Decision Science Letters 4 (2015) 537–550

Contents lists available at GrowingScience

Decision Science Letters

homepage: www.GrowingScience.com/dsl

A study on the performance of differential search algorithm for single mode resource constrained project scheduling problem

Nazanin Rahmani[a], Vahid Zeighami[b] and Reza Akbari[c*]

[a]*College of Engineering, Science and Research Branch, Islamic Azad University, Yasuj, Iran,*
[b]*Department of Mathematics and Industrial Engineering, Ecole Polytechnique, de Montreal, Montreal, Quebec, Canada*
[c]*Department of Computer Engineering and Information Technology, Shiraz University of Technology, Shiraz, Iran*

C H R O N I C L E

Article history:
Received March 29, 2015
Received in revised format:
May 12, 2015
Accepted May 12, 2015
Available online
May 18 2015

Keywords:
Differential search algorithm
Resource constrained project scheduling problem
Single mode

A B S T R A C T

Differential Search (DS) algorithm is a new meta-heuristic for solving real-valued numerical optimization. This paper introduces a new method based on DS for solving Resource Constrained Project Scheduling Problem (RCPSP). The RCPSP is aimed to schedule a set of activities at minimal duration subject to precedence constraints and the limited availability of resources. The proposed method is applied to PSPLIB case studies and its performance is evaluated in comparison with some of state of art methods. Experimental results show that the proposed method is effective. Also, it is among the best algorithms for solving RCPSP.

1. Introduction

The resource constrained project scheduling problem (RCPSP) is an important problem in project management, manufacturing and resource optimization. The RCPSP occurs frequently in high scale projects management such as software development, construction of power plants, industrial projects, etc. (Hartmann & Kolisch, 2000; Kolisch & Hartmann, 2006). RCPSP can be separated in different classes such as single mode RCPSP and multi-mode RCPSP with non-regular objective functions, stochastic RCPSP, bin-packing-related RCPSP problems, and multi resource constrained project scheduling problem. A comprehensive study on different types of RCPSP has been presented by Yang et al. (2001). In single mode RCPSP, a project consisting of a set of activities with fixed durations and resource requirements is considered (Zeighami et al., 2013). In multi-mode RCPSP, each activity can be executed under different durations with renewable and non-renewable resources (Coelho & Vanhoucke, 2011). In stochastic RCPSP, activity durations are not deterministic because during the project implementation there may be a series of random factors affecting the duration of activities

* Corresponding author.
E-mail address: akbari77@gmail.com (R. Akbari)

doi: 10.5267/j.dsl.2015.5.005

(Zheng et al., 2014). In RCPSP with non-regular objective functions, the goal is to minimize the activities' costs (Neumann & Zimmermann, 1999). In bin-packing-related RCPSP problems, the resource capacity represents the bin size, while a task's resource consumption requirement represents an item size (Kumar, 2014). In multi-resource-constrained project scheduling problems, a job may require a set of operations, or a set of successive resources (Kumar, 2014).

In this work, the basic single mode RCPSP is considered. This type of RCPSP is a difficult problem to solve. Limitation of resources and precedence constraint makes it difficult. The RCPSP consists of executing a group of activities limited by constraints. Processing every activity requires predefined amount of resources. Every project has its own precedence constraints, which means that each activity can be processed when all its predecessors are finished. Otherwise, an activity cannot start before the completion of all of its predecessors. In general, the purpose of project schedules is to minimize its completion time or makespan (S_{n+1}), subject to precedence and resource constraints (Kolisch & Hartmann, 2006). In recent years, different types of algorithms (e.g. exact, heuristics, and meta-heuristics) have been proposed to solve single-mode RCPSP. A comprehensive survey on project scheduling under resource constraint has been presented in (Orji & Wei, 2013). Like other NP-hard problems, exact methods are no efficient in solving large-sized RCPSP problems. The exact methods are suitable for small-sized RCPSP problems.

To solve large-sized RCPSP, alternative methods are required and the meta-heuristic methods can be used (Hartmann & Kolisch, 2000; Kolisch & Hartmann, 2006; Kolisch & Hartmann, 1999; Kolisch & Padman, 2001). These methods have the ability to generate near-optimal solutions even for large-sized RCPSP problems. It is possible to categorize these methods in two classes. The first class includes methods that keep one solution during each iteration. For example tabu search (Baar et al., 1998) and simulated annealing (Bouleimen & Lecocq, 2003) set in this class. These methods start by single solution and try to improve the solution, iteration by iteration until the termination condition is met. The second class includes methods that keep a set of solution during each iteration and try to solve the problem with a population of individuals. In recent years, many population and swarm based optimization algorithms have been presented in literature that can be used to solve RCPSP. Most of representative methods in this field are based on genetic algorithms, particle swarm optimization and bee algorithms. Genetic algorithms have been used to solve RCPSP (Hartmann, 1998; Hartmann, 2002; Mendes et al., 2009; Ranjbar et al., 2008). These methods showed the efficiency in solving single mode RCPSP. A magnet based crossover operator was used by Zamani (2013) to improve the performance of genetic algorithm in solving RCPSP. Different variants of particle swarm optimization have been used to solve RCPSP (Jarboui et al., 2008; Luo et al. 2006; Zhang et al., 2008). Recently, a particle swarm optimization (PSO) based hyper-heuristic algorithm for solving RCPSP has been presented by Koulinas et al. (2014). The hyper-heuristic is aimed to work as an upper-level algorithm that controls several low-level heuristics which operates to the solution space. The multiple justification particle swarm optimization (MJPSO) using stacking justification for further improvement has been presented by Fahmy et al. (2014). A hybrid particle swarm optimization procedure to solve the preemptive RCPSP in which a maximum of one interruption per activity is allowed has been presented by Shou et al. (2015). A pseudo PSO (P-PSO) has been introduced by Nasiri (2012) to cope with the complexity of scheduling problem. In P-PSO, particles use the path relinking procedure to fly toward local and global best positions. The improved PSO presented by Jia and Seo (2013) uses particle swarm and employs a double justification and an operator for particle movement along with rank-priority-based representation, greedy random search, and serial scheduling scheme.

