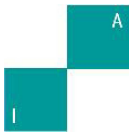
Inteligencia Artificial 22(64) (2019), 123-134
doi: 10.4114/intartif.vol22iss64pp123-134



INTELIGENCIA ARTIFICIAL

http://journal.iberamia.org/

An Enhanced Discrete Bees Algorithms for Resource Constrained Optimization Problems

Mohamed Amine Nemmich[1, A], Fatima Debbat[1, B], Mohamed Slimane[2, C]
[1] Department of Computer Science, University Mustapha Stambouli of Mascara, Algeria
[2] Université de Tours, Laboratoire d'Informatique Fondamentale et Appliquée de Tours (LIFAT), Tours, France
[A] amine.nemmich@gmail.com, [B] debbat_fati@yahoo.fr, [C] mohamed.slimane@univ-tours.fr

Abstract In this paper, we propose two slightly different models based on Bees Algorithm (BA) for the Resource-Constrained Project Scheduling Problem (RCPSP). The studied RCPSP is a NP-hard combinatorial optimization problem which involves resource, precedence, and temporal constraints. It has been applied to many applications. The main objective is to minimize the expected makespan of the project. The proposed Enhanced Discrete Bees Algorithms iteratively solve the RCPSP by utilizing intelligent foraging behaviours of honey bees. The potential solution is represented by the multidimensional bee, where the Activity List representation (AL) is considered. This projection involves using the Serial Schedule Generation Scheme (SSGS) as decoding procedure to construct the active schedules. In addition, the conventional local search of the basic BA is replaced by a neighbouring technique, based on the Swap operator, which takes into account the specificity of the solution space of project scheduling problems and reduces the number of parameters to be tuned. As second variant, the Negative Selection is embedded into our approach to overcome its drawback. The proposed models are tested on well-known benchmark problem instance sets from Project Scheduling Problem Library (PSPLIB) and compared with other approaches from the literature. The promising computational results reveal the effectiveness of the proposed approaches for solving the RCPSP problems of various scales.

Keywords: Optimization, Project scheduling, Resource-constraints, Bees Algorithm, Serial Schedule Generation Scheme, Activity list representation

1 Introduction

Constraint optimization forms an important part of many problems in engineering and industry. Most of the real world optimization problems have constraints of different types which modify the shape of search space. Engineering design, VLSI design, structural optimization, economics, allocation and location problems are just a few examples of fields in which constrained optimization problems are encountered. As constrained optimization problems are difficult to solve due to the presence of various types of constraints (in the form of equalities or inequalities) and their interrelationship between the objective functions [1].

The Resource constrained project scheduling problem (RCPSP) is a well-known computationally intractable combinatorial problem with considerable practical relevance. A vast range of applications in logistics management, manufacturing operations and project management can be modeled as RCPSP such as production planning, scheduling of port operations, school and university timetabling, scheduling of port operations, sports league scheduling, medical research scheduling, assembly shop scheduling, software project, network packet switching and grid computing [2]. The objective of the RCPSP is to schedule the activities in order to minimize the project completion time (makespan) against resource and precedence constraints [3].

Until now, many investigations on RCPSPs have made remarkable progress. Several exact (or deterministic) approaches have been explored firstly such as branch-and-bound procedures, lower bounding, linear programming, mathematical formulations and constraint programming [4]. These procedures are typically used to solve small-sized instances since the execution time needed is impractical when the number of activities increases

ISSN: 1137-3601 (print), 1988-3064 (on-line)

[5]. In addition, constructive heuristics and meta-heuristic methods have been proposed to obtain near-optimum solutions in reasonable amount of time. The constructive heuristics consider two main components, namely the Priority Rule and Schedule Generation Scheme (SGS) [6,7]. Many studies solve the RCPSP with the use of meta-heuristics methods such as Simulated Annealing (SA) [8], Tabu Search (TS) [9], Genetic Algorithm (GA) [10,11,12], Ant Colony Optimization (ACO) [13], Particle Swarm Optimization (PSO) [3,7,14,15], Artificial Bee Colony (ABC) [5,16, 17], Bee Swarm Optimization (BSO) [17], Artificial Immune Systems (AIS) [18], Scatter Search (SS) [19], Multi-Agent Optimization (MAO) [20], etc.

One of the most recent and effective optimization meta-heuristic is the Bees Algorithm (BA). It imitates the foraging behavior of honeybee in nature to solve optimization problems [21]. In this investigation, two modified versions of BA are used and adapted in order to solve the RCPSP.

The rest of this paper is organized as follows. Section 2 provides a description of the RCPSP problem. Section 3 presents the new adaptation of the Bees Algorithm for solving the considered problem. Computational experience and comparative analysis are given in Section 4. Finally, section 5 concludes this work.

