A Novel Learning Algorithm Based on a Multi-Agent Structure for Solving Multi-Mode Resource-Constrained Project Scheduling Problem

Omid Mirzaei and Mohammad-R. Akbarzadeh-T.

Abstract The resource-constrained project scheduling problem (RCPSP) includes activities which have to be scheduled due to precedence and resource restrictions such that an objective is satisfied. There are several variants of this problem currently, and also different objectives are considered with regards to the specific applications. This paper tries to introduce a new multi-agent learning algorithm (MALA) for solving the multi-mode resource-constrained project scheduling problem (MMRCPSP), in which the activities of the project can be performed in multiple execution modes. This work aims to minimize the total project duration which is referred to its makespan. The experimental results show that our method is a new one for this specific problem and can outperform other algorithms in different areas.

Keywords Multi-agent systems · Machine learning · Multi-mode resource-constrained project scheduling problem · MMRCPSP

O. Mirzaei (✉)
Department of Computer Engineering, Mashhad Branch,
Islamic Azad University, Mashhad, Iran
e-mail: O.Mirzaei@mshdiau.ac.ir

M.-R. Akbarzadeh-T.
Center of Excellence on Soft Computing and Intelligent Information Processing, Ferdowsi University of Mashhad, Mashhad, Iran
e-mail: akbarzadeh@ieee.org

1 Introduction

Project scheduling has become a popular subject in recent years both in science and practice. It has drawn increasing attentions in many real life applications and industries such as project management and crew scheduling, fleet management and also machine assignment, construction engineering, automobile industry, software development and the last but not least, make-to-order firms in which the capacities have been reduced in order to cope with lean management concepts. Furthermore, the resource-constrained project scheduling problem (RCPSP) is proven to be an NP-hard optimization problem [1].

Recently, in the literature of project scheduling, the RCPSP problem has been considered as a standard problem in the field. Within the classical type of this problem, the activities of the project have to be scheduled in such a way that an objective is satisfied. The most common objective in classical mode is makespan minimization. Thus, one has to consider not only technological precedence constraints but also the limitations of the renewable resources required to accomplish the activities. The precedence relations between the activities are demonstrated using a graph representation which is called Activity-On-Node (AON) Diagram. There exist several extensions on this single problem. The classical type of this problem can be extended to multi-mode resource-constrained project scheduling problem (MMRCPSP) in which each task can be done in many different execution modes [2]. Each mode stands for another way of mixing different levels of resource requirements with an affiliated duration.

According to the categorization scheme suggested by Slowinski [3], renewable, nonrenewable, and doubly constrained resources [4] are the three classifications of resources necessary for the execution of a project. Renewable re-sources (such as hour, day, week and month) are available on a period-by-period basis while nonrenewable resources (such as money, energy and raw material) are limited on a total project basis. Doubly constrained are those resources which are limited on both total project basis and per-period basis.

A broad variety of methods have been proposed for the multi-mode resource-constrained project scheduling problem so far. These algorithms are able to be classified into three main groups: exact algorithms [4], heuristic algorithms, and agent-based algorithms [5]. The heuristic approaches themselves can be divided into two strategies: classical meta-heuristic (for instance genetic algorithms [6], tabu search [2], simulated annealing [2, 7], ant colony [8] and bee colony [9]), and nonstandard meta-heuristic (for instance local search-oriented solutions [10] and population-based algorithms [11]).

The schedule generation scheme which is used by different algorithms can be performed either in serial [12] or parallel [13]. Each method makes use of one of these schemes to construct schedules and obtain the overall project makespan. With regards to empirical experiments which have been done so far, the parallel schedule generation scheme do not always lead to optimal solutions [5]. Hence, we have chosen the serial schedule generation scheme in this work.

This study concerns an agent based solution for the multi-mode resource-constrained project scheduling problem. The activities of the project are considered as agents who will make a multi-agent system considering the AON network. Each agent has two devices for making its decisions: (1) learning automaton and (2) heuristic-based stochastic local dispatcher. LA shows significant theoretical convergence properties in both single and multi automata environments. Considering this matter, a motivation for using them is that they are excellent tools for multi-agent reinforcement learning solutions [14]. The local dispatcher is considered in each agent to add a degree of randomness in selecting the order of activities and also the execution mode of each activity. A global dispatcher is also considered the same as [5] to avoid the algorithm from getting stuck into local optima.