Three variants of bee algorithms called Bees Algorithm (BA), Artificial Bee Colony (ABC), and Bee Swarm Optimization (BSO) have been used to solve RCPSP (Ziarati et al., 2011; Akbari et al., 2012). Also, the facility layout problem (FLP) concept and integration with the permutation-based artificial bee colony (PABC) algorithm has been used by Jia and Seo (2013) for RCPSP. Beyond the methods considered in above paragraphs, other meta-heuristic methods have been used to solve RCPSP. An Activity-List based Nested Partitions (ANLP) algorithm for solving RCPSP was presented by Xiao et al. (2014). This method partitions the feasible solution space which is formulated by activity-lists into sub-regions by the nested partitions approach. Ant colony optimization (Merkle et al., 2002) is another swarm-based optimization algorithm used to solve RCPSP. Firefly algorithm is known as another meta-

heuristic methods which has been used by Sanaei et al. (2013) for scheduling problems. Generally, different types of meta-heuristics have been used to solve this type of RCPSP. DS is one of the most recently introduced meta-heuristics by Pinar Civicioglu (2012) to solve optimization problems. DS is based on simulation of Brownian-like random-walk movement used by an organism to migrate. The DS has been designed based on the superorganism migration. Using the migration aliving being such as superorganism can find food, protection, in new fertile position. Amount of fertility can be used to inspire an algorithm from the behaviors of superorganism migration. It seems that DS has the ability to solve complex engineering problems. This work is aimed to study the performance of DS algorithm in solving RCPSP problems.

The remaining of this paper is organized as follows: the problem formulation is given in Section 2. In Section 3, the basic concepts of DS algorithm is presented. Section 4 presents the details of the proposed method for solving RCPSP problems. The experimental results are given in Section 5. Finally, section 6 concludes this work.

2. Problem formulation

The single mode resource constrained project scheduling is a type of constrained optimization problems that can be defined as follows:

Assume that we have a project Prj which is shown as a directed acyclic Graph $G = (A, E)$ where A represents a set of activities of the project which are represented by nodes and E shows the precedence relationships among the activities which are represented by edges of the graph. As an example, a project with nine activities is shown in Fig. 1. We assume that the project has $n + 1$ activities and K renewable resource type $R = \{R_1, R_2, \dots, R_K\}$ where R_i is the finite capacity of resource type i. $A = \{A_0, A_1, A_2, \dots, A_{n+1}\}$ is a set of $n + 1$ activities. A_0 and A_{n+1} are dummy activities (e.g. activities A_0 and A_{10} in Figure 1). They specify start and end of the project. When all activities are processed, the project is completed. Each activity A_i has fixed duration d_i. $D = \{d_0, d_1, d_2, \dots, d_{n+1}\}$ is a set of durations. For dummy activities A_0 and A_{n+1}, duration is zero ($d_0 = d_{n+1} = 0$). Each activity A_i requires r_{ij} units of resource R_j during each period of its execution (Kolisch & Hartmann, 1999). The time and resource requirements of each activity in Figure 1 is given in Table 1. Here, we assume that each activity needs only one resource R_1 with 7 instances.

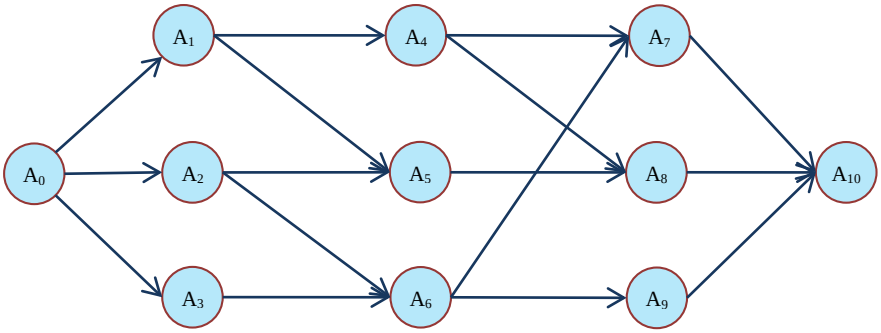

Fig. 1. A project with nine activities

Dummy activities do not require any resources ($r_{0j} = r_{n+1,j} = 0$ where $j \in \{1,2,3, \dots, K\}$). Precedence between activities (edges in Graph) are represented by E. Set of pairs such as $(A_i, A_j) \in E$ means that activity A_i precedes activity A_j. A_i is finished before A_j is started. Considering the precedence limitation, we assume that $S = \{S_0, S_1, \dots, S_{n+1}\}$ is a feasible schedule where S_i is the start time of activity A_i. The objective is to find an ordering of the activities that minimizes the makespan of the schedule S_{n+1} under resource and precedence constraints. The problem can be modeled as:

$$Minimize \quad Makespan\ (S_{n+1}) \tag{1}$$

The resource limitation constraint is describe as:

$$\sum_{i \in P(t)} r_{ij} \leq R_j \; j = 1,2, \dots, K \;\; and \;\; 0 \leq t \leq S_{n+1} \tag{2}$$

where $P(t) = \{i \in A | S_i \leq t \leq S_i + d_i\}$ and t is the specified time. Set P presents all the activities in time t that can be processed. These activities were started but they were not finished. Precedence limitation can be described as:

$$S_i + d_i \leq S_j \;\; S_i, S_j \in S \;\; and \;\; (A_i, A_j) \in E \tag{3}$$

According to this equation, if A_i precedes activity A_j then A_j starts when A_i is completely executed. For given example, the minimum makespan is 11.