2 Description of RCPSP Problem

In this section, the classic single-mode RCPSP is discussed. It is strongly NP-hard [22]. It can be defined as follows. A project involves a set of activities $V = \{0, 1,..., n, n+1\}$ to be scheduled subject to both temporal (precedence) and resource constraints, where each activity has to be processed without preemption in order to complete the project. Each activity j requires a known processing duration d_j ($j = 0,...,n+1$) and consumes $r_{j,k}$ units of resource k for all resource types $k = 1, \ldots, K$ during each period of its duration [6,16]. Each type of resource $k \in K$ has a limited capacity R_k that cannot be exceeded in any time period. A resource can be re-allocated to other activities when that resource is released from finished activities. Let $A(t)$ is the activities set being processed at time instance t. Each activity has precedence constraints. Let P_j be the set of immediate predecessors of activity j. Precedence constraints force each activity j to be scheduled after all predecessor activities P_j are completed [15]. By convention the dummy (pseudo) activities 0 and $n + 1$ refer to the start and the end of schedule (i.e. project), respectively. Activity 0 is the source that has no predecessor while activity $(n + 1)$ has no successor. In addition, these activities have no processing duration and have no resource requirements for any type of resources. The goal of RCPSP is to find optimal schedule with minimal total project duration time [15].

An example of resource constrained project scheduling problem is given in Figure 1. In the diagram, the activities are represented as nodes and the precedences by directed arcs. The project consists of $n = 6$ non-dummy activities which have to be scheduled, consuming $K = 2$ type of renewable resources with a capacity of 4 and 2 units respectively.

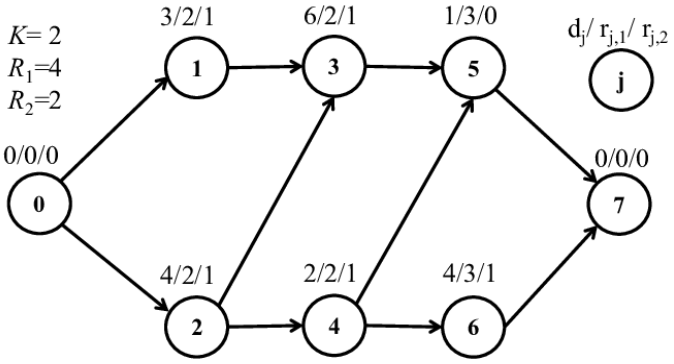


Figure 1: A project example for RCPSP

Let F_j is the finish time of activity j, then a schedule can be represented by a vector of finish times (F_1,...,F_m,...,F_{n+1}). Subsequently, the conceptual model of the RCPSP can be defined as follows [6,16]:

$$\min F_{n+1} \tag{1}$$

$$F_h \leq F_j - d_j \qquad j = 1,\ldots,n+1; h \in P_j \tag{2}$$

$$\sum_{j \in A(t)} r_{j,k} \leq R_k \qquad k \in K; t \geq 0, \tag{3}$$

$$F_j \geq 0 \qquad j = 1,\ldots,n+1. \tag{4}$$

In the RCPSP, the main objective is to identify a feasible schedule that would minimize a project's makespan (Equation1), i.e. the finish time of the last activity processed F_{n+1}, taking into consideration the precedence (Equation 2) and resource limitation (Equation 3) constraints. Equation (4) illustrates the constraint of the decision variables [6,16].

A feasible schedule, of the above problem is represented in Figure 2 with an optimal makespan of 15.

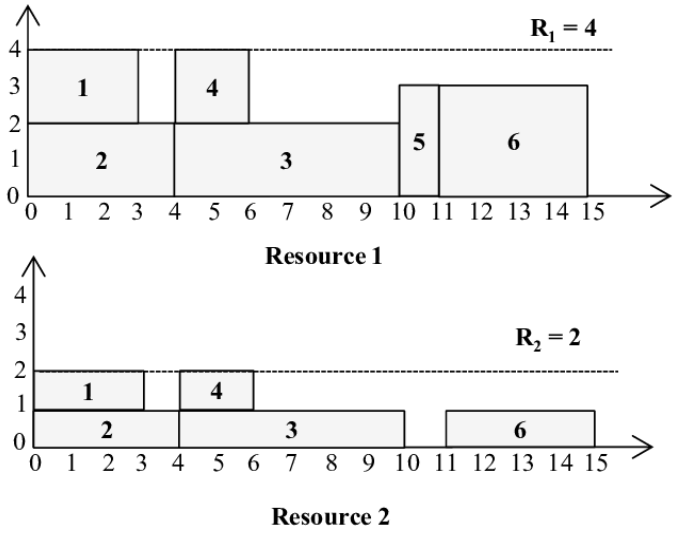


Figure 2: A feasible schedule of the activities

3 Proposed Enhanced Discrete Bees Algorithms for RCPSP

The Bees Algorithm (BA) is a population-based optimization meta-heuristic that mimics the natural foraging behavior of bees. BA was initially designed to solve continuous optimization problems [21]. However, the original BA can be modified and adapted for solving combinatorial problems, such as RCPSP.

Two slightly different algorithms are proposed to solve resource constrained project scheduling problem (RCPSP).

3.1 The First Enhanced Discrete Bees Algorithm (EDBA1)

In the proposed Enhanced Discrete Bees Algorithm (EDBA1), each bee represents a feasible solution for the considered problem, while each activity in a schedule is represented as one component of a bee in the BA process. The representation and decoding process for RCPSP plays a key role in the performance of an approach. In this work, the permutation-based (or the activity list, (AL)) representation was adopted to model solutions of the problem [23], hence the search space is a set of permutations. Each solution is a $1 \times n + 2$ activities vector including dummy activities. The index of a vector determines the order (i.e. relative priority) of an activity among the others. Therefore, a solution (i.e. activity list) is a permutation of activities such that every activity comes in the list before all activities that are its successors in precedence relation. Values are not repeated in the permutations. In this representation, each activity must have a higher index than each of its predecessors. As an

example, consider the feasible schedule of the RCPSP problem described in Figure 1. The solution corresponding is $\lambda = [0, 2, 1, 3, 4, 5, 6, 7]$.