In comparison with our main Ref. [5], there are two main differences. Firstly, the schedules are constructed locally here which increases the flexibility of the solution and allows the project to be dynamic. Secondly, local dispatchers have been added to each agent as their second tool for making their corresponding decisions. These dispatchers are heuristic-based stochastic, which will lead to a degree of randomness in the agents' decisions. Randomness has been incorporated for the cases in which the agents cannot make a decision through their learning devices (rationality) and the heuristic value leads to faster convergence of the algorithm.

The organization of this paper runs as follow: Section 2 provides the description of the problem and also its model. Section 3 contains our proposed multi-agent based learning algorithm for the multi-mode resource constrained project scheduling problem. Section 4 includes some experiments and comparative outcomes. Lastly, Sect. 5 states some conclusions and discusses future work.

2 Problem Description and Model

The standard MMRCPSP can be formulated as follows. We consider a project which consists of J activities (jobs) labeled $j = 1, 2, \ldots, J$. The processing time (or duration) of an activity j is denoted as p_j. When an activity begins, it may not be interrupted and its mode may not be changed. As mentioned before, there are precedence constraints between some of the activities due to technological requirements. These precedence constraints are given by sets of immediate predecessors P_j, demonstrating that if all of activity j‘s predecessors are not completed, it may not begin before. These can be all represented using an activity-on-node network (diagram) which is assumed to be acyclic. There are two more activities which are called source and sink which display the start and end of the project. These activities are "dummy" with duration of zero and no resource needs. Figure 1 demonstrates a sample AON network in the MMRCPSP problem.

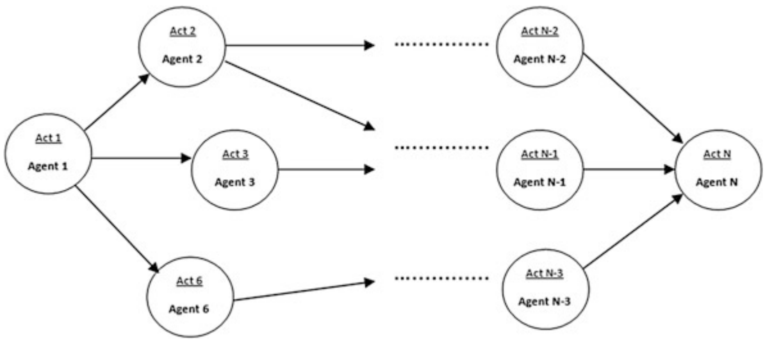

Fig. 1 A sample AON network for MMRCPSP problem

Each activity needs a definite number of resources to be done with the exclusion of dummy source activity and also the sink activities. R represents the set of renewable resources. For each renewable resource $r \in R$, the per period availability is invariable and given by K_r^ρ. N indicates a set of nonrenewable resources. The overall availability of each nonrenewable resource $n \in N$ for the whole project is shown by K_r^v [4].

Any of these activities can be done in a set of different modes of execution. A combination of various resources and/or levels of resource requirements with a specific duration are referred to as a mode [4]. Activity j may be carried out in M_j modes marked as $m = 1, 2, \ldots, M_j$ and its duration which is done in mode m is given by d_{jm}. Moreover, whenever activity j is carried out in mode m, it uses K_{jmr}^ρ units of renewable resource r each time it is in process, where we presume $w.\log K_{jmr}^\rho \leq K_r^\rho$ for each renewable resource $r \in R$ [5]. Otherwise, activity j could not be carried out in mode m. Furthermore, it uses K_{jmr}^v units of nonrenewable resources $n \in N$.

There may be different objectives for this kind of problem. These include objectives based on renewable and nonrenewable resources and also robustness based objectives. Here, the objective of our work is to lessen the makespan of the project. Our work is based on this assumption that the parameters are nonnegative and integer-valued.

3 Multi-Agent Learning Algorithm

The first decision making unit which is used by agents is learning automaton. This unit is an adaptive one, which is located in an accidental environment and learns the best possible action based on past actions and environmental feedback. Properly, it can be illustrated by a quadruple $\{A, f, d, U\}$, in which A stands for a set of actions which are possibly taken, the random reinforcement signal given by the environment is shown by f, d represents the probability distribution over all actions and the learning scheme which is used to update d is demonstrated by U. At each instant k, $k = 1, 2, 3, \ldots$, the automaton selects an action considering its action probability vector d(k) [5].

$$d(k) = [d_1(k), d_2(k), \ldots, d_r(k)], \quad \sum d_i(k) = 1 \tag{1}$$

The environment receives the selected action as input and its response (feedback) to these actions serves as input to the automaton.