Table 1
The time and resource requirements of activities in Fig. 1

Activity	A_1	A_2	A_3	A_4	A_5	A_6	A_7	A_8	A_9
Time	2	3	3	3	2	1	5	3	2
Resource	3	4	2	1	2	3	1	4	2

3. Differential Search Algorithm

Differential Search Algorithm is a new and effective evolutionary algorithm which was inspired by migration of superorganisms utilizing the concept of Brownian-like random-walk motion (Civicioglu, 2012). In DS algorithm, it is assumed that random solution of the population is matching to the artificial-superorganism migration to optimum solution of the problem. During the movement, artificial-superorganism examines whether some randomly selected area are desirable and it is a fertile area. If the selected area during the migration is temporarily a good choice to stop over, the members of the artificial-superorganism decide to stay at this area. They repeatedly continue their movement from this area to find more suitable areas. Pseudo-code of DS algorithm is given in Appendix A. In this Pseudo-code, N represents number of organisms in the superorganism and D is the size of the respective problem. Artificial-organisms are shown by $X_i = x_{i,j} (i \in \{1,2,3, \dots N\} \;\; and \;\; j \in \{1,2,3, \dots D\})$ and artificial-superorganism is shown by $Superorganism_g = [X_i]$ $(g \in \{1,2,3, \dots, maxgen\})$. An artificial-superorganism contains N artificial-organisms as its elements. Artificial-organisms have members where each member is shown by $x_{i,j} (j \in \{1,2,3, \dots, D\})$ and each $x_{i,j}$ is initially defined by:

$$x_{i,j} = rand \times (up_j - low_j) + low_j. \tag{4}$$

Randomly selected individuals of the artificial-organisms move in the direction of the targets of $donor = [X_{random\_shuffling(i)}]$ in order to find stopover sites. $random\_shuffling$ is a function that randomly changes the order of the numbers of the members in the set of $i = \{1,2,3, \dots, N\}$. Considering the DS Pseudo-code, scale variable (which is defined as $Scale = randg\,[2 \times rand] \times (rand2 - rand3)$) controls the size of the change occurred in the positions of the members of the artificial-organisms. $Rand_g$ that is a gamma random number generator and standard random function together make scale value. This is made that an artificial-superorganism direction is changed in the habitat. Members of the artificial-organisms take part in stopover site search by a random search. Lines 8-29 from the DS Pseudo-code show this matter. Stopover site is generated using the following equation:

$$StopoverSite = Superorganism + Scale \times (donor - Superorganism) \tag{5}$$

In Lines 31-33, the stopover site is controlled to remain in determined search space range. In DS algorithm, the stopover site found by the search process is evaluated and if the new discovered stopover site of an artificial-organism has better quality than the current sources of that artificial-organism, it goes to that stopover site. While the artificial-organisms of a superorganism change site, that respective superorganism continues its movement in the direction of the global optimum.

4. DS algorithm for RCPSP

This section presents the proposed DS algorithm for solving RCPSP problem in details. Considering the performance of the DS in optimization problems, it seems that the method could be effective to solve RCPSP problems. The flowchart of the proposed method is presented in Fig. 2.

According to Fig. 2, DS for the RCPSP has four phases: 1) Input, 2) Initialization, 3) Update, 4) Terminate. The Input phase receives triples (N, D, G) as input and transfer them to the initialization phase. The second phase places the organisms on the search space randomly using Eq. (4). In *Initialization* and *Update* phases, each artificial organism represents a schedule for the RCPSP problem. If the problem has N activities, the artificialorganism will migrate in the search space with N dimensions. A position is represented as a priority list $\vec{P}(p_1, p_2, \dots, p_n)$ where each element of this list fixedly represents an activity and its corresponding value shows the priority of that activity (Akbari et al., 2012). Based on this representation, the position vector $X_i = (x_{i1}, x_{i2}, \dots, x_{in})$ of artificial organism i represents the priority values of N activities. The lower and upper bounds of each priority value are set at 0 and 1, respectively. The priority values smaller than 0 are set to 0 and the priority values larger than 1 are set to 1 (Ziarati et al., 2011).

The third phase receives the initial solutions as input and iteratively updates these solution until the termination condition is met. The solutions are based on the behavior of superorganism and its members. The member of an artificial-superorganism controls its migration by considering the amount of fertility of targets. Members of superorganism migrate to more fertile locations by considering the fertile intensity that associated with that location.

