Using Firefly Algorithm to Solve Resource Constrained Project Scheduling Problem

Pejman Sanaei[1], Reza Akbari[2], Vahid Zeighami[3] and Sheida Shams[4]

[1,3]Department of Mathematics, Shiraz University, Shiraz, Iran

[2]Department of Computer Engineering and Information Technology, Shiraz University of Technology, Shiraz, Iran

[4]Department of Management, Shiraz University, Shiraz, Iran

{pejman.sanaei@gmail.com[1], akbari@sutech.ac.ir[2], vahid.zeighami@gmail.com[3], sheida.shams@gmail.com[4]}

Abstract. The Firefly Algorithm (FA) is among the most recently introduced meta-heuristics. This work aims at study the application of FA algorithm to solve the Resource Constrained Project Scheduling Problem (RCPSP). The algorithm starts by generating a set of random schedules. After that, the initial schedules are improved iteratively using the flying approach proposed by the FA. By termination of algorithm, the best schedule found by the method is returned as the final result. The results of the state-of-art algorithms are used in this work in order to evaluate the performance of the proposed method. The comparison study shows the efficiency of the proposed method in solving RCPSP. The proposed method has competitive performance compared to the other RCPSP solvers.

Keywords: Firefly Algorithm, Resource Constrained Project Scheduling problem.

1 Introduction

The RCPSP is among the difficult problems to solve. The difficulty of RCPSP arises from the limitation of resources and precedence constraints. In a single-mode RCPSP, a set of activities A and K renewable resource type R are given as a project. Each activity A_j has duration d_j and requires units of resource R_k during each period of its duration. The first and last activities are known as dummies with zero duration and consume no resource type. In such situation, the objective of a RCPSP solver is to minimize the makespan of a schedule S by considering the constraints.

Due to NP-hardness of the RCPSP problems, the exact methods have difficulty in solving RCPSP problems of large size. Hence other alternative are required. As an alternative, the heuristic methods [1]-[4] can be used. The heuristics methods

have the ability to produce near-optimal solution even for the large-size RCPSP problems. Meta-heuristic methods can be used as another way to solve the RCPSP problems. These methods can be divided in two classes. The first class contains methods such as tabu search [5] and simulated annealing [6] that maintain only one solution during each iteration. The second class contains methods such as genetic algorithm [7]-[10], Ant Colony Optimization [11], particle swarm optimization [12]-[14], and bee algorithms[15] -[16] that maintain a set of solution during each iteration. The efficiency of the mentioned methods have been combined by each others and a set of hybrid methods have been proposed. ANGEL [17], ACOSS [18], Neurogenetic [19], and Hybrid-GA [20] are among hybrid methods presented in literature. It seems that better performance may be obtained by hybrid methods.

Firefly Algorithm is among recently introduces meta-heuristic method. The algorithm is inspired from the intelligent behaviors of fireflies. The FA is a meta-heuristic method which was first introduced by Yang to solve the numerical problems [21]. Due to the efficiency of the FA on other problems, it seems that the FA may be useful to solve RCPSP problems. It seems that few methods based on FA have been proposed over the RCPSP problem. This idea has been considered by the authors and a new method based on the standard FA is proposed to solve RCPSP problems. The proposed method starts by initializing a population of fireflies. After initialization, the arrangement of the activities in each solution is updated iteration by iteration based on the behavior of fireflies until the termination condition is met. After termination, the best result found by the method is returned as the final result.

The remaining of this paper is organized as follows: Section 2 presents the details of the proposed method for solving RCPSP problems. The experimental results are given in Section 3. Finally, Section 4 concludes this work.

2 Firefly Algorithm for RCPSP

This section presents the proposed FA algorithm for solving RCPSP problem in details. The FA has been designed based on the flashing lights of a swarm of fireflies. The flashing light is vital for a firefly. Using the flashing light, a firefly can find mate, protect himself/herself from predators, and attracts the potential preys. To inspire an algorithm from the behaviors of fireflies it seems that the flashing light intensity should be used. A firefly controls its movement by considering the flashing light intensity of targets. A swarm of fireflies attracts to the brighter and more interesting locations by the flashing light intensity that associated with that location. In the implementation of the FA, it is assumed that the brighter locations represent better solutions. Hence, the algorithm tries to help the fireflies to find such locations in the search space. In general, the brightness of the flashing light is considered as the objective function. The attraction of a firefly depends on the flashing brightness of the target location. The brightness decreases as the distance between a firefly and the target location increases.