Many automaton update schemes with different properties have been studied and planned up to now. Among these, linear reward-penalty, linear reward-inaction and linear reward-ε-penalty are some important instances of linear update schemes. The purpose of all these schemes is fundamentally to boost the opportunity to choose an action when it brings about a success and decrease it when it results in a failure. The general algorithm is given by below equations:

If a_m is the action taken at time t:

$$d_m(t+1) = d_m(t) + \alpha_{reward} f(t)(1 - d_m(t)) - \alpha_{penalty}(1 - f(t))d_m(t) \tag{2}$$

If $a_j \neq a_m$:

$$d_j(t+1) = d_j(t) - \alpha_{reward} f(t) d_j(t) + \alpha_{penalty}(1 - f(t))[(q-1)^{-1} - d_j(t)) \tag{3}$$

The parameters that illustrate the reward and penalty are α_{reward} and α_{penalty}. If the algorithm indicates $\alpha_{\text{reward}} = \alpha_{\text{penalty}}$, it is a sign of linear reward-penalty (L_{R-P}); a linear reward-inaction (L_{R-I}) is involved when $\alpha_{\text{penalty}} = 0$, and α_{penalty} is called linear reward-ε-penalty (L_{R-P}) when it is small compared to α_{reward}. $f(t) \in [0, 1]$ is the reward given by the environment as feedback for the action taken at instant t, and the number of actions is shown by q.

The ε-optimality property of (L_{R-I}) method in all stationary environments has made us to apply it for learning the activity order and the best execution modes of activities [5]. LRO is the learning rate (reward parameter) which is used for learning the activity order and LRM is the one that is used for learning the execution mode of each activity.

In the proposed algorithm, a local frame is developed and enlarged along the execution of the project. This frame is used for making local schedules of the activities which have been added to it before. This procedure is repeated for some iterations (itrs) to complete the learning of agents. If an agent is observed for the first time and it is added to scheduled activities list, all of its successors are added to the frame if their predecessors presented in the frame before. To present an initial outline of our proposed algorithm, we have demonstrated it in Fig. 2.

According to Fig. 3, each agent makes use of two learning automata to choose its own execution mode and also the order of visiting its successor activities. The algorithm is willing to choose this order with regards to the agent's decision. Agents make this decision through consulting their learning devices (i.e. learning automata). These learning automata pick out an alternative based on their action

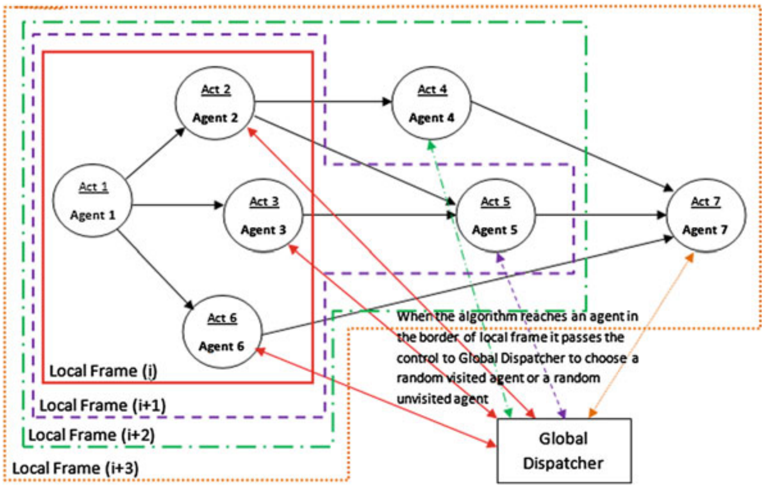
Act 1
Agent 1
Act 2
Agent 2
Act 3
Agent 3
Act 4
Agent 4
Act 5
Agent 5
Act 6
Agent 6
Act 7
Agent 7
Local Frame (i)
Local Frame (i+1)
Local Frame (i+2)
Local Frame (i+3)
When the algorithm reaches an agent in the border of local frame it passes the control to Global Dispatcher to choose a random visited agent or a random unvisited agent
Global Dispatcher

