-sections.

## 3.1 Initialization

The ABC-GA algorithm starts with the $m$ solutions being placed randomly in the $n$-dimensional solution space where $n$ represents number of the activities to be scheduled. Each solution represent an $n$-dimensional vector $\vec{x}_i = (x_{i1}, x_{i2}, \dots, x_{in})$ where $0 \leq x_{ij} \leq 1 \quad \forall j \in \{1,2,\dots,n\}$. Each component of the solution vector represents a priority of the corresponding activity in the project. Hence, it can be seen as a priority list. The initialization process is represented in Figure 4. This Figure relates to the example introduced in Figure 2. For each of the initial solution vector, a set of $n$ random numbers are generated to show the initial priority of the corresponding activities. After that, these initial priorities and their corresponding activities are sorted in descending order. This process may result an infeasible solution. Hence, at the next step, the constraint handling process in applied on the initial solution vector in order to obtain a feasible solution. For this purpose, the activity which violates the constraints is determined and changed with the next activity with smaller priority. After that the constraint handling process is applied on this newly emerged activity list. This process is iterated until the infeasible solution is converted to a feasible one.

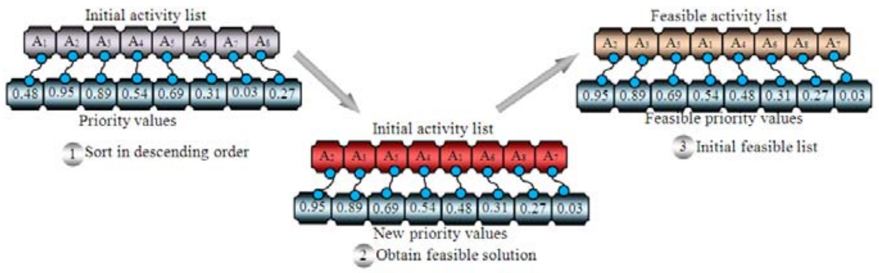


**Figure4.**Initialization of an individual in ABC-GA algorithm.

## 3.2 ABC Phase

Initialization phase starts the ABC-GA algorithm. The initialization phase generates a set of schedules which are used by the ABC and GA phases to provide new schedules with smaller makespans. The ABC and GA phase are iterated cycle by cycle until the termination condition is met. The ABC phase has four steps: 1) compute feasible boundaries, 2) send employed bees, 3) send onlooker bees and 4) send scout bees. The details of these steps are described in the following sub-sections. The ABC method has been successfully used by the authors to solve the RCPSP problem[12, 13]. The ABC phase of the ABC-GA method is based on the works presented in[12, 13].

### 3.2.1 Compute Feasible Ranges

The first step of the ABC phase tries to compute the feasible ranges of the activities in each of the schedules passed from the initialization or ABC phase. These feasible ranges are used after the “send employed bees” and “send onlooker bees” steps in order to maintain the priorities of their activities in feasible ranges. The feasible ranges for the activities of an activity list $A$ are generated as follows: we start from the beginning of the activity list $A$ and compute the lower bound and the upper bound of each activity $A_k \in A$. For this purpose the immediate predecessor $A_r$ and the successor $A_s$ of the activity $A_k$ are selected in the list. The corresponding priorities of $A_r$ and $A_s$ are used to compute the possible range for the priority value of activity $A_k$. Under this way, the lower bound and upper bound of the feasible priority for the activity $A_k$ are set at $P_s + \frac{P_k - P_s}{2}$ and $P_k + \frac{P_r - P_k}{2}$ respectively, where $P_k$, $P_r$, and $P_s$ represents the corresponding priority values of the activity $A_k$, predecessor activity $A_r$, and the successor activity $A_s$. The new priority of each activity $A_k \in A$ is selected from the following range:

$$P_k \in \left[P_s + \frac{P_k - P_s}{2}, P_k + \frac{P_r - P_k}{2}\right] \tag{4}$$

For the activity with no predecessor (successor), its lower bound (upper bound) is set at 0 (1). Consider the following example: it is assumed that the activity list $\Pi_i$ and its corresponding priority list $\vec{P_i}$ is given as shown in Figure 4. The priority values and new lower bound and upper bound for each activity list are computed as follows: Since the immediate successor $A_1$ is $A_6$ and there is no predecessor, we have $P_{j,1} \in \left[P_{j,6} + \frac{p_{j,1} - p_{j,6}}{2}, 1\right] = \left[0.31 + \frac{0.54 - 0.31}{2}, 1\right] = [0.42,1]$. Similarly, the same scenario is repeated for all the activities, as given below.