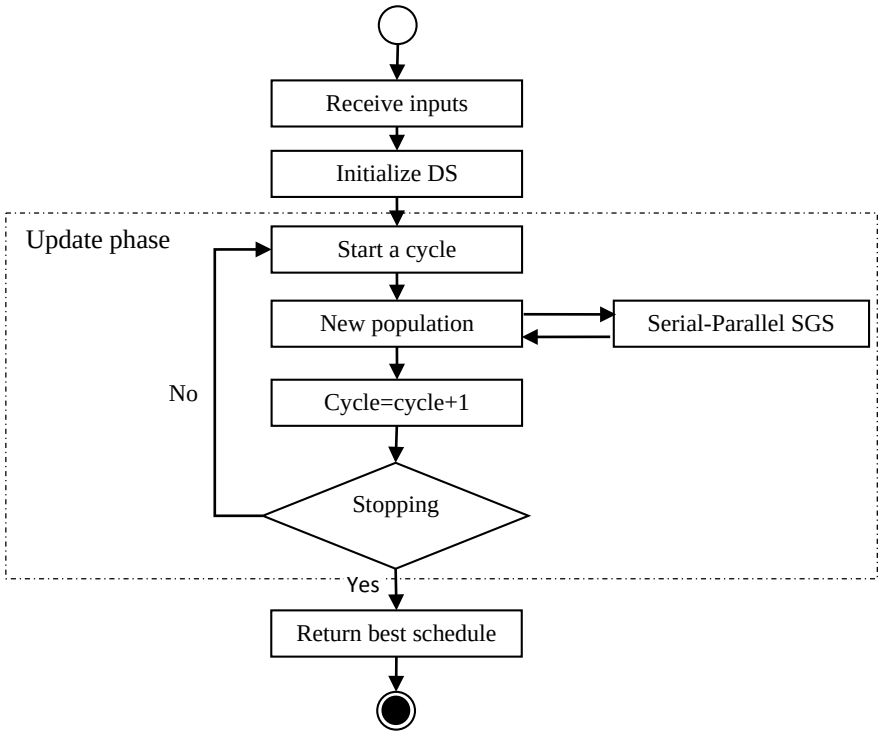

Fig. 2. Flowchart of the proposed method

Naturally, the fertile locations represent better solutions. The method needs to evaluate the fitness of the solutions proposed by each of the artificial organism. To evaluate the fitness, amount of fertility of area should be determined. For this purpose, the method needs to generate the schedule from the priority list. The stopover site is considered as the makespan of the schedule which is presented by an artificialorganism. Hence, we need to use a schedule generation scheme (SGS) such as Serial-SGS or Parallel-SGS (Kolisch, 1996). In this work both of the SGS methods are considered. For this purpose, a random number in range of [0,1] is generated in order to select the SGS which is used by the algorithm

to construct the schedule. If the random number is larger than 0.5, the Parallel-SGS is selected else the Serial-SGS is selected. The Serial-Parallel SGS provides the ability for DS algorithm to use advantage of both scheme. The Serial-Parallel SGS module in Fig. 2 performs this task.

Next, the positions of the artificial organisms are updated using the migration pattern of the standard DS and a permutation method. The proposed method uses the permutation-based representation which was first introduced by Hong et al. (2005). After calculating the new position of the artificial organism i, the permutation process is used by the algorithm to permute this new solution. Each of the artificial organisms i represents feasible activity lists and their corresponding priority lists. The priority list of the artificial organisms i is updated and a new priority list will be obtained. After that a random list is generated (Ziarati et al., 2011). Then, each priority value of the artificial organism i is compared with its corresponding value in the random vector. If the random value is smaller than its corresponding priority value, then the corresponding activity will be swapped with its neighbor. After swapping the priority values and their corresponding activities, the obtained activity list is examined against the constraints according to Eq. (1) and Eq. (2). If the constraints satisfaction is violated, the infeasible activity list is resolved to the feasible one. After moving to the new position, the fertile intensity of the new artificial organism is evaluated. If the better fertile site is obtained, the position of the artificial organism is updated. This comparison is accomplished for each artificial organisms. After the comparison, the artificial organisms are ranked based on the amount of fertility of their stopover site and the best solution found is updated. The fourth phase terminates the search process after the maximum number of generation G is reached. It should be noted that the input parmeter G is set in such a way that the maximum number of schedules defined in the experiment are produced. After termination, the best schedule (i.e. the schedule with minimum makespan) which is found by DS algorithm is returned as output. It seems that the proposed method is capable to solve RCPSP problem efficiently. The specifications of sample project described in Fig. 1 and Table 1 is given to the DS algorithm. The proposed algorithm has the ability to solve this problem successfully. The obtained result by DS algorithm is given in Fig. 3. As it can be seen, the minimum makespan for this project is given by DS methods under resource and precedence constraints in Table 1 and Fig. 1.

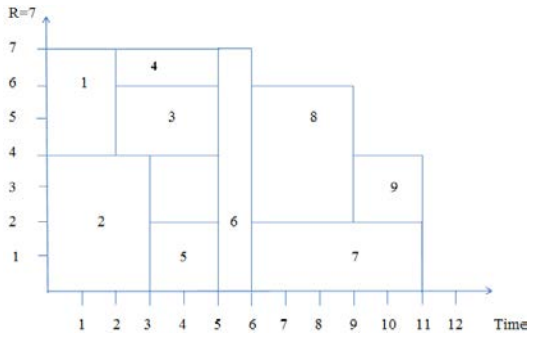

Fig. 3. The scheduling obtained by DS algorithm for sample given in Fig. 1

5. Experiments

This section presents the performance of the proposed DS method on the single mode test cases in PSPLIB library (Project Scheduling Problem Library) in terms of success rate and the deviation from the optimal solution. The performance of the proposed method is considered under the following configurations. Two types of experiments are conducted: the first experiments study the effect of SGS and the second experiments show the comparative performance.

The population size (N) is set at 30. Different number of schedules are used here as the termination condition of the DS method. The numbers of schedules are set at 1000, 5000, and 50000 to evaluate the effect of SGS and 1000, and 5000 for the comparison study. The success rates of the DS method are obtained over the j30, j60, and j120 case studies from the PSPLIB. Similarly, for the comparison study, j30, j60, and j120 are used. For both of the experiments, the average results of 10 independent runs are reported.

5.1. The effect of SGS

The social behaviors in DS method provide the ability to use it for scheduling problem successfully. The effect of SGS on the performance of the proposed method in terms of success rate and average deviation is considered here.