A given solution needs to be decoded and evaluated, but firstly, the solution must be feasible. To do so, the Schedule Generation Schemes (SGS) [24] are generally applied for generating an active schedule from the solution, based on the resource availability and precedence constraints. The SGS is the core decoding procedures for the RCPSP. The basic division distinguishes serial (SSGS) and parallel (PSGS) scheme [6]. In this work, the decoding Serial SGS (SSGS) is adopted, since it always leads to active (i.e. feasible) schedules meaning that using this procedure one is guaranteed to be able to generate an optimal schedule [6]. In addition, the SSGS can be used directly as the decoding procedure for the activity list to obtain the schedule without additional modifications. Kolisch [6] showed that the parallel scheme does not generally perform better than the serial scheme.

The SSGS for Activity List starts from zero to build a schedule by stepwise improvements. The SSGS performs $g = 1,..., n$ iterations, where n is the number of activities. There are two sets of activity in each g state. The scheduled set S_g comprises all the activities which have been already scheduled, and the eligible or decision set D_g, which includes the activities not scheduled for which all predecessor are scheduled. In each iteration one activity j is selected from the decision set D_g and scheduled at it earliest precedence and resource feasible completion time. Next, the decision set of eligible activities and the resource profiles of partial schedule are updated [23]. The scheme terminates on iteration n when all n non dummy activities are scheduled. An application of the SSGS can be recorded by a list $\lambda = [j_1, j_2, \ldots, j_n]$, where j_g represents the activity which is selected and scheduled in iteration g. This list is precedence feasible. The makespan (i.e. project duration) of the solution is given by the maximum completion time of all predecessor activities of activity n+1. Due to the lack of space, SSGS pseudo code is omitted but can be found in [24].

In the proposed approach, each bee represents a solution (i.e. schedule) in a combinatorial space which can be presented as permutation; therefore the search space is a set of permutations. These permutations are placed in the space relating to the order of their elements. Movement in the search space is achieved by changing order of the elements.

The proposed EDBA1 approach starts working with the *ns* scout bees being placed randomly across the search space to generate early solutions, which in fact are lists of activities. The discrete uniform distribution technique is used for randomly generating the solutions during this initialisation in which all possible values have equal probabilities. In order to get feasible solutions in the initial population, the SSGS is called to calculate the new timings for all activities (i.e. schedules an activity in earliest possible time in which both precedence constraints and resource constraints for given activity are met) and acquire the fitness values. Hence, the SSGS eliminates all options which do not respect precedence, and all activities of each solution are successively examined and reordered according to the precedence feasibility condition from left to right. Then, the fitness of the sites visited by the scouts is set to the makespan of the project.

The sampled scouts are ranked, and the m fittest scout bees are picked for local search (i.e. neighborhood search) [21]. To avoid duplication, a neighborhood (called a flower patch or site) is created around every best solution. Amongst these m selected scouts, the top-rated $e \leq m$ (elite) scouts recruit *nre* foragers, and the remaining $m - e$ (non-elite) best scouts recruit $nrb \leq nre$ foragers for local exploration. One neighboring technique based on the swap operator is introduced, eliminating two parameters from the original algorithm which are neighbourhood size (*ngh*) and shrinking factor (*sf*). The patch idea is replaced by this operator to be able to perform a local search and the, shrinking procedure is also removed from the algorithm. However, the abandonment procedure is kept to help the algorithm to improve the global search part. The local search operator is used to generate a variety of new neighboring feasible solutions for the forager bees: performing 'one' swap mutation. In this technique, two activities from each best solution are simply selected at random and their positions are swapped. The algorithm picks the top bee (i.e. solution or schedule) from each site (i.e. flower patch) as the new representative scout of the site to participate in the 'waggle dance' in the next generation. This step is called local search or exploitation [25,26]. If the fitness in a site has stagnated for a predefined number *stlim* of iterations, the scout associated to the flower patch is abandoned (i.e. the local fitness peak has been attained) and randomly re-initialized to a new position in the solution space. If the abandoned position corresponds to the best-so-far solution, it is recorded.

The remaining ($ns - m$) scout bees are assigned to look for any new potential solutions via random sampling of the solution space, and hopefully escaping from local minima. This step is called global search, i.e. exploration [25].

At the end of each iteration, the population of the bee colony is updated and formed out of two groups. The first group comprises the m bees related to the best solutions (schedules) of the high-fitness flower patches (the results of the local exploitative search), and the second group comprises $ns - m$ scouts assigned to conduct random searches (the results of the global explorative search) [26,27]. Sequences of local and global search are repeated

until one of the two stopping criteria is met: an adequate makespan is revealed, or a given number of optimization cycles have passed [21]. Figure 3 illustrates the pseudo code of the EDBA employed for solving the RCPSP problem.

The EDBA1 explores their neighborhoods in a selective manner to discover the most promising solutions, and find the global minimum of the objective function which is the minimal makespan (total completion time) schedule of the project. Therefore, maintaining the diversity and making tradeoff between diversification (i.e. global exploratory search) and intensification (i.e. local exploitative search) is indispensable to produce high-quality solutions.

It is clear that our algorithm does not require complex improvement techniques to obtain results. Our goal is to design an approach that is easily adaptable to the RCPSP and different variety of combinatorial optimization problems.