$$P_{j,1} \in \left[0.31 + \frac{0.54 - 0.31}{2}, 1\right] = [0.42,1]\,, P_{j,5} \in \left[0, 0.69 + \frac{0.86 - 0.69}{2}\right] = [0,0.77]$$

$$P_{j,2} \in \left[0.89 + \frac{0.95 - 0.89}{2}, 1\right] = [0.92,1]\,, \qquad P_{j,6} \in \left[0.03 + \frac{0.31 - 0.03}{2}, 0.31 + \frac{0.54 - 0.31}{2}\right] = [0.17,0.42]$$

$$P_{j,3} \in \left[0.69 + \frac{0.89 - 0.69}{2}, 0.89 + \frac{0.95 - 0.89}{2}\right] = [0.79,0.92]\,, P_{j,7} \in \left[0, 0.03 + \frac{0.31 - 0.03}{2}\right] = [0,0.17]$$

$$P_{j,4} \in \left[0, 0.48 + \frac{0.54 - 0.48}{2}\right] = [0,51]\,, P_{j,8} \in [0,1] = [0,1]$$

In this approach, the newly emerged activity lists are feasible. The lower bound and upper bound of each dimension of the search space is defined as $lb = \mathrm{P}_{\mathrm{i,s}} + \frac{\mathrm{p_{i,k}} - \mathrm{p_{i,s}}}{2}$ and $\mathrm{ub} = \mathrm{P}_{\mathrm{i,k}} + \frac{\mathrm{p_{i,r}} - \mathrm{p_{i,k}}}{2}$. Hence the value of each element must be limited to $[lb, ub]$.

### 3.2.2 Send Employed Bees

The ABC phase involves employed, onlooker and scout bees to improve previous schedules. For this purpose, the ABC phase needs to evaluate the fitness of its individuals. Hence, one of the scheduling scheme described in Section 2 is used here. The ABC-GA method uses both the serial and parallel scheduling in order to use their advantages. Under this way, a random real number $\mathrm{r_{SGS}} (0 \le \mathrm{r_{SGS}} \le 1)$ is used to select the type of SGS which is used to construct the schedule. This random number is used as follows:

$$\mathrm{SGS} = \begin{cases} \text{Serial} - \text{SGS} & \text{if } \mathrm{r_{SGS}} \le 0.5 \\ \text{Parallel} - \text{SGS} & \text{if } \mathrm{r_{SGS}} > 0.5 \end{cases} \tag{5}$$

The random $\mathrm{r_{SGS}}$ generated for a given individual is used to decide the type of schedule that should be used. The SGS helps the method to evaluate the fitness of an individual. The fitness of the individual is defined as the makespan of the schedule. Beside the serial/parallel scheduling, the ABC-GA method uses the double justification in order to obtain better performance.

By generating the schedules for the individuals and computing their fitness, the movement patterns are used by the ABC phase in order to generate new schedules. The first pattern is used by the employed bees. An employed bee $i$ selects another employed bee as its neighbor in order to optimize the current schedule. After that the solution found by the employed bee$i$ is replaced by the new solution suggested by equation (6) if the new solution has better performance:

$$x_{i,t+1} = \begin{cases} x_{i,t} + wr_i(x_{i,t} - x_{k,t}) & iff(x_{i,t+1}) \ge f(x_{i,t}) \\ x_{i,t} & if\ f(x_{i,t+1}) < f(x_{i,t}) \end{cases} \tag{6}$$

where $x_{i,t}$ and $x_{i,t+1}$ respectively represent the current and next position of the solution $i$. The randomly chosen index $k \in \{1,2, \ldots, i-1, i+1, \ldots, FoodNumber\}$ represents the neighbor solution. The function $f$ shows the fitness of a solution provided by the algorithm. The fitness of a solution is computed based on the makespan provided by that solution. Finally, $-1 \leq r_i \leq 1$ is a random coefficient and $w_1$ is a weight which controls the importance of the neighbor solution.

The first movement pattern given in Equation 6, helps the employed bees to update their positions. The new position of an employed bee may return an infeasible solution. To solve this problem, the new solutions generated by the employed bees are passed to a boundary checking module. This module uses the approach described in Section 3.2.1 for maintaining the priorities of activities in feasible ranges. More precisely, the corresponding priority list of the activity list $A$ which is presented by an employed bee is considered.