Table 2
The effect of SGS on the average deviation of DS

	SGS	1000	5000	50000
J30	parallel	0.32	0.22	0.07
	serial	0.29	0.19	0.04
	both	0.20	0.11	0.02
J60	parallel	12.41	11.85	11.35
	serial	12.35	11.65	11.20
	both	11.75	11.10	10.90
J120	parallel	36.50	35.51	34.86
	serial	35.84	35.13	34.55
	both	35.14	33.67	33.14

Table 2 shows the effects of SGS on the average deviation. It can be seen that the best results are obtained when both schemes are used. Also, using serial-SGS provides the ability for the DS method to obtain better performance compared with parallel-SGS. The results show that the type of SGS has positive effect on the performance of DS method. The success rate shows the number of instances in a case study which are successfully solved by DS method. Table 3 shows the success rates of the proposed algorithm. From the results, it can be seen that the DS method obtains better performance when it uses both serial and parallel SGS. Based on this experiment, both of the SGS are used by DS method in comparison study given in the next section.

Table 3
The effect of SGS on the success rate of DS

	SGS	1000	5000	50000
J30	parallel	85.41%	90.62%	96.87%
	serial	87.92%	91.45%	97.50%
	both	88.96%	93.96%	98.33%
J60	parallel	72.30%	73.54%	77.70%
	serial	72.50%	74.16%	78.13%
	both	72.91%	74.58%	78.75%
J120	parallel	29.84%	31.33%	33.33%
	serial	30.00%	31.66%	34.16%
	both	30.33%	32.66%	35.00%

5.2. Comparative study

The success rates and average deviations obtained in the previous experiment showed that DS method had efficiency in solving RCPSP problem. In this section, the best results obtained by the proposed algorithm are compared with a set of state of art methods. DJ in the proposed method DS-DJ is related to double justification which is used for generating better schedules (Akbari et al., 2012). In DJ, the right and left justifications are used to adjust the start time of each activity in scheduling. Better performance may be obtained by DJ used in DS algorithm. Due to the large number of methods presented in recent years for RCPSP, only a subset of these methods are selected for comparison. However, we tried to select different types of meta-heuristics for this comparison.

The average deviations and success rates of the proposed method in comparison with the other state-of-art methods over the j30 case study after 1000 and 5000 schedule generation are given in Table 4. The results of the other methods reported in Tables 4, 5 and 6 are directly obtained from their corresponding papers. In some papers, average deviation and success rate of their presented methods for one of the experiments (i.e. 1000 and 5000 schedule generation) have not been reported. In such

cases "-" sign is used to show the unavailability of the results. The results show that the proposed method obtains the first rank after 1000 schedule generations and obtains the fourth rank after 5000 schedule generation. DS method has the best success rates after 1000 and 5000 schedules. It seems that the proposed method has competitive performance and produce comparable results in comparison with the other methods.

The performance of the proposed DS method over the j60 case study after 1000 and 5000 schedule generation is given in Table 5. Due to the larger number of activities in this case study, the proposed method and the other ones have more difficulty to solve. The success rates decrease compared to J30 case study as well as the algorithms have more deviation. The second rank is obtained by the proposed DS method after 1000 schedule generation. However, the proposed method surpass the other methods after 5000 schedule generation. The investigated method has more difficulty in solving j120 case study and their performance decrease drastically. Table 6 shows the results obtained over the j120 case study. The results show that the DS algorithm had good and competitive performance compared with other methods. The first and the second rank is obtained by the proposed method after 1000 and 5000 schedule generation, respectively. The best success rates after 1000 and 5000 schedules are obtained by DS algorithm.

Table 4
Average deviation and success rates over J30

J30		Average Deviation:opt		Success Rate	
Approach	Reference	1000	5000	1000	5000
DS-DJ	**The propsoed study**	**0.20**	**0.11**	**88.96**	**93.96**
MJPSO	Fahmy et al., 2014	0.22	0.05	-	-
PSO-HH	Koulinas et al., 2014	0.26	0.04	-	-
FA-DJ	Sanaei et al., 2013	-	0.12	-	93.75
P-PSO	Nasiri, 2012	0.30	0.10		
PABC	Qiong & Seo, 2013	0.34	0.17	86.60	91.74
ABC	Akbari et al., 2012	0.35	0.12	-	
BA-DJ	Ziarati et al., 2011	0.42	0.19	83.96	91.05
BSO-DJ	Ziarati et al., 2011	0.45	0.22	83.55	90.21
ABC-DJ	Ziarati et al., 2011	0.47	0.28	82.50	90.00
Improved PSO	Jia & Seo, 2013	0.49	-	-	-
BA	Ziarati et al., 2011	0.63	0.33	78.54	86.25
BSO	Ziarati et al., 2011	0.65	0.36	77.30	85.63
ABC	Ziarati et al., 2011	0.98	0.57	72.71	83.84
GA	Alcaraz & Maroto, 2001	0.33	0.12	-	–
GA-DJ	Valls et al., 2005	0.34	0.20	-	–
GA	Hartmann, 2002	0.38	0.22	-	–
SA	Bouleimen & Lecocq, 2003	0.38	0.23	-	-
TS	Nonobe & Ibaraki, 2002	0.46	0.16	-	-
GA	Hartmann, 1998	0.54	0.25	81.50	-
PSO	Chen et al., 2010	0.54	-	-	
AS	Schirmer, 2000	0.65	0.44	-	
PSO	Zhang, 2005	0.69	0.42	-	-
GA	Hartmann, 1998	1.38	1.22	70.60	-
ACO	Chen et al., 2010	1.57	-	-	

In general, based on the results obtained for PSPIB scheduling problems, the proposed DS method provides well in solving single-mode RCPSP problems.