Enhanced Discrete Bees Algorithm (EDBA) for RCPSP

1. Initialize population with random solutions
2. Generate and evaluate initial feasible solutions from the population by using *SSGS*
   //Forming new population.
3. While (stopping criterion not met)
   a. Select sites for neighborhood search.
   b. Recruit bees for selected sites using *neighboring technique* (more bees for best e sites): use *SSGS* to generate and evaluate feasible solutions.
   c. Select the fittest bee (i.e. a schedule with smaller makespan) from each patch.
   d. Assign remaining bees to search randomly: use *SSGS* to generate and evaluate feasible solutions.
4. End While.
5. Return the best feasible schedule found so far;

Figure 3: Pseudo code of the proposed EDBA1 for RCPSP problem

3.2 The Second Enhanced Discrete Bees Algorithm (EDBA2)

In the first Enhanced Discrete Bees Algorithm (denoted by EDBA1), it is possible that the algorithm retains duplicate solutions, which means that each selected site may sometimes unintentionally produce the same sequences as other sites during the local search. Furthermore, there is a probability that the second best solution, and sometimes even the third best solution on a site, may have better fitness values than the best solution of other sites. Keeping deceitful or duplicitous solutions for the next generation could cause high computational time as well as difficulties at the local optimum.

To overcome this disadvantage, our idea is to introduce the Negative Selection in Artificial Immune Systems. This process is based on the principles of T cell maturation and self / non-self-discrimination in biological immune systems [28]. The objective is to improve the algorithm in choosing the best solutions from selected sites after the local neighborhood research for the next generation. This new version is denoted by EDBA2.

In the EDBA1, after the neighborhood search, the best bee of each site will be recorded for the next iteration. For the EDBA2, all the solutions derived from the local search will be sorted (according to their fitness values) and transferred to the repertoire (R0) and the best bees are selected by the Negative Selection process. Figure 4 presents the local search procedure of the second Enhanced Discrete Bees Algorithm (EDBA2). Figure 5 shows the adopted Negative Selection model.

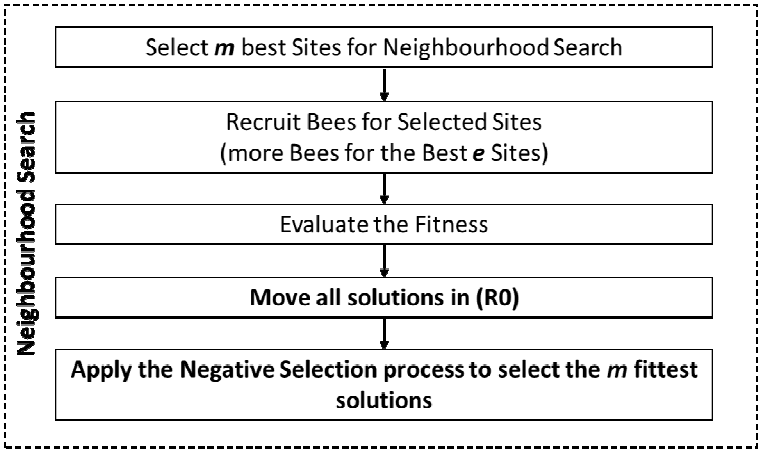


Figure 4: Local search of EDBA 2

Two major phases of Negative Selection process, detector generation and anomaly monitoring, are therefore considered. The first solution in repertoire (R0) will be copied into the Self-strings (S) and into the repertoire (R). Then, the next solution in the repertoire (R0) will be considered by associating it with strings in (S). If it is recognized, it will be eliminated. Otherwise, it will be introduced in the repertoire (R). In the opposite direction, if all the strings of (S) do not correspond to the solution from the repertoire (R0), then it is eliminated and replaced by the solution of the repertoire (R0).

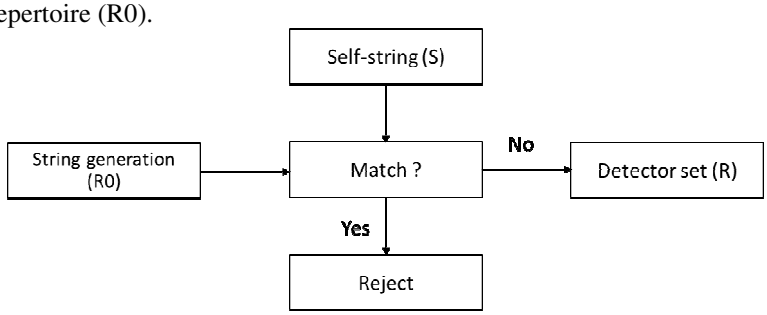


Figure 5: Negative Selection model

4 Computational Experiments

The proposed EDBA algorithms were implemented using Octave programming language and all experiments were performed on a 3.40 GHz Intel Core i3 3240 CPU and 4 GB RAM running a 64-bit operating system. The standard PSPLib (Project Scheduling Problem Library) test problems (i.e. J30, J60, J90 and J120) [29] were used for evaluating and comparing the EDBA approaches with existing algorithms from the literature. J30, J60 and J90 consist of 480 instances with 30, 60 and 90 non-dummy activities, respectively. J120 consists of 600 instances with 120 non-dummy activities. Hence, there are total 2040 RCPSP instances are simulated in this investigation. All projects involved 4 renewable resources, and each activity with a maximum of 3 successors. Table 1 illustrates the 30 activities example instance j301_6 in PSPLIB. In this instance case, there are 4 types of resource, 30 activities and sufficient processors available are assumed.