Fig. 2 The proposed multi-agent algorithm with its local frames

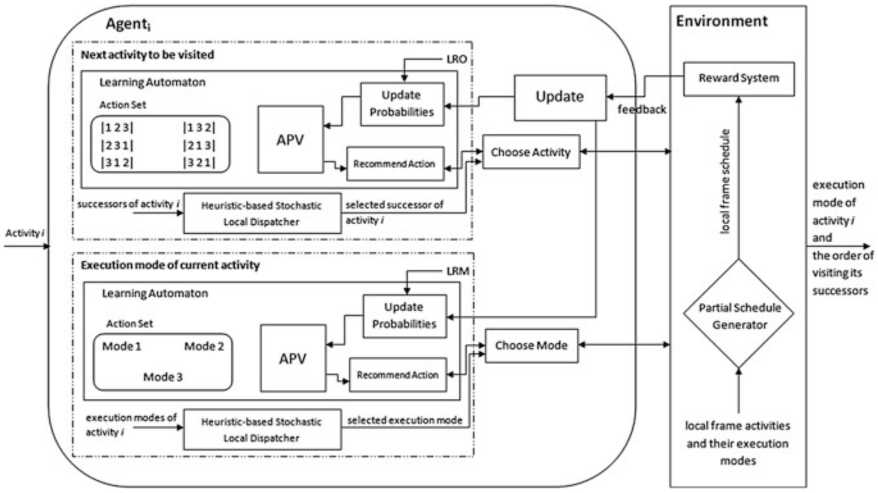

Fig. 3 The inner structure of an agent

probability vector. The length of this vector depends on operational choices. For instance, if the agent should decide between two choices for visiting its successor activities, the length of this vector will be two. The reward system makes use of the information from partially made schedules, and it will update the action probability vector of learning automata in the corresponding agent consistent with the reinforcement (reward) rules in Eqs. (2) and (3). If the makespan of the constructed schedule at instant t was:

- Better : $\mathrm{f(t) = 1}$
- Equal :

$$\mathrm{f(t) = f_{eq}(f_{eq} \in [0, 1])} \tag{4}$$

- Worse : $\mathrm{f(t) = 0}$

The speed of learning can be easily changed by modifying both $\mathrm{f_{eq}}$ and the learning rates LRO and LRM. A higher value for f_{eq} can make the learning faster, particularly for this kind of problems where efforts to develop novel schedules only rarely bring about improvements in quality [5]. The settings of the two learning rates are reliant on each other. A suitable arrangement of these values will be vital for achieving a good general performance.

Hence, the agents use a degree of rationality in making their decisions through learning automata and then, they get closer to optimal decisions by receiving feedbacks from the environment. Another decision making unit has been inserted in each agent named "heuristic-based stochastic local dispatcher" to add a degree of randomness to decisions made by agents. If the decisions made by rationality are wrong to a certain probability, the randomness prevents the agent from getting stuck into local optima. So, if this unit takes the control to some probability (γ), it will make the decisions according to a heuristic value. Moreover, there is also a global dispatcher in our proposed algorithm which is used when we reach an agent in the border of local frame and its successors are not presented within it. In this case, the control is given to this global dispatcher. It has a specific probability (δ) of selecting a random suitable agent from the list of previously observed agents, or else it selects a random suitable unvisited agent. This probability can be either static or dynamic through the process. The inner structure of each agent is illustrated in Fig. 3.

Once again, it should be mentioned that when an agent wants to make a decision through using its local dispatcher for choosing the execution mode of the current activity or the next activity which wants to be scheduled, it uses a heuristic value. This value is inversely proportional to the activities duration and their resource requirements. It means that when the local dispatcher wants to choose the next activity for scheduling, it makes this decision with regards to activities duration and their resource requirements, i.e. the activities which have the shortest duration and the lowest resource requirements in their best execution modes have a bigger probability to be selected. The aforementioned heuristic value can be well understood referring to the below equation:

$$\begin{gathered}\mathrm{h_{i,j} = \frac{1}{D_{i,j} \times RR_{i,j}}} \\ i = 1, 2, \ldots, \mathit{Successors},\; j = 1, 2, \ldots, \mathit{Execution\ Modes\ of\ Activity}\ i\end{gathered} \tag{5}$$