In our implementation each firefly represents a schedule for the problem at the hand. If the problem has N activities, the fireflies will fly 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. Based on this representation, the position vector $\overrightarrow{x_i} = (x_{i1}, x_{i2}, \dots, x_{in})$ of Firefly *i* represents the priority values of *N* activities. The lower bound and upper bound of each priority value are set at 0 and 1 respectively. The priority values smaller than 0 are set at 0 and the priority values larger than 1 are set at 1.

Input:

n: number of fireflies in the swarm
G_{max}: maximum number of generation
CS: RCPSP problem

Initialization:

Objective function f(x), $x = (x1, \dots, xd)$
Generate initial swarm of fireflies xi $(i = 1, 2, \dots, n)$
Light intensity Ii at xi is determined by $f(xi)$
Define light absorption coefficient ã

Update:

While $t < G_{max}$
  For i = 1 : n all n fireflies
    Evaluate the flashing light intensity of firefly i
  End for
  For i = 1 : n all n fireflies
    For j = 1 : i all n fireflies
      If ($Ij > Ii$), Move firefly i towards j in d-dimension
        Attractiveness varies with distance r_{ij} via $e^{-\gamma r_{ij}}$ Evaluate new solutions and update light intensity
      End if
    End for j
  End for i
  Rank the fireflies and find the current best
End while

Termination:

Select the best-so-far solution and return it as the final result

---

Fig. 1. Pseudopod of the FA for RCPSP problem.

The pseudo code of FA applied to solve the RCPSP is shown in Fig. 1. The FA for the RCPSP has three main phases. The first phase starts the algorithm. This phase which is called initialization, places the fireflies on the search space randomly. The second phase called "update", receives the initial solutions as input and iteratively update these solution until the termination condition is met. the third phase is called "termination" ends the algorithm and returns the best solution found by the swarm. The details of these phases are given in the next sub-sections.

2.1 Initialization

The "Initialization" is the first of the proposed FA algorithm. After receiving the input parameters such as the size of population (n), the maximum number of generations (G_{max}), and the case study, the FA method starts with the n Fireflies being placed randomly in the solution space. After initialization, the method needs to evaluate the fitness of the solutions proposed by each of the fireflies in the swarm. To evaluate the fitness, the flashing light intensity of the solution should be determined. For this purpose, the method needs to generate the schedule from the priority list. Also, in this phase, the light absorption coefficient γ is determined.

2.2 Position Updating

After initializing the swarm of fireflies, the FA method tries to improve the initial solutions based on the behavior of fireflies. The update phase of the proposed method has two main steps: in the first step, the flashing light intensities of the fireflies are calculated and in the second steps the positions of the fireflies are updated using the movement pattern of the standard FA and a permutation method.

Firefly evaluation*)* To update the positions, we need to compute the fitness of the solutions which are found by the fireflies. For this purpose, the flashing light intensity of the fireflies is computed. For the RCPSP problem, the flashing light intensity is considered as the makspan of the schedule which is presented by a firefly. Hence, we need to use a schedule generation scheme (SGS) such as Serial-SGS or Parallel-SGS [22]. In this work both of the SGS methods are considered. For this purpose, a random number r_{SGS} in range of [0,1] is generated in order to select the SGS which is used by the algorithm to construct the schedule. as shown in equation (5), if r_{SGS} is larger than a predefined threshold th, the Parallel-SGS is select else the Serial-SGS is selected. In our implementation, the threshold th is set at 0.5.

$$\mathrm{SGS} = \begin{cases} \mathrm{Serial-SGS} & \text{if } r_{SGS} \leq \mathrm{th} \\ \mathrm{Parallel-SGS} & \text{if } r_{SGS} > \mathrm{th} \end{cases} \quad (1)$$

Using the Serial-Parallel SGS provide the ability for the algorithm to use the potentiality of both scheme. Moreover, the forward-backward improvement (FBI) is used in the proposed FA algorithm. Using serial (parallel) SGS, and FBI, the makespan of each solution presented by a firefly is calculated as its flashing light intensity. The flashing light intensity of the firefly j is calculated using following equation:

$$I_j = fit(\vec{x}_j) = \frac{1}{Make\_Span_j} \quad (2)$$

where $Make\_Span_j$ is the value of the makespan proposed by the Firefly j.