Table 5
Average deviation and success rates over J60

J60		Average Deviation:lb		Success Rate	
Approach	Reference	1000	5000	1000	5000
PSO-HH	Koulinas et al., 2014	11.74	11.13		
DS-DJ	**This study**	**11.75**	**11.10**	**72.91**	**74.58**
MJPSO	Fahmy et al., 2014	11.86	11.19	-	-
FA-DJ	Sanaei et al., 2013	-	11.20	-	74.37
P-PSO	Nasiri, 2012	12.02	11.33	-	-
Improved PSO	Jia & Seo, 2013	12.12	-	-	-
PABC	Qiong & Seo, 2013	12.35	11.96	72.50	74.03
BA-DJ	Akbari et al., 2012	12.55	12.04	72.30	73.96
BSO-DJ	Ziarati et al., 2011	12.58	12.29	72.08	73.34
ABC-DJ	Ziarati et al., 2011	12.61	12.24	71.67	73.34
ABC	Ziarati et al., 2011	12.75	11.48	-	-
BA	Ziarati et al., 2011	13.35	12.83	66.25	68.34
BSO	Ziarati et al., 2011	13.67	12.70	64.38	70.63
ABC	Ziarati et al., 2011	14.57	13.12	61.88	67.09
GA	Hartmann, 2002	12.21	11.70	-	-
B & B	Dorndorf et al., 2000	12.50	-	-	76.20
GA	Hartmann, 1998	12.68	11.89	-	-
GA	Hartmann, 1998	12.74	12.74	-	-
GA-DJ	Valls et al., 2005	12.21	11.27	-	-
GA	Alcaraz & Maroto, 2001	12.57	11.86	-	-
SA	Bouleimen & Lecocq, 2003	12.75	11.90	-	-
AS	Schirmer, 2000	12.94	12.58	-	-
TS	Nonobe & Ibaraki, 2002	12.97	12.18	-	-

Table 5
Average deviation and success rates over J120

J120		Average Deviation:lb		Success Rate	
Approach	Reference	1000	5000	1000	5000
DS-DJ	**The propsoed study**	**35.15**	**33.67**	**30.33**	**32.66**
PSO-HH	Koulinas et al., 2014	35.20	32.59	-	-
MJPSO	Fahmy et al., 2014	35.60	33.78	-	-
FA-DJ	Sanaei et al., 2013	-	34.07	-	32.16
PABC	Qiong & Seo, 2013	36.84	35.79	29.50	31.20
P-PSO	Nasiri, 2012	36.77	35.16	-	-
ABC	Akbari et al., 2012	36.29	34.18	-	-
Improved PSO	Jia & Seo, 2013	37.22	-	-	-
BA-DJ	Ziarati et al., 2011	37.72	36.76	29.84	31.17
BSO-DJ	Ziarati et al., 2011	37.84	36.51	29.17	30.84
ABC-DJ	Ziarati et al., 2011	37.85	36.82	29.34	30.34
ALNP	Xiao et al., 2014	37.49	36.74	-	-
BA	Ziarati et al., 2011	40.38	38.12	17.84	20.84
BSO	Ziarati et al., 2011	41.18	37.86	17.00	22.50
ABC	Ziarati et al., 2011	43.24	39.87	15.34	18.17
p-ACO	Herbots et al., 2004	-	36.01	-	19.00
ACO	Herbots et al., 2004	-	37.85	-	29.33
ACO	Merkle et al., 2002	-	38.02	-	26.50
s-ACO	Merkle et al., 2002	-	39.82	-	26.70
GA-DJ	Valls et al., 2005	35.39	33.24	-	-
GA	Hartmann, 2002	37.19	35.39	-	-
GA	Alcaraz & Maroto, 2001	39.36	36.57	-	-
GA	Hartmann, 1998	39.37	36.74	-	-
AS	Schirmer, 2000	39.85	38.70	-	-
GA	Hartmann, 1998	39.93	38.49	-	-
TS	Nonobe & Ibaraki, 2002	40.86	37.88	-	-
SA	Bouleimen & Lecocq, 2003	42.81	37.68	-	-

6. Conclusion and future works

In recent years, different types of meta-heuristics have been presented by researchers to cope with complex problems. Differential Search Algorithm which is one of the most recently introduced methods of this type which was originally proposed for solving real-valued numerical optimization problems. In this work a new method based on DS for solving resource-constrained project scheduling problem was proposed. The proposed method gives a set of initial schedules and tries to improve them using the migration behavior of the superorganism. For this purpose, the DS method is adapted to obtain an arrangement of the activities which results the best schedule. The comparative study of the well-known PSPLIB benchmarks showed that the proposed DS algorithm had the ability to produce competitive results compared to the other metaheuristic methods. In addition, the proposed DS method has high efficiency in solving RCPSP problems. It seems that this method has high potential to incorporate different types of heuristics, local search, and constraint handling approaches, etc. in order to improve its performances. Also, hybridization of the DS method with other heuristic or meta-heuristics may provide a way to improve their efficiency.

References

Agarwal, A., Colak, S., & Erenguc, S. (2011). A neurogenetic approach for the resource-constrained project scheduling problem. *Computers & Operations Research*, *38*(1), 44-50.

Akbari, R., Zeighami, V., & Ziarati, K. (2011). Artificial bee colony for resource constrained project scheduling problem. *International Journal of Industrial Engineering Computations*, *2*(1), 45-60.

Akbari, R., Zeighami, V., & Akbari, I. (2012). An ABC-Genetic method to solve resource constrained project scheduling problem. *Artificial Intelligence Research*, *1*(2), p185.

Alcaraz, J., & Maroto, C. (2001). A robust genetic algorithm for resource allocation in project scheduling. *Annals of Operations Research*, *102*(1-4), 83-109.

Baar, T., Brucker, P., & Knust, S. (1999). *Tabu search algorithms and lower bounds for the resource-constrained project scheduling problem* (pp. 1-18). Springer US.

Bouleimen K. and Lecocq H.. (2003). 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.