Table 1 Average computation times

Algorithm	SGS	c15 10_1	c21 31_2	j10 61_7	j12 31_10	j18 30_9	j20 24_5	j30 49_7	m1 57_1	m5 21_4	n3 54_10	r4 39_3
CPSO [16]	Serial	7.57	16.38	6.15	8.44	17.94	16.34	31.09	4.52	13.33	18.24	17.92
HGA [17]	Both	10.23	18.86	9.05	10.43	19.49	18.46	34.07	7.32	16.31	21.03	20.12
MARLA [5]	Serial	1.29	2.09	0.80	1.21	2.22	2.43	5.15	1.65	1.79	3.01	1.99
MALA [proposed]	Serial	1.59	3.01	0.86	1.26	3.10	4.11	16.56	2.38	2.60	4.15	3.56

Where D and RR refer to the activities duration and their resource requirements respectively.

4 Experimental Results

Throughout this section, the performance of the multi-agent learning algorithm will be evaluated for the multi-mode resource-constrained project scheduling problem (MMRCPSP). Our algorithm has been implemented in MATLAB 2009 on a system with an Intel(R) Core(TM)2 Duo processor, 4 GB RAM and a 64-bit windows 7 operating system. The famous sets of instances produced by the project generator ProGen for the MMRCPSP have been used to test the proficiency of the proposed algorithm [15] and the infeasible instances have been expelled from the experiments.

We have chosen two heuristic algorithms and one multi-agent algorithm to show the performance of our proposed method. The first algorithm is a kind of nonstandard meta-heuristic solution and is based on particle swarm which had been led to good results according to its authors' claims [16]. The second one is based on genetic algorithm and is a classical meta-heuristic approach [17]. Finally, the multi-agent algorithm is another agent-based solution which was claimed to be very effective for this kind of problem [5]. The following empirically obtained parameters have been used to get the whole results: $\text{LRO} = 0.4$, $\text{LRM} = 0.4$, $f_{eq} = 0.01$, $\gamma = 0$, $\delta = 0.5$ and $\text{itrs} = 3$.

The initial criterion which has been used for comparing the efficiency of the algorithms from the view point of computational burden is the average computation times. These values have been measured accurately for 10 times of execution on numerous datasets and are gathered in Table 1. The instances of datasets have been specified below each one.

It is clear from Table 1 that our suggested algorithm has much lower computational times in all of the datasets than two heuristic algorithms. Moreover, we have separately applied all of the algorithms on j10 dataset to see the distribution of computation times on it. In Table 2, we present the results of this simulation. It

Table 2 Distribution of the computation times (%)—Dataset = j10

Algorithm	SGS	[0.1,0.3)	[0.3,0.5)	[0.5,0.7)	[0.7,1)	[1,3)	[3,5)	[5,7)	[7,9)	[9, 12]
CPSO [16]	Serial	0	0	0	0	0	52.38	23.80	19.04	4.78
HGA [17]	Both	0	0	0	0	0	0	28.57	66.66	4.77
MARLA [5]	Serial	0	0	4.76	95.24	0	0	0	0	0
MALA [proposed]	Serial	0	0	28.57	71.43	0	0	0	0	0

is totally clear from the table that the multi-agent algorithm has much lower computation times in contrast with heuristic ones. If we take it into consideration more precisely, we can find out that our proposed algorithm has a higher distribution over [0.5, 0.7) and a lower distribution over [0.7, 1). This means that our proposed method suggests solutions with shorter computation times in most of the cases when we compare it with the other heuristic algorithms.

Finally, to compare the performance of our algorithm with other algorithms we have chosen five datasets from the PSPLIB. Then, we have calculated the optimal solutions found by each of the algorithms, the average deviation from the optimal solutions and also the maximum deviation in all of the experiments. The outcomes are all gathered in Table 3. Considering this table, it can be well understood that the population-based algorithm which is based on particle swarm outperforms the others up to the point that the number of activities do not exceed from twelve in j12 dataset.

Putting all the outcomes together, it is clear that the proposed algorithm has higher computational time but better efficiency in comparison with the other competitive multi-agent solution [5] for the static multi-mode resource-constrained project scheduling problem. The utmost performance of the suggested algorithm can be obtained in a real