Update position) After computing the flashing light intensity of all the fireflies in the swarm. The algorithm proceeds to update the position of fireflies. For this purpose uses a loop of pairwise comparison of flashing light intensity. In the comparison of two fireflies, each of the fireflies which has a lower light intensity is attracted toward the other firefly with the higher light intensity. The new position of the firefly is computed using the following equation:

$$x_i = x_i + \beta_0 \times e^{-\gamma r_{ij}^2} \times (x_i - x_j) + \alpha(rand - 0.5) \quad (3)$$

where x_i and x_j represent the locations of the firefly fireflies i and j respectively. $rand$ is a random number which is drawn from the rang of $[0,1]$. α is a randomization parameter which can be selected from range $[0,1]$. The third part of the equation ($\alpha(rand - 0.5)$) controls the random movement of the firefly. The second part controls the movement of the firefly i toward firefly j. β_0 is the initial attractiveness at $r = 0$, and γ is an absorption coefficient which controls the decrease of the light intensity. The value of β_0 can be determined based on the problem. Usually it is set at 1. The attractiveness function of a firefly can be computed as:

$$\beta_{(r)} = \beta_0 \times \exp(-\gamma r^m), with\ m \geq 1 \quad (4)$$

In equation (3), r_{ij} represents the distance between two fireflies i and j at x_i and x_j locations in the search space respectively. It can be defined as a Cartesian distance:

$$r_{ij} = \|x_i - x_j\| = \sqrt{\sum_{k=1}^{d}(x_{i,k} - x_{j,k})^2} \quad (5)$$

where $x_{i,k}$ is the k^{th}component of the spatial coordinate x_i of the firefly i and d represents the number of dimensions of the search space.

The proposed method uses the permutation-based representation which was first introduced in [23]. After calculating the new position of the firefly i using equation (3), the permutation process is used by the algorithm to permute this new solution. To provide the update step in a more clear way, we use an example which is given in Fig. 2. As can be seen in this figure, each of the fireflies i and j represent feasible activity lists and their corresponding priority lists. The priority list of the firefly i is updated using equation (3) and a new priority list will be obtained. After that a random list is generated. Then, each priority value of the firefly 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 and if the constraints satisfaction is violated, the infeasible activity list is resolve to the feasible one.