The numbers of generated solutions (schedules) are set as 1000, 5000 and 50000 for each instance. The parameters' settings of the EDBA are listed in Table 2. Values of parameters used in numerical experiments were selected, based on those recommended by our previous investigations [30,31].

The success rate was used to measure the performance of the algorithms. It shows the number of instances in a case study which are successfully solved by the algorithms. For an instance, an algorithm is called successful if it can find the optimal schedule which has the minimum makespan.

A set of simulations has been carried out to verify how many cases of PSPLIB library can be solved by the proposed EDBA. The objective is to find optimal solution (for J30) or lower bound solution (for J60, J90 and J120).

Table 1: 30 activities case (j301_6) with precedence and resource requirement constraints [29]

Activity#	Successors			Activity#	Duration	Required resources			
						R1	R2	R3	R4
1	2	3	4	1	0	0	0	0	0
2	5	7	8	2	10	0	0	0	4
3	11			3	1	0	0	0	10
4	6	16		4	9	4	0	0	0
5	15	23		5	3	6	0	0	0
6	10	12		6	1	3	0	0	0
7	9	14	25	7	7	0	4	0	0
8	13			8	1	0	0	0	2
9	24			9	4	10	0	0	0
10	22			10	10	0	0	0	2
11	14	16	24	11	6	0	0	10	0
12	13	21		12	2	0	0	0	6
13	17	24	30	13	3	0	7	0	0
14	18			14	1	0	0	3	0
15	16	29		15	3	0	0	0	6
16	19			16	1	0	0	10	0
17	18			17	3	0	0	0	7
18	20	31		18	10	0	0	0	9
19	28			19	1	0	6	0	0
20	26			20	3	5	0	0	0
21	28			21	4	0	3	0	0
22	28			22	2	8	0	0	0
23	27			23	4	1	0	0	0
24	26	31		24	2	3	0	0	0
25	30			25	4	0	9	0	0
26	29			26	6	0	0	0	7
27	30			27	9	0	0	0	7
28	31			28	2	0	0	0	5
29	32			29	1	0	0	9	0
30	32			30	1	0	0	9	0
31	32			31	9	0	0	4	0
32				32	0	0	0	0	0
Available resources						12	10	10	12

Table 2: Parameters Setting [30,31]

Algorithm	Parameters	Value
EDBA	Scout bees (ns)	12
	Elite sites (e)	2
	Best sites (m)	6
	Recruited bees for elite sites (nre)	29
	Recruited bees for remaining best sites (nrb)	9
	Limit of stagnation cycles for site abandonment (stlim)	10
	Number of generation schedules (genmax)	1000/5000/50000

Tables 3-6 present the computational results, comparing the proposed EDBA1 and EDBA2 algorithms with 11 other approaches for the four instance sets of RCPSP. Each algorithm is briefly defined by a few keywords, the SGS employed, and the reference. The best results for each instance set are shown in boldface; the second best is in italics.

From the results, it can be seen that the success rate values decrease (i.e. the performance of the proposed models decrease) as the problem size increase. This involves that the RCPSP with larger scale is more difficult to

solve since the problem is NP-hard problem. Furthermore, for each set of instances, the success rate values increase as the schedules number increase.

The percentage of problem instances successfully solved by EDBA1 varies in range of [76.88%, 93.96%], [67.08%, 77.71%], [65.63%, 74.58%] and [17.17%, 31.67%] for J30, J60, J90 and J120, respectively. Whereas, the success rates of the second proposed method EDBA2 are between 89.21% and 90.62% for J30, 67.29% and 76.04% for J60, 67.08% and 74.38% for J90, and 18.67% and 31.50% for J120.

Table 3 presents the success rates for the instance set J30 in which all problem instances have been solved to optimality. The results show that the proposed EDBA1 obtains the fifth rank among the 13 existing heuristics after EDBA2, BA, ACROSS and BSO for 1000 schedules and obtains the first rank after 5000 and 50000 schedules, respectively. The second improved algorithm EDBA2 obtained the first after 1000 and the second after 5000 and the sixth after 50000 schedule generations, respectively.

Table 3: Average success rate (%) from optimal makespan for J30

Algorithm / SGS	**SGS**	**Reference**	**Schedules**		
			1000	**5000**	**50000**
GA	Serial	Hartmann [10]	51.87	53.75	58.96
PSO	Serial	Zhang et al. [14]	53.54	58.13	61.26
ANGEL	Serial	Tseng and Chen [32]	71.46	78.70	89.11
GAPS	Parameterized active schedule	Mendes et al. [12]	57.30	63.29	68.33
ACOSS	Serial & Parallel	Chen et al. [15]	77.41	85.04	***93.27***
Neurogenetic	Serial & Parallel	Agrawal et al. [33]	74.13	81.33	91.05
OOP-GA	Serial	Montoya-Torres et al. [11]	53.19	57.48	65.82
PSO+	Novel SGS	Chen et al. [15]	67.80	75.05	87.15
ABC	Serial / Priority list	Akbari et al. [5]	72.71	80.84	90.42
BSO	Serial / Priority list	Ziarati et al. [17]	77.30	85.63	92.09
BA	Serial / Priority list	Ziarati et al. [17]	***78.54***	86.25	92.50
EDBA 1	Serial / Activity list	Proposed approach 1	76.88	**89.58**	**93.96**
EDBA 2	Serial / Activity list	Proposed approach 2	**80.21**	***87.71***	90.62

Table 4 shows the lower bound results for instances with 60 activities (J60). Due to the larger number of activities, the proposed approach and the other ones have more difficulty to solve. The success rates decrease compared to J30 case study. In an experiment of 1000 and 5000 schedules, EDBA2 scheme has been proven to be better than other algorithms followed by EDBA1 as second best. After 50000 schedules, EBDA1 is the most efficient method and EBDA2 became the second for this case study.