Chen, W., Shi, Y. J., Teng, H. F., Lan, X. P., & Hu, L. C. (2010). An efficient hybrid algorithm for resource-constrained project scheduling. *Information Sciences*, *180*(6), 1031-1039.

Chen, R. M., Wu, C. L., Wang, C. M., & Lo, S. T. (2010). Using novel particle swarm optimization scheme to solve resource-constrained scheduling problem in PSPLIB. *Expert systems with applications*, *37*(3), 1899-1910.

Civicioglu, P. (2012). Transforming geocentric cartesian coordinates to geodetic coordinates by using differential search algorithm. *Computers & Geosciences*,*46*, 229-247.

Coelho, J., & Vanhoucke, M. (2011). Multi-mode resource-constrained project scheduling using RCPSP and SAT solvers. *European Journal of Operational Research*, *213*(1), 73-82.

Dorndorf, U., Pesch, E., & Phan-Huy, T. (2000). A time-oriented branch-and-bound algorithm for resource-constrained project scheduling with generalised precedence constraints. *Management Science*, *46*(10), 1365-1384.

Fahmy, A., Hassan, T. M., & Bassioni, H. (2014). Improving RCPSP solutions quality with Stacking Justification–Application with particle swarm optimization. *Expert Systems with Applications*, *41*(13), 5870-5881.

Hartmann, S., & Kolisch, R. (2000). Experimental evaluation of state-of-the-art heuristics for the resource-constrained project scheduling problem. *European Journal of Operational Research*, *127*(2), 394-407.

Hartmann, S. (1998). A competitive genetic algorithm for resource-constrained project scheduling. *Naval Research Logistics (NRL)*, *45*(7), 733-750.

Hartmann, S. (2002). A self-adapting genetic algorithm for project scheduling under resource constraints. *Naval Research Logistics (NRL)*, *49*(5), 433-448.

Herbots, J., Herroelen, W., & Leus, R. (2004). Experimental investigation of the applicability of ant colony optimization algorithms for project scheduling. *DTEW Research Report 0459*, 1-25.

Jarboui, B., Damak, N., Siarry, P., & Rebai, A. (2008). A combinatorial particle swarm optimization for solving multi-mode resource-constrained project scheduling problems. *Applied Mathematics and Computation*, *195*(1), 299-308.

Jia, Q., & Seo, Y. (2013). Solving resource-constrained project scheduling problems: conceptual validation of FLP formulation and efficient permutation-based ABC computation. *Computers & Operations Research*, *40*(8), 2037-2050.

Jia, Q., & Seo, Y. (2013). An improved particle swarm optimization for the resource-constrained project scheduling problem. *The International Journal of Advanced Manufacturing Technology*, *67*(9-12), 2627-2638.

Koulinas, G., Kotsikas, L., & Anagnostopoulos, K. (2014). A particle swarm optimization based hyper-heuristic algorithm for the classic resource constrained project scheduling problem. *Information Sciences*, *277*, 680-693.

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

Kolisch, R., & Hartmann, S. (2006). Experimental investigation of heuristics for resource-constrained project scheduling: An update. *European journal of operational research*, *174*(1), 23-37.

Kolisch, R., & Hartmann, S. (1999). *Heuristic algorithms for the resource-constrained project scheduling problem: Classification and computational analysis* (pp. 147-178). Springer US.

Kolisch, R., & Padman, R. (2001). An integrated survey of deterministic project scheduling. *Omega*, *29*(3), 249-272.

Kumar, N. (2014). Study on meta-heuristics for resource constrained project scheduling problem. *International Journal of Engineering, Management & Sciences*, 1(2), 14-24.

Luo, X., Wang, D., Tang, J., & Tu, Y. (2006, June). An improved pso algorithm for resource-constrained project scheduling problem. In *Intelligent Control and Automation, 2006. WCICA 2006. The Sixth World Congress on* (Vol. 1, pp. 3514-3518). IEEE.

Mendes, J. J. D. M., Gonçalves, J. F., & Resende, M. G. (2009). A random key based genetic algorithm for the resource constrained project scheduling problem. *Computers & Operations Research*, *36*(1), 92-109.

Merkle, D., Middendorf, M., & Schmeck, H. (2002). Ant colony optimization for resource-constrained project scheduling. *Evolutionary Computation, IEEE Transactions on*, *6*(4), 333-346.

Nasiri, M. M. (2013). A pseudo particle swarm optimization for the RCPSP. *The International Journal of Advanced Manufacturing Technology*, *65*(5-8), 909-918.

Neumann, K., & Zimmermann, J. (1999). Methods for resource-constrained project scheduling with regular and nonregular objective functions and schedule-dependent time windows. In *Project Scheduling* (pp. 261-287). Springer US.

Nonobe, K., & Ibaraki, T. (2002). Formulation and tabu search algorithm for the resource constrained project scheduling problem. In *Essays and surveys in metaheuristics* (pp. 557-588). Springer US.

Orji, I. M., & Wei, S. (2013, April). Project scheduling under resource constraints: a recent survey. In *International Journal of Engineering Research and Technology* (Vol. 2, No. 2 (February-2013)). ESRSA Publications.

Ranjbar, M., Kianfar, F., & Shadrokh, S. (2008). Solving the resource availability cost problem in project scheduling by path relinking and genetic algorithm. *Applied Mathematics and Computation*, *196*(2), 879-888.

Sanaei, P., Akbari, R., Zeighami, V., & Shams, S. (2013, January). Using firefly algorithm to solve resource constrained project scheduling problem. In*Proceedings of Seventh International Conference on Bio-Inspired Computing: Theories and Applications (BIC-TA 2012)* (pp. 417-428). Springer India.