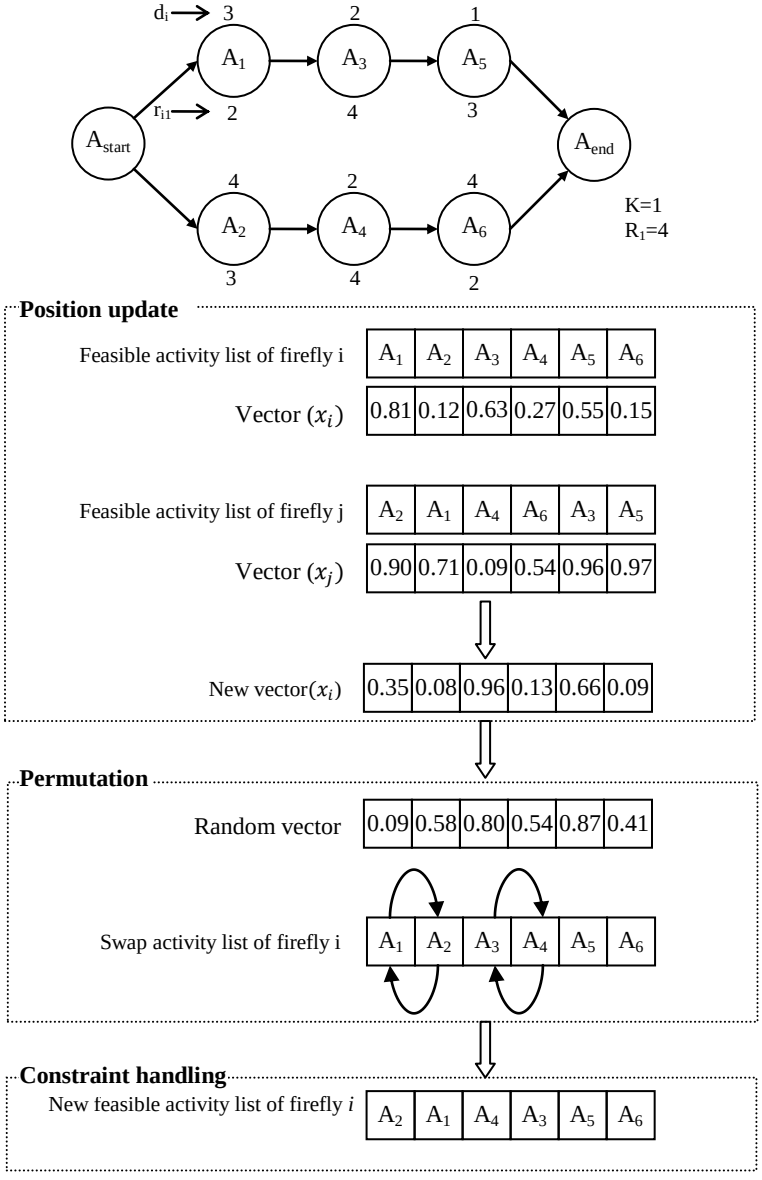

Fig. 2. Updating of the firefly-represented permutation and a precedence graph.

After moving to the new position, the flashing light intensity of the new firefly is evaluated. If the better flashing light intensity obtained, the position of the firefly is updated.

This comparison is done for each couple of the fireflies. After the comparison, the fireflies are ranked based on their flashing light intensity and the best solution found by the swarm is updated.

2.3 Termination

In our implementation the proposed FA algorithm is terminated after predefined number of schedule generations. As can be seen from Fig. 1, G_{max} shows the termination condition. G_{max} is computed as $maximum\ number\ of\ schedules$ After termination, the FA algorithm returns the schedule with minimum makespan as the ultimate output.

3 Experiments

This section presents the performance of the proposed FA method on the Single Mode Data Sets cases in PSPLIB library [24] in terms of success rate and the deviation from the optimal solution. These experiments are conducted under the following setting: The population size (n) is set at 30. Different number of schedules are used here as the termination condition of the FA method. The numbers of schedules are set at 1000, 5000, and 50000 to evaluate the success rate and 5000, and 50000 for the comparative study. The success rates of the FA method are obtained over the j30, j60, j90, and j120 case studies from the PSPLIB. For the comparative study, the j30, j60, and j120 are used. For both of the experiments, the average results of 10 independent runs are reported. The main parameters including β_0, $\gamma$and α set to 1, 0.8 and 0.2 respectively.

3.1 Success Rate

The success rate shows the number of instances in a case study which are successfully solved by the FA method. For an instance, the FA method is called successful if it can find the optimal schedule which has the minimum makespan. Table 1 shows the success rates of the proposed algorithm on PSPLIB case studies. From the results, it can be seen that the proposed method has good performance over the PSPLIB case studies. As the number of schedules increases the success rate of the FA method increases too. The complexity of the case studies increases as the number of activities in its instances increase. Hence, it is obvious that the proposed method has more difficulty in solving case studies with more activities. For example the success rate of the proposed method drops from 98.33% over the j30 case study to 35.00% over the j120 case study.

Table 1. Average success rate for PSPLIB case studies.

	#schedules		
Case study	1000	5000	50000
j30	89.58%	93.75%	98.33%
J60	72.91%	74.37%	78.75%
J90	72.30%	75.00%	77.08%
J120	20.33%	32.16%	35.00%

3.2 Comparative Study

The success rates showed the FA method has efficiency in solving RCPSP problem. Although, success rate shows the ability of a method to reach the optimal results, the performance of a method can be considered in term of average deviation from the optimal solution.

Table 2. Average deviation for j30 case study.