Table 4: Average success rate (%) from lower bound for J60

Algorithm / SGS	**SGS**	**Reference**	**Schedules**		
			1000	**5000**	**50000**
GA	Serial	Hartmann [10]	43.33	52.30	55.84
PSO	Serial	Zhang et al. [14]	45.21	53.96	58.55
ANGEL	Serial	Tseng and Chen [32]	53.95	61.33	64.91
GAPS	Parameterized active schedule	Mendes et al. [12]	46.39	52.40	56.07
ACOSS	Serial & Parallel	Chen et al. [15]	55.12	60.94	65.21
Neurogenetic	Serial & Parallel	Agrawal et al. [33]	56.37	63.86	68.35
OOP-GA	Serial	Montoya-Torres et al. [11]	48.52	54.04	59.26
PSO+	Novel SGS	Chen et al. [15]	58.66	62.05	66.76
ABC	Serial / Priority list	Akbari et al. [5]	61.88	67.09	71.88
BSO	Serial / Priority list	Ziarati et al. [17]	64.38	70.63	71.88
BA	Serial / Priority list	Ziarati et al. [17]	66.25	68.34	71.67
EDBA 1	Serial / Activity list	Proposed approach 1	***67.08***	***71.25***	**77.71**
EDBA 2	Serial / Activity list	Proposed approach 2	**67.29**	**71.67**	***76.04***

The J90 simulation results are shown in Table 5. The proposed EDBA2 and EDBA1 in this study rank first and second, respectively, on the comparison table for 1000 and 5000 schedules. When 50000 schedules are examined, our EDBA1 outperforms all the approaches used in this comparative study, followed by the EDBA2 algorithm.

Table 5: Average success rate (%) from lower bound for J90

Algorithm / SGS	**SGS**	**Reference**	**Schedules**		
			1000	**5000**	**50000**
GA	Serial	Hartmann [10]	39.95	44.30	53.96
PSO	Serial	Zhang et al. [14]	42.63	49.34	55.27
ANGEL	Serial	Tseng and Chen [32]	53.08	58.23	61.49
GAPS	Parameterized active schedule	Mendes et al. [12]	45.85	48.72	54.21
ACOSS	Serial & Parallel	Chen et al. [15]	55.72	58.10	62.03
Neurogenetic	Serial & Parallel	Agrawal et al. [33]	57.64	60.53	64.82
OOP-GA	Serial	Montoya-Torres et al. [11]	47.44	50.58	56.84
PSO+	Novel SGS	Chen et al. [15]	61.65	65.24	67.17
ABC	Serial / Priority list	Akbari et al. [5]	60.13	65.21	68.75
BSO	Serial / Priority list	Ziarati et al. [17]	64.59	69.17	69.59
BA	Serial / Priority list	Ziarati et al. [17]	64.80	67.09	70.00
EDBA 1	Serial / Activity list	Proposed approach 1	***65.63***	***71.25***	**74.58**
EDBA 2	Serial / Activity list	Proposed approach 2	**67.08**	**71.88**	***74.38***

The last case study with 120 activities is the most challenging to solve. Table 6 displays the success rates for this case study. The investigated approaches have more difficulty and their performance decrease drastically. The results demonstrate that the EDBA algorithms had good and competitive performance compared with other algorithms. The first and the second rank are obtained by the proposed EDBA2 and BA, respectively, after 1000 schedule generations. The two proposed EDBA algorithms perform equally and achieve the first best performance when the number of schedules is 5000. The best success rates after 50000 schedules are obtained by EDBA1 approach. The second rank is obtained by the second version of the proposed algorithm.

Table 6: Average success rate (%) from lower bound for J120

Algorithm / SGS	**SGS**	**Reference**	**Schedules**		
			1000	**5000**	**50000**
GA	Serial	Hartmann [10]	7.15	9.18	16.33
PSO	Serial	Zhang et al. [14]	9.44	11.89	20.42
ANGEL	Serial	Tseng and Chen [32]	14.85	15.36	19.91
GAPS	Parameterized active schedule	Mendes et al. [12]	10.12	12.83	18.78
ACOSS	Serial & Parallel	Chen et al. [15]	14.56	17.72	20.69
Neurogenetic	Serial & Parallel	Agrawal et al. [33]	14.32	16.85	21.08
OOP-GA	Serial	Montoya-Torres et al. [11]	10.39	13.51	19.82
PSO+	Novel SGS	Chen et al. [15]	14.60	16.48	21.26
ABC	Serial / Priority list	Akbari et al. [5]	15.34	18.97	22.84
BSO	Serial / Priority list	Ziarati et al. [17]	17.00	22.50	25.17
BA	Serial / Priority list	Ziarati et al. [17]	***17.84***	20.84	24.50
EDBA 1	Serial / Activity list	Proposed approach 1	17.17	**25.00**	**31.67**
EDBA 2	Serial / Activity list	Proposed approach 2	**18.67**	**25.00**	***31.50***

Different models of bee algorithms have been proposed in literature such as Artificial Bee Colony (ABC) [5], Bees Algorithm (BA) [17] and Bee Swarm Optimization (BSO) [17] to solve the RCPSP (Tables 3-6). Each approach uses different types of bees to provide appropriate level of exploration over search space while maintaining exploitation of good solutions. The three bee algorithms use the priority-based representation [14] for their individuals and the SSGS to generate the schedule.