### 3.2.3 Send Onlooker Bees

The second movement pattern is used by the onlooker bees. An onlooker bee follows one of the solutions found by the employed bees as its own leader. It uses the roulette wheel approach to selects an advertised solution $k$ with the following probability:

$$p_k = \frac{f(\vec{x}_k)}{\sum_{j=1}^{FoodNumber} f(\vec{x}_j)} \tag{7}$$

After selecting the leader solution, the onlooker bee updates its position using the same approach used by the employed bee. The only difference is that the onlooker bees use different value for the weight $w$. Under this probabilistic way, all the employed bees share their knowledge with onlooker bees. However, the employed bees with better performances have more chances to be selected by the onlooker bees. Similar to the employed bees, the feasibility of new schedules produced by the onlooker bees are investigated. For this purpose, the boundary checking is applied in order to maintain the priorities in the feasible ranges.

### 3.2.4 Send Scout Bees

The third movement pattern which is given in Equation 8 is used by the scout bees. At each cycle, the ABC phase evaluates the solutions to check if they should be abandoned or not. If a solution could not improve itself after predefined number of iterations (Max_Trial), it is abandoned and be replaced by a new randomly generated solution if the new solution has better performance. The abandoned solution is updated as follows:

$$x_{id,t+1} = \begin{cases} rand(1) & if f\left(x_{i,t+1}\right) \geq f\left(x_{i,t}\right) \\ x_{id,t} & if f\left(x_{i,t+1}\right) < f\left(x_{i,t}\right) \end{cases}, \tag{8}$$

where $rand(1)$ returns a random number between 0 and 1. After reinitializing the scout bee, the constraint handling is performed in order to obtain a feasible solution.

## 3.3 Genetic Phase

The proposed ABC phase is succeeded by the genetic phase. The produced schedules by the ABC phase are passed to the genetic phase in order to obtain new schedules with smaller makespans. The genetic algorithms are based on two main biological mechanisms: crossover and mutation. The genetic algorithms have been widely used by researchers to solve the RCPSP problems [14-17]. In the present research, a modified version of the genetic algorithm proposed by Hartman [17] is deployed.

### 3.3.1 Crossover

The crossover operator proposed by Hartman [17] is used as follows. First, a $1 \times n$ binary vector $(x_1, x_2, \dots, x_n)$ with $x_i \in \{0,1\}$ is generated, where $n$ represents the number of non-dummy activities to be scheduled in the project. The value of each component $x_i$ is randomly generated using a random generator with uniform distribution. Second, the produced binary vector is used by the algorithm to generate new solutions from the current population.

Figure 5 shows the scheduling generation process. First, two schedules from the current population are selected randomly based on the crossover probability. These two schedules are referred as father and mother. The sequence of activities in father and mother schedules determines the schedules inherited by the children. The random binary vector is used to generate two children which are called daughter and son. Each position $i \in \{1,2,\dots,n\}$ of the daughter's schedule will be a copy of the $i$-th position of her mother if $x_i = 0$ while this position will be a copy of the corresponding position in her father activity if $x_i = 1$. The reverse process is used to generate the son's schedule. In the son's schedule, each position $i \in \{1,2,\dots,n\}$ will be a copy of the $i$-th position of his father if $x_i = 1$ while this position will be a copy of the corresponding position in his mother activity if $x_i = 0$. In the priority list representation, as no position can be repeated, the daughter's vector is filled with the first available position of its mother if $x_i = 1$ and with the first available position of its father if $x_i = 0$. The similar way is used to construct the son's vector.

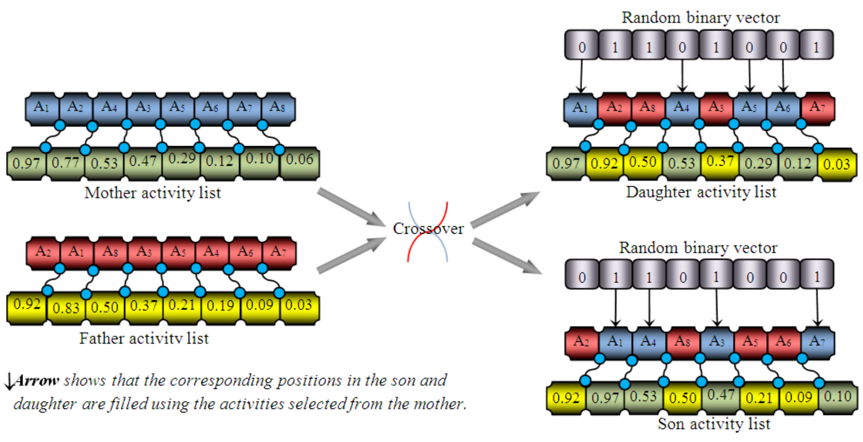


**Figure 5.**Using uniform crossover for generating new solutions from the current population.

As can be seen, this process is iterated $n$ times for generating daughter or son schedule. At each step of this process, after selecting an activity from the mother (father) and copying this activity to the daughter or son schedule, the corresponding schedule in father(mother) is removed. This is occurred due to the constraint that each activity should be considered one time in each schedule. Also in this process, the associated priority value of each activity which is copied to the son or daughter schedules is copied. By generating, the new schedules, the ABC-GA algorithm replaces the parent schedules with the children if the children have better fitness compared to their parents.

For the sake of more clarity, the scheduling process in Figure 5 is considered. In this example, the randomly generated vector is $X = (0,1,1,0,1,0,0,1)$. The process considers $x_1 = 0$. Due to the zero value of first position in the random vector, the first available activity (i.e. $A_1$) from the mother's activity is selected and copied to the first position of the daughter's activity. Also the corresponding priority value 0.97 is copied. Next, the activity $A_1$ is removed from the mother and father. The second position in the random vector is 1. Hence, $A_2$ and its corresponding priority from the father is copied to the daughter's schedule and then it is removed from the parent schedules. Next, due to $x_3 = 1$, the third position of the daughter's schedules is filled by $A_8$. In a similar way, due to $x_4 = 0$, $x_5 = 1$, $x_6 = 0$, $x_7 = 0$ and $x_8 = 1$, the corresponding positions in the daughter's schedules are filled by $A_3$, $A_6$,$A_5$, and $A_7$which are selected from the father mother, mother, and father correspondingly. The similar approach is used to generate the son schedule.

The uniform crossover operator results two new child schedules from the parent schedules. After generating each schedule, the algorithm should investigate its feasibility. An interesting characteristic of the crossover operator is that the newly generated schedules are always feasible and the ABC-GA algorithm doesn't need to investigate them against the constraints. Hence, the algorithm need not further time to check the schedules against the constraints.

### 3.3.2 Mutation

The crossover operator has the main role in generating new schedules in the GA phase. However, a mutation crossover is also used here in order to provide appropriate level of diversity throughout the execution of the algorithm. The mutation is performed with the mutation probability $r_m$. The mutation process is performed as follows: first a random number $r$ is generated. If $r < r_m$ then an schedules is selected randomly in order to mutate one of its position. A random position $r_p$ is selected and its corresponding priority value is randomly modified.

## 3.4 Termination

The proposed method terminates after generating a predefined number of schedules. After termination, the schedule with minimum makespan obtained by the population is returned as the output.

## 4 Experiments

It seems that the proposed algorithm is a good compromise which utilizes the potentials of ABC and genetic algorithms. A set of experiments are conducted in order to investigate the performance of the proposed hybridization. The experiments are divided in two categories. The first category evaluates the proposed method in comparison with the bee algorithms proposed by the authors in their previous works [12, 13] in terms of success rate. The second category investigate the performance of the proposed method against other bee algorithms in terms of average deviation.

The Single Mode Data Sets cases given in PSPLIB library [18] are used here to compare the proposed algorithm with other state-of-art algorithms. This library contains benchmarks with j30, j60, j90, and j120 case studies. The numbers of schedules are chosen at 1000, 5000 and 50000, and a total of 10 runs for each experimental setting are conducted. The proposed algorithm is evaluated under the configuration given in Table 1. The values of the parameters are determined based on our empirical studies.