Algorithm	***Reference***	***5000***	***50,000***
ACOSS	Chen et al.[18]	0.06	0.01
Scatter Search—FBI	Debels et al. [25]	0.11	0.01
GA—hybrid, FBI	Valls et al. [20]	0.06	0.02
FA – FBI	***This Study***	***0.12***	***0.02***
GA – forw.–backw., FBI	Alcaraz et al. [26]	0.06	0.03
JPSO	Ruey-Maw Chen [27]	0.14	0.04
Sampling—LFT, FBI	Tormos and Lova [28]	0.13	0.05
TS—activity list	Nonobe and Ibaraki [29]	0.16	0.05
Sampling – LFT, FBI	Tormos and Lova[28]	0.16	0.07
GA—self-adapting	Hartmann [8]	0.22	0.08
GA—activity list	Hartmann [30]	0.25	0.08
ANGEL	Tseng and Chen [17]	0.09	-
Sampling – LFT, FBI	Tormos and Lova [28]	0.17	0.09
Neurogenetic	Agrawal et al. [19]	0.10	-
Sampling – random, FBI	Valls et al. [20]	0.28	0.11
GA – late join	Coelho and Tavares [31]	0.33	0.16
SA – activity list	Bouleimen and Lecocq[6]	0.23	-
PSO-Priorities of activities	Zhang [23]	0.42	-
Sampling—adaptative	Schirmer and Riesenberg[32]	0.44	-
TS—schedule scheme	Baar et al.[5]	0.44	-
Sampling—adaptative	Kolisch and Drexl [33]	0.53	-
GA – random key	Hartmann[30]	0.56	0.23
Sampling – LFT	Kolisch[22]	0.53	0.27
Sampling – global	Coelho and Tavares [31]	0.54	0.28
PSO-Permutation	Zhang[23]	0.61	-
GA—priority rule	Hartmann [30]	1.12	0.88
Sampling—WCS	Kolisch [22]	1.28	-
Sampling—LFT	Kolisch [22]	1.29	1.13
Sampling—random	Kolisch [34]	1.48	1.22
GA—problem space	Leon and Ramamoorthy [35]	1.59	-

Average deviation has been considered as an important measure to compare the performance of the investigated methods against each others. This measure shows the ability of an algorithm in converging towards the optimal solution. Here, the

average deviation is used to investigate the performance of the proposed method in comparison with the other state-of-art methods. The performance of the state-of-art methods reported here are directly obtained from their original papers.The average deviations of the proposed method in comparison with the other state-of-art methods over the j30 case study are given in Table 2. The results show that the proposed method obtains the forth rank. Only three hybrid methods called ACOSS ,Scatter Search-FBI and GA-hybrid, FBI have better performance than the proposed FA method.

Table 3. Average deviation for j60 case study.

Algorithm	***Reference***	***5000***	***50,000***
ACOSS	Chen et al.[18]	10.98	10.67
Scatter search—FBI	Debels et al.[25]	11.10	10.71
GA—hybrid, FBI	Valls et al.[20]	11.10	10.73
FA - FBI	***This Study***	***11.20***	***10.80***
GA – forw.–backw.FBI	Alcaraz et al.[26]	11.19	10.84
JPSO	Ruey-Maw Chen[27]	11.43	11.00
GA—self-adapting	Hartmann[8]	11.70	11.21
GA—activity list	Hartmann[30]	11.89	11.23
ANGEL	Tseng and Chen [17]	11.27	-
Neurogenetic	Agrawal et al. [19]	11.29	-
Sampling – LFT, FBI	Tormos and Lova [28]	11.62	11.36
Sampling – LFT, FBI	Tormos and Lova [28]	11.82	11.47
GA – forw.–backw	Alcaraz and Maroto[36]	11.86	-
Sampling – LFT, FBI	Tormos and Lova [28]	11.87	11.54
SA – activity list	Bouleimen and Lecocq[6]	11.90	-
TS – activity list	Nonobe and Ibaraki [29]	12.18	11.58
Sampling–random,FBI	Valls et al.[20]	12.35	11.94
Sampling – adaptive	Schirmer [32]	12.58	-
GA – late join	Coelho and Tavares[31]	12.63	11.94
GA – random key	Hartmann[30]	13.32	12.25
GA – priority rule	Hartmann[30]	12.74	12.26
Sampling – adaptive	Kolisch and Drexl[33]	13.06	-
Sampling – WCS	Kolisch[33]	13.21	-
Sampling – global	Coelho and Tavares[31]	13.31	12.83
Sampling – LFT	Kolisch [22]	13.23	12.85
TS – schedule scheme	Baar et al.[5]	13.48	-
GA – problem space	Leon and Ramamoorthy[35]	13.49	-
Sampling – LFT	Kolisch[22]	13.53	12.97
Sampling – random	Kolisch[34]	14.30	13.66
Sampling – random	Kolisch [34]	15.17	14.22

The performance of the proposed FA method over the j60 case study after 5000 and 50000 schedule generation is given in Table 3. Similar to the j30 case study, the forth rank is obtained by the proposed FA method. The hybrid methods ACOSS Scatter search-FBI and GA-hybrid, FBI obtained the first and second ranks respectively. Table 4 shows the results obtained over the j120 case study. The proposed FA algorithm is among the four best algorithms. The results show that the FA algorithm has competitive performance compared to the other

methods such as ACOSS, GA-hybrid-FBI, GA -forw-backw.FBI, and Scatter Search -FBI.