We can see from the results that our proposed EDBA algorithms which uses the activity-list as a default representation of a solution commonly offers better results than other bee algorithms using random-key representation schemes in majority of cases. In contrast, the priority list-based BA, followed by BSO surpasses

our EDBA1 approach over j30 for 1000 schedules and EDBA2 over j120 for 50000 schedules. In addition, BA provides better performance than EDBA1 approach for j120 when we set number of schedules at 1000.

To sum up, the best results were obtained by the proposed EDBA1 and/or EDBA2 algorithms. EDBA2 ranks first, compared to the other approaches, with 6 out of 12 case studies, followed by EDBA1 in 5 out of 12 cases. The two algorithms give the same result in only one case study (J120 with 5000 schedules).

EDBA2 is a good alternative if the number of schedules generated is not too high. Otherwise, EDBA1 is the best option.

In brief, the proposed EDBA algorithms are highly ranked between state-of-the-art algorithms and reveals robust competitiveness as an effective technique in solving medium and large-scale RCPSP. Such level of improvement is associated with the BA intelligent behavior, the use of the permutation based representation formalism for the schedules, the employment of serial SGS in order to generate feasible schedules for the RCPSP, and the application of move operators for local search.

5 Conclusion

This paper presents two Enhanced Discrete Bees Algorithms for solving the classical resource constrained project scheduling (RCPSP) with the makespan minimization criterion. The proposed algorithms, which are based on foraging behaviors of honey bees, use a population of different types of bees to find the optimal solution. Its codifications are based on the activity-list representation schemes with the Serial Schedule Generation Scheme (SSGS) to yield feasible solutions. In local search, we use one structure of neighborhoods, called swap operator. In addition, the site abandonment procedure is used to maximize the diversity of the population and to push down the solution from local optimum. In second variant, the Negative Selection process is introduced in order to obtain better performance. The improved algorithms are developed with reduction of parameters to be tuned.

The performance of our approaches has been evaluated on the well know PSPLIB instances. The computational experiments confirm that the proposed models produce consistently high-quality solutions and it is more effective than other bee inspired algorithms and several well-known heuristics from the literature. Moreover, the use of negative selection rules has demonstrated their effectiveness to boost the solution quality of RCPSP if the number of schedules to be analyzed is low.

As future research, the proposed approach could be used for solving other important variants of the RCPSP, e.g. multiple modes and multiple projects, and different kinds of hard combinatorial optimization problems. Also, hybridization with the other heuristic or meta-heuristics may offer a way to improve their efficiency.

References

[1] Harish Garg. A hybrid PSO-GA algorithm for constrained optimization problems. *Applied Mathematics and Computation,* 274: 292–305, 2016.

[2] Sönke Hartmann and Dirk Briskorn. A survey of variants and extensions of the resource-constrained project scheduling problem. *European Journal of Operational Research*, 207(1):1-14, 2010.

[3] Ruey-Maw Chen. Particle swarm optimization with justification and designed mechanisms for resource-constrained project scheduling problem. *Expert Systems with Applications,*38(6):7102–7111, 2011.

[4] Olivier Liess and Philippe Michelon. A constraint programming approach for the resource-constrained project scheduling problem. *Annals of Operations Research,*157:25–36, 2007.

[5] Reza Akbari, Vahid Zeighami and Koorush Ziarati. Artificial Bee colony for resource constrained project scheduling problem. *International Journal of Industrial Engineering Computations,*2(1):45–60, 2011.

[6] Rainer Kolisch. Serial and parallel resource-constrained project scheduling methods revisited: Theory and computation. *European Journal of Operational Research,* 90(2):320–333, 1996.

[7] Amer Fahmya, Tarek M Hassanb and Hesham Bassioni. Improving RCPSP solutions quality with Stacking Justification – Application with particle swarm optimization. *Expert Systems with Applications,* 41(13):5870–5881, 2014.

[8] Kamel Bouleimen and Lecocq H. A new efficient simulated annealing algorithm for the resource-constrained project scheduling problem and its multiple mode version. *European Journal of Operational Research*,149(2):268–281, 2003.

[9] Ikonomou A, John Galletly and Daniel RC. Solving resource-constrained project scheduling problems using Tabu Search. In *International Conference on Intelligent Systems for Manufacturing IFIP Advances in Information and Communication Technology*, pages 311–322. Springer, 1998

[10] Sönke Hartmann. A competitive genetic algorithm for resource-constrained project scheduling. *Naval Research Logistics*.;45(7):733–750, 1998

[11] Jairo R.Montoya-Torres, Edgar Gutierrez-Franco and Carolina Pirachicán-Mayorga. Project scheduling with limited resources using a genetic algorithm. *International Journal of Project Management*. 2010;28(6):619–628.