Schirmer, A. (2000). Case-based reasoning and improved adaptive search for project scheduling. *Naval Research Logistics (NRL), 47*(3), 201-222.

Shou, Y., Li, Y., & Lai, C. (2015). Hybrid particle swarm optimization for preemptive resource-constrained project scheduling. *Neurocomputing*, *148*, 122-128.

Tseng, L. Y., & Chen, S. C. (2006). A hybrid metaheuristic for the resource-constrained project scheduling problem. *European Journal of Operational Research*, *175*(2), 707-721.

Valls, V., Ballestín, F., & Quintanilla, S. (2005). Justification and RCPSP: A technique that pays. *European Journal of Operational Research*, *165*(2), 375-386.

Valls, V., Ballestin, F., & Quintanilla, S. (2008). A hybrid genetic algorithm for the resource-constrained project scheduling problem. *European Journal of Operational Research*, *185*(2), 495-508.

Xiao, L., Tian, J., & Liu, Z. (2014, June). An Activity-List based Nested Partitions algorithm for Resource-Constrained Project Scheduling. In *Intelligent Control and Automation (WCICA), 2014 11th World Congress on* (pp. 3450-3454). IEEE.

Yang, B., Geunes, J., & O'brien, W. J. (2001). Resource-constrained project scheduling: Past work and new directions. *Department of Industrial and Systems Engineering, University of Florida, Tech. Rep*.

Zamani, R. (2013). A competitive magnet-based genetic algorithm for solving the resource-constrained project scheduling problem. *European Journal of Operational Research*, *229*(2), 552-559.

Zhang, H., Li, X., Li, H., & Huang, F. (2005). Particle swarm optimization-based schemes for resource-constrained project scheduling. *Automation in Construction*, *14*(3), 393-404.

Zhang, C., Sun, J., Zhu, X., & Yang, Q. (2008). An improved particle swarm optimization algorithm for flowshop scheduling problem. *Information Processing Letters*, *108*(4), 204-209.

Zhang, H., Li, X., Li, H., & Huang, F. (2005). Particle swarm optimization-based schemes for resource-constrained project scheduling. *Automation in Construction*, *14*(3), 393-404.

Zeighamia, V., Akbarib, R., & Ziaratic, K. (2013). Development of a method based on particle swarm optimization to solve resource constrained project scheduling problem. *Scientia Irancia*, *20*(6), 2123-2137.

Zheng, H. Y., Wang, L., & Wang, S. Y. (2014, July). A co-evolutionary teaching-learning-based optimization algorithm for stochastic RCPSP. In*Evolutionary Computation (CEC), 2014 IEEE Congress on* (pp. 587-594). IEEE.

Ziarati, K., Akbari, R., & Zeighami, V. (2011). On the performance of bee algorithms for resource-constrained project scheduling problem. *Applied Soft Computing*, *11*(4), 3720-3733.

Appendix A

Pseudocode of the proposed DS method for RCPSP (Civicioglu, 2012)

Input $(\boldsymbol{N}, \boldsymbol{D}, \boldsymbol{G})$**:**

N: The size of the population, where $i=\{1,2,3,...,N\}$
D: The dimension of the problem.
G:Number of maximum generation.

Initialization:

1: $Superorganism=initialize()$, where $Superorganism=[ArtificialOrganism_i]$
2: $y_i=Evaluate(ArtificialOrganism_i)$

Update:

3: **for** $cycle= 1:G$ **do**
4: $donor= Superorganism_{Random\_shuffling(i)}$
5: $Scale= randg\ [2.rand].(rand_2 - rand_3)$
6: $StopoverSite= Superorganism + Scale\ .\ (donor - Superorganism\ )$
7: $p_1= 0.3\ .rand_4$ and $p_2= 0.3.rand_5$
8: **if** $rand_6 < rand_7$ **then**
9 : **if** $rand_8 < p_1$ **then**
10: $r= rand(N,D)$
11: **for** $Counter1= 1: N$ **do**
12: $r(Counter1,:) = r(Counter1,:) < rand_9$
13: **endfor**
14: **else**
15: r = ones(N,D)
16: **for** $Counter2= 1: N$ **do**
17: $r(Counter2, randi(D)\ ) = r(Counter2, randi(D)) < rand_{10}$
18: **endfor**
19: **endif**
20: **else**
21: $r = ones(N,D)$
22: **for** $Counter3= 1:N$ **do**
23: $d = randi(D, 1, \lfloor P_2 \cdot rand \cdot D \rceil)$
24: **for** $Counter4= 1: size(d)$ **do**
25: $r(\ Counter3, d(Counter4))=0$
26: **endfor**
27: **endfor**
28: **endif**
29: $individuals_{I,J} \leftarrow r_{I,J} > 0 | I \in i\ , J \in [1, D]$
30: $StopoverSite(individuals_{I,J}) := Superorganism(individuals_{I,J})$
31: **if** $(StopoverSite_{i,j} < low_{i,j}$ or $StopoverSite_{i,j} > up_{i,j})$ **then**
32: $StopoverSite_{i,j} := rand.\ (up_j - low_j) + low_j$
33: **endif**
34: $y_{StopoverSite;i} = evaluate(StopoverSite_i)$
35 **if** $\left(y_{StopoverSite;i} < y_{Superorganism;i}\right)$ **then**
36 $y_{Superorganism,i} := y_{StopoverSite;i}$
37 **else**
38 $y_{StopoverSite;i} := y_{Superorganism;i}$

550

endif
if $\left(y_{StopoverSite;i} < y_{Superorganism;i}\right)$ **then**
$ArtificialOrganism_i \coloneqq StopoverSite_i$
else
$ArtificialOrganism_i \coloneqq ArtificialOrganism_i$
endif
endfor

Terminate
Return *best stopover site*