Table 4. Average deviation forj120case study.

Algorithm	***Reference***	***5000***	***50,000***
ACOSS	Chen et al.[18]	32.48	30.56
GA—hybrid, FBI	Valls et al.[20]	32.54	31.24
GA – forw.–backw.FBI	Alcaraz et al.[26]	33.91	31.49
Scatter Search – FBI	Debels et al.[25]	33.10	31.57
FA - FBI	***This Study***	***34.07***	***32.85***
JPSO	Ruey-Maw Chen [27]	33.88	32.89
GA – self-adapting	Hartmann[8]	35.39	33.21
Sampling – LFT, FBI	Tormos and Lova [28]	34.41	33.71
Neurogenetic	Agrawal et al.[19]	34.15	-
ANGEL	Tseng and Chen[17]	34.49	-
Ant system	Merkle et al.[11]	35.43	-
GA—activity list	Hartmann[8]	36.74	34.03
Sampling – LFT, FBI	Tormos and Lova [28]	35.56	34.77
Sampling – LFT, FBI	Tormos and Lova [28]	35.81	35.01
GA – forw.–backw	Alcaraz and Maroto[36]	36.57	-
TS – activity list	Nonobe and Ibaraki [29]	37.88	35.85
GA – late join	Coelho and Tavares[31]	38.41	36.44
Sampling–random,FBI	Valls et al.[20]	37.47	36.46
SA – activity list	Bouleimen and Lecocq[6]	37.68	-
GA – priority rule	Hartmann[8]	38.49	36.51
Sampling – adaptive	Schirmer [32]	38.70	-
Sampling – LFT	Kolisch [33]	38.75	37.74
Sampling – WCS	Kolisch[22]	38.77	-
GA – random key	Hartmann [8]	42.25	38.83
Sampling – adaptive	Kolisch and Drexl[33]	40.45	-
Sampling – global	Coelho and Tavares [31]	40.46	39.41
GA – problem space	Leon and Ramamoorthy [35]	40.69	-
Sampling – LFT	Kolisch[22]	41.84	40.63
Sampling – random	Kolisch[34]	43.05	41.44
Sampling – random	Kolisch[34]	47.61	45.60

5 Conclusions

A new method for solving resource-constrained project scheduling problem was proposed in this work. This method is based on the Firefly Algorithm which is originally proposed by Yang for optimizing numerical functions. The proposed method gives a set of initial schedules and tries to improve them using the search behavior of the fireflies. For this purpose, the original FA 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 FA algorithm has the ability to produce competitive results compared to the other state-of-art methods. Although, the proposed FA method has efficiency

in solving RCPSP problems, more efficiency may be obtained by incorporating local search methods. Although, hybridization of this method with the other meta-heuristics seems interesting.

References

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

[2] R. Kolisch, S. Hartmann, Experimental investigation of heuristics for resource-constrained project scheduling: an update, European Journal of Operational Research 174 (2006) 23–37.

[3] R. Kolisch, S. Hartmann, Heuristic algorithms for solving the resource-constrained project scheduling problem: classification and computational analysis, in: J. Weglarz (Ed.), Project Scheduling: Recent Models, algorithms and Applications, Kluwer Academic Publishers, Berlin, 1999, pp. 147–178.

[4] R. Kolisch, R. Padman, An integrated survey of deterministic project scheduling, Omega 29 (2001) 249–272.

[5] Baar T, Brucker P, Knust S. Tabu-search algorithms and lower bounds for the resource-constrained project scheduling problem. In: Voss S, Martello S, Osman I, Roucairol C, editors. Meta-heurisitics: advances and trends in local search paradigms for optimization. Dordrecht: Kluwer; 1998. p. 1–8.

[6] K. Bouleimen and H. Lecocq. "A new efficient simulated annealing algorithm for the resource– constrained project scheduling problem and its multiple modes version". European Journal of Operational Research, 149, (2003) 268–281.

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

[8] S. Hartmann, A self-adapting genetic algorithm for project scheduling under resource constraints, Naval Research Logistics 49 (2002) 433–448.

[9] J.J.M. Mendes, J.F. Goncalves, M.G.C. Resende, A random key based genetic algorithm for the resource constrained project scheduling problem, Computers & Operations Research 36 (2009) 92–109.