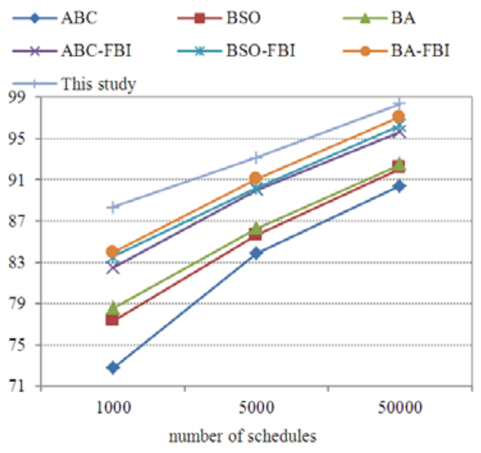
## 4.1 Comparison with Bee Algorithms

Our previous studies on the bee algorithms[12][12, 13] showed that this type of meta-heuristics have the ability to provide good result on the RCPSP problems. In this experiment, the main objective is to study the effect of the genetic algorithm on the performance of the standard ABC algorithm. For this purpose, a set of experiments have been conducted to verify how many cases of PSPLIB library can be solved by the hybrid of ABC and genetic algorithm. In this experiments, the algorithms try to find optimal solution or lower bound solution. We refer to the percentage of the instances that an algorithm capable to find their optimal solutions or lower bounds as the success rate. Finding the optimal solution or the lower bounds means that an algorithm successfully solve that instance. The success rate is defined as (N/M)*100, where N is the number of instances successfully solve by an algorithm and M is the total number of instances in each case study.

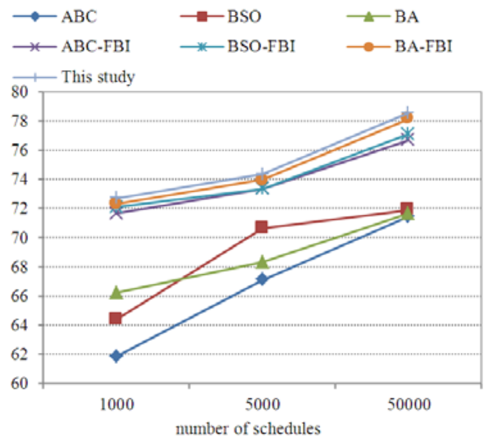
**Table 1.**Settings of the parameters

| Parameter | Value |
|---|---|
| $w$ for employeds | 0.8 |
| $w$ for onlookers | 1.2 |
| Max_Trial | 5 |
| Crossover probability | 0.6 |
| Mutation probability | 0.05 |
| Population size | 35 |

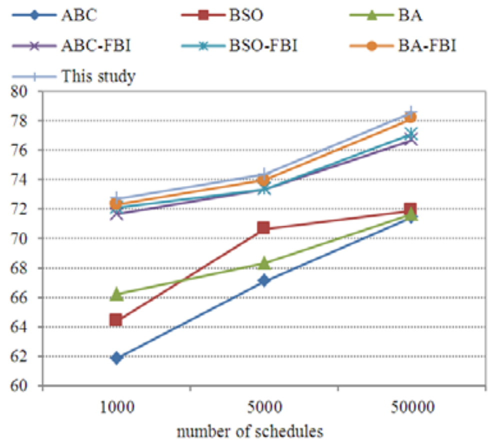
In addition to the success rate, another measure called the average percent deviation is used here for comparison. This measure is defined as the average of (A-CPM)*100/CPM over the set of instances in a case study, where A shows the makespan found by an algorithm on an instance and the CPM is the length of the schedule in which every activity is scheduled as early as possible without considering the resource restrictions.



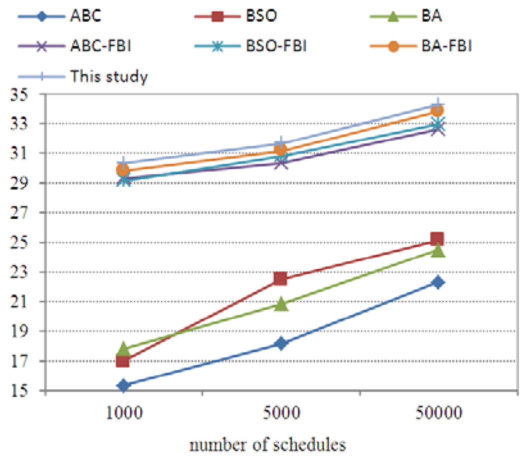
(a) j30 case study



(b) j60 case study



(c) j90 case study



(d) j120 case study

**Figure 6.** Average success rates of the proposed method in comparison with other variants of ABC method for j30, j60, j90, and j120 case studies.

The average percentage of the instances in the j30, j60, j90, and j120 case studies which are solved successfully by the investigated algorithms are given in Figure 6. The vertical axis shows the success rate and the horizontal one shows the number of schedules produced by an algorithm before termination. From the results, it can be seen that the proposed hybridization has positive effect on the standard ABC algorithm. The proposed hybrid method has the ability to find more optimal solutions.

The first experiment shows the superiority of the hybrid method in term of the success rate. In the second experiment, the average standard deviation is considered. Tables 2-4 show the average deviation of the investigated algorithms on j30, j60, and j120 case studies after 1000, 5000, and 50000 schedule generation. Similar to the first measure, better performance may be obtained by using the ABC in conjunction with the genetic algorithm. The performance depends on the search process used by the proposed algorithm. The ABC-GA algorithm works by interleaving global and local searches. The ABC phase works a global search technique and the genetic algorithm acts as a local search. In one hand the global search provide the ability to consider a large part of search space and navigate the algorithm towards regions which contain better solutions. In other hand, the local search helps the algorithm to pay more attention on the neighbor solutions. Apart from the local and global searches, our empirical study showed that the serial/parallel scheduling provide better performance in comparison with serial or parallel scheduling.

**Table 2.**Average deviation (%) from optimal makespan for j30 case study.

| j30 | Algorithm | 1000 | 5000 | 50,000 |
|---|---|---|---|---|
| 1 | **ABC- GA, FBI** | **0.25** | **0.15** | **0.02** |
| 2 | BA-FBI | 0.42 | 0.19 | 0.04 |
| 3 | BSO-FBI | 0.45 | 0.22 | 0.07 |
| 4 | ABC-FBI | 0.47 | 0.28 | 0.09 |
| 5 | BA | 0.63 | 0.33 | 0.16 |
| 6 | BSO | 0.65 | 0.36 | 0.17 |
| 7 | ABC | 0.98 | 0.57 | 0.20 |

**Table 3.**Average deviation (%) from critical path lower bound for j60 case study.

| J60 | Algorithm | 1000 | 5000 | 50,000 |
|---|---|---|---|---|
| 1 | **ABC- GA, FBI** | **11.80** | **11.38** | **10.90** |
| 2 | BA-FBI | 12.55 | 12.04 | 11.16 |
| 3 | BSO-FBI | 0.45 | 0.22 | 0.07 |
| 4 | ABC-FBI | 12.61 | 12.24 | 11.23 |
| 5 | BA | 13.35 | 12.83 | 12.41 |
| 6 | BSO | 13.67 | 12.70 | 12.45 |
| 7 | ABC | 14.57 | 13.12 | 12.53 |

**Table 4.**Average deviation (%) from critical path lower bound for j120 case study.

| J120 | Algorithm | 1000 | 5000 | 50,000 |
|---|---|---|---|---|
| 1 | **ABC- GA, FBI** | **35.86** | **34.37** | **33.14** |
| 2 | BA-FBI | 37.72 | 36.76 | 34.55 |
| 3 | BSO-FBI | 37.84 | 36.51 | 34.86 |
| 4 | ABC-FBI | 37.85 | 36.82 | 35.02 |
| 5 | BSO | 41.18 | 37.86 | 35.70 |
| 6 | BA | 40.38 | 38.12 | 36.12 |
| 7 | ABC | 43.24 | 39.87 | 37.36 |

### 4.2 Discussion

In general, the results demonstrate that the ABC method has good performance on the RCPSP problem. The ABC method has the ability to be integrated with some other algorithms, such as genetic algorithms, in order to provide more efficiency in solving hard problems, such as RCPSP. Hence, in this work, the ABC method is integrated with the GA, and a new method called ABC-GA is emerged. The results showed that the ABC-GA method utilizes the efficiency of genetic algorithms along with the ABC in order to provide better results compared to the ABC and the other bees algorithms. The present research showed that the ABC has the ability to hybridize with the other heuristic, meta-heuristic, and local search methods in order to obtain better solutions.

## 5 Conclusions

A new hybrid method for solving resource-constrained project scheduling problem has been proposed in the present work. The method iteratively updates the initial arrangements of activities using an adaptive decision making process. The method utilizes potentials of both the ABC and GA methods. The ABC-GA integrates the ABC and GA in two phases for each cycle. Each cycle starts by the ABC phase. The schedules obtained by the ABC phase are passed to the GA phase. In

GA phase, new schedules are generated by applying the GA operators. After that, the results of GA phase are passed to ABC phase, and the next iteration starts. The performances of the proposed method have been investigated on a set of well-known benchmarks. The overall performance showed that the proposed method provides high efficiency in solving RCPSP problem, and better performance can be obtained by integrating ABC and GA.

## References

[1] Blazewicz J., Lenstra JK., and RinnooyKan AHG. Scheduling projects to resource constraints: classification and complexity. Discrete Applied Mathematics. 1983; 5: 11-24.http://dx.doi.org/10.1016/0166-218X(83)90012-4

[2] Hartmann S., and Briskorn D., A survey of variants and extensions of the resource-constrained project scheduling problem. European Journal of Operational Research. 2010; 207: 1-14.http://dx.doi.org/10.1016/j.ejor.2009.11.005

[3] L.Y. Tseng, S.C. Chen, “A hybrid metaheuristic for the resource-constrained project scheduling problem” European Journal of Operational Research. 2006; 175: 707-721.http://dx.doi.org/10.1016/j.ejor.2005.06.014

[4] W. Chen, Y.J. Shi, H.F. Teng, X.P. Lan, L.C. Hu, An efficient hybrid algorithm for resource-constrained project scheduling, Information Sciences. 2010; 180: 1031-1039.http://dx.doi.org/10.1016/j.ins.2009.11.044

[5] Agarwal, S. Colak, S. Erenguc, “A neurogenetic approach for the resource constrained project scheduling problem”, Computers and Operations Research. 2011; 38: 44-50.http://dx.doi.org/10.1016/j.cor.2010.01.007

[6] V. Valls, F. Ballestin, and M. S. Quintanilla. “Ahybrid genetic algorithm for the resource-constrained project scheduling problem”. European Journal of Operational Research. 2008; 185: 495-508.http://dx.doi.org/10.1016/j.ejor.2006.12.033

[7] R. Kolisch. Serial and parallel resource–constrained project scheduling methods revisited: Theory and computation. European Journal of Operational Research. 1996; 90: 320-333.http://dx.doi.org/10.1016/0377-2217(95)00357-6

[8] Valls, V., Ballest, F., & Quintanilla, S. Justification and RCPSP: A technique that pays. European Journal of Operational Research. 2005; 165: 375-386.http://dx.doi.org/10.1016/j.ejor.2004.04.008

[9] L. Deng, Y. Lin, and M. Chen. "Hybrid ant colony optimization for the resource-constrained project scheduling problem", Journal of Systems Engineering and Electronics. 2010; 21(1): 67-71.

[10] D. Karaboga, B. Basturk, A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm, Journal of Global Optimization. 2007; 39: 459-471.http://dx.doi.org/10.1007/s10898-007-9149-x

[11] R. Akbari, V. Zeighami, and K. Ziarati, “MLGA: A Multilevel Cooperative Genetic Algorithm”, IEEE Fifth International Conference on Bio-Inspired Computing: Theories and Applications (BIC-TA 2010). September 2010; 271-277.

[12] K. Ziarati, R. Akbari, and V. Zeighami, “On the Performance of Bee Algorithms for Resource Constrained Project Scheduling Problem”, Journal of Applied Soft Computing, Elsevier. 2011; 11(4): 3720-3733.http://dx.doi.org/10.1016/j.asoc.2011.02.002

[13] R. Akbari, V. Zeighami, K. Ziarati, Artificial bee colony for resource constrained project scheduling problem, International Journal of Industrial EngineeringComputations. 2011; 2: 45-60.http://dx.doi.org/10.5267/j.ijiec.2010.04.004

[14] J.J. Mendes, J.F. Goncalves, M.G.C. Resende, A random key based genetic algorithm for the resource constrained project scheduling problem, Journal of Computers and Operations Research. 2009; 36: 92-109.http://dx.doi.org/10.1016/j.cor.2007.07.001

[15] D. Debels, B. De Reyck, R. Leus, and M. Vanhoucke. A hybrid scatter search / Electromagnetism meta–heuristic for project scheduling. European Journal of Operational Research, 2004. To appear.

[16] S. Hartmann. A self-adapting genetic algorithm for project scheduling under resource constraints. Naval Research Logistics. 2002; 49: 433-448.http://dx.doi.org/10.1002/nav.10029

[17] S. Hartmann. A competitive genetic algorithm for resource-constrained project scheduling. Naval Research Logistics. 1998; 45: 733-750.http://dx.doi.org/10.1002/(SICI)1520-6750(199810)45:7<733::AID-NAV5>3.0.CO;2-C

[18] Project Scheduling Problem Library. Available from: http://www.129.187.106.231/psplib/.