[12] Jorge J M Mendes, José F Gonçalves and Mauricio G C Resende. A random key based genetic algorithm for the resource constrained project scheduling problem. *Computers & Operations Research*, 36(1):92–109, 2009

[13] Daniel Merkle, Martin Middendorf and Hartmut Schmeck. Ant colony optimization for resource-constrained project scheduling. *IEEE Transactions on Evolutionary Computation,* 6(4):333–346, 2002

[14] Hong Zhang, Xiaodong Li, Heng Li and Fulai Huang. Particle swarm optimization-based schemes for resource-constrained project scheduling. *Automation in Construction,*14(3):393–404, 2005

[15] Ruey-Maw Chen, Chung-Lun Wu, Chuin-Mu Wang and Shih-Tang Lo. Using novel particle swarm optimization scheme to solve resource-constrained scheduling problem in PSPLIB. *Expert Systems with Applications*, 37(3):1899–1910, 2010.

[16] Yan-jun Shi, Fu-Zhen Qu, Wang Chen and Bo Li. An artificial bee colony with random key for resource-constrained project scheduling. In *International Conference on Intelligent Computing for Sustainable Energy and Environment. ICSEE 2010*, pages 148–157. Springer, 2010.

[17] Koorush Ziarati, Reza Akbari and Vahid Zeighami. On the performance of bee algorithms for resource-constrained project scheduling problem. *Applied Soft Computing*, 11(4):3720–3733, 2011.

[18] Rina Agarwal, Manoj K Tiwari and Sanat K Mukherjee. Artificial immune system based approach for solving resource constraint project scheduling problem. *The International Journal of Advanced Manufacturing Technology*, 34(5-6)584–593, 2006.

[19] Dieter Debels, Bert De Reyck, Roel Leus and Mario Vanhoucke. A hybrid scatter search/electromagnetism meta-heuristic for project scheduling. *European Journal of Operational Research,* 169(2):638–653, 2006.

[20] Xiao-long Zheng and Ling Wang. A multi-agent optimization algorithm for resource constrained project scheduling problem. *Expert Systems with Applications*, 42(15-16): 6039–6049, 2015.

[21] Duc T Pham, Afshin Ghanbarzadeh, Ebubekir Koç, Sameh Otri, Shafqat Rahim and Monji Zaidi. The Bees Algorithm — A novel tool for complex optimisation problems. In *Proceedings of International Conference on Intelligent Production Machines and Systems*, pages 454–459, 2006

[22] Jacek Blazewicz, Jan K Lenstra and AHG R Kan. Scheduling subject to resource constraints: classification and complexity. *Discrete Applied Mathematics,* 5(1):11–24, 1983.

[23] Rainer Kolisch and Sönke Hartmann. Experimental investigation of heuristics for resource-constrained project scheduling: An update. *European Journal of Operational Research,*174(1):23–37, 2006.

[24] Rainer Kolisch and Sönke Hartmann. Heuristic algorithms for the resource-constrained project scheduling problem: classification and computational analysis. *Project Scheduling International Series in Operations Research & Management Science*, 14:147–178, 1999.

[25] Duc T Pham and Marco Castellani. The Bees Algorithm: Modelling foraging behaviour to solve continuous optimization problems. In *Proceedings of the Institution of Mechanical Engineers, Part C*: *Journal of Mechanical Engineering Science*, 223(12):2919–2938, 2009

[26] Mohamed A Nemmich, Fatima Debbat and Mohamed Slimane. A data clustering approach using bees algorithm with a memory scheme. In *Demigha O, Djamaa B, Amamra A, editor. CSA 2018: Advances in Computing Systems and Applications*, pages 261–270. Springer, 2018.

[27] Mohamed A Nemmich and Fatima Debbat. Bees algorithm and its variants for complex optimisation problems. In: *The 2nd International Conference on Applied Automation and Industrial Diagnostics (ICAAID 2017)*. Djelfa, 2017.

[28] Stephanie Forrest, Lawrence Allen, Alan S Perelson and Rajesh Cherukuri. Self-nonself discrimination in a computer. In *IEEE Computer Society Symposium on Research in Security and Privacy*, pages 202-212. IEEE, 1994.

[29] Rainer Kolisch, Christoph Schwindt and Arno Sprecher. Benchmark instances for project scheduling problems. *Project Scheduling International Series in Operations Research & Management Science,* 14:197–212,1999

[30] Mohamed A Nemmich, Fatima Debbat and Mohamed Slimane. A novel hybrid firefly bee algorithm for optimization problems. *International Journal of Organizational and Collective Intelligence*, 8(4):21–46, 2018.

[31] Mohamed A Nemmich, Fatima Debbat and Mohamed Slimane. A Permutation-Based Bees Algorithm for Solving Resource-Constrained Project Scheduling Problem. *International Journal of Swarm Intelligence Research*, 10(4):1-24, 2019.

[32] Lin-Yu Tseng and Shih-Chieh Chen. A hybrid metaheuristic for the resource-constrained project scheduling problem. *European Journal of Operational Research,*175(2):707–721, 2006.

[33] Anurag Agarwal, Selcuk Colak and Selcuk Erenguc. A Neurogenetic approach for the resource-constrained project scheduling problem. *Computers & Operations Research.* 38(1):44–50, 2011.