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

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

[12] B. Jarboui, N. Damak, P. Siarry, A. Rebai, A combinatorial particle swarm optimization for solving multi-mode resource-constrained project schedulingproblems, Applied Mathematics and Computation 195 (2008) 299–308.

[13] Luo, X., Wang, D., Tang, J., & Tu, Y. (2006). An improved PSO algorithm for resource constrained project scheduling problem, intelligent control and automation,2006. In The sixth world congress on WCICA 2006 (Vol. 1, pp. 3514–3518).

[14] Zhang, C., Sun, J., Zhu, X., & Yang, Q. (2008). An improved particle swarmoptimization algorithm for flowshop scheduling problem. InformationProcessing Letters, 108(4), 204–209.

[15] 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, Vol.11, No. 4, pp. 3720-3733, 2011.

[16] Akbari R., Zeighami V., and Ziarati K., Artificial Bee colony for resource constrained project scheduling problem, International Journal of Industrial Engineering Computations, DOI: 10.5267/j.ijiec.2010.04.004.
[17] Tseng, L.Y., & Chen, S. C.(2006). A hybrid metaheuristic for the resource-constrained project scheduling problem, European Journal of Operational Research, 175, 707–721.
[18] 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, 1031–1039.
[19] Agarwal, A., Colak, S., &Erenguc, S.(2010). A Neurogenetic approach for the resource-constrained project scheduling problem. Computers & Operations Research doi:10.1016/j.cor.2010.01.007.
[20] Valls V., Ballestın F., & Quintanilla, S.(2008). A hybrid genetic algorithm for the resource-constrained project scheduling problem. European Journal of Operational Research, 185, 495–508.
[21] X.-S. Yang, "Firefly Algorithms for multimodal optimization", Lecture Notes in Computer Science, vol. 5792, pp. 169-178, 2009.
[22] Kolisch R. Serial and parallel resource-constrained project scheduling methods revisite: theory and computation. European Journal of Operational Research 1996;90:320–33.
[23] Hong Zhang, Xiaodong Li, Heng Li, Fulai Huang, "Particle swarm optimization-based schemes for resource-constrained project scheduling", in *Journal of Automation in Construction*, Vol. 14 (2005) 393– 404.
[24] Project Scheduling Problem Library – PSPLIB: <http://www.129.187.106.231/psplib/>.
[25] 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.
[26] J. Alcaraz, C. Maroto, and R. Ruiz. "Improving the performance of genetic algorithms for the RCPS problem". Proceedings of the Ninth International Workshop on Project Management and Scheduling, (2004) pages 40–43, Nancy,.
[27] Ruey-Maw Chen "Particle swarm optimization with justification and designed mechanisms for resource-constrained project scheduling problem" Expert Systems with Applications, Volume 38, Issue 6, June 2011, Pages 7102-7111
[28] Tormos P, Lova A. Integrating heuristics for resource constrained project scheduling: one step forward. Technical Report, Department ofStatistics and Operations Research, Universidad Politecnica de Valencia; 2003.
[29] Nonobe K, Ibaraki T. Formulation and tabu search algorithm for the resource constrained project scheduling problem. In: Ribeiro CC, Hansen P, editors. Essays and surveys in metaheuristics. Dordrecht: Kluwer Academic Publishers; 2002. p. 557–588.
[30] Hartmann S. A competitive genetic algorithm for resource-constrained project scheduling. Naval Research Logistics 1998;45:279–302.
[31] J. Coelho and L. Tavares. "Comparative analysis of meta–heuricstics for the resource constrained project scheduling problem". Technical report, Department of Civil Engineering, Instituto Superior Tecnico, Portugal, 2003.
[32] A. Schirmer. "Case–based reasoning and improved adaptive search for project scheduling". Naval Research Logistics, 47 (2000) 201–222.
[33] R. Kolisch and A. Drexl. "Adaptive search for solving hard project scheduling problems". Naval Research Logistics, 43 (1996) 23–40.
[34] Kolisch R. Project scheduling under resource constraints: efficient heuristics for several problem classes. Wurzburg: Physica-Verlag; 1995.
[35] Leon VJ, Ramamoorthy B. Strength and adaptability of problem-space based neighborhoods for resource constrained scheduling. Operations Research Spektrum 1995;17:173–82.
[36] J. Alcaraz and C. Maroto. "A robust genetic algorithm for resource allocation in project scheduling". Annals of Operations Research, 102 (2001) 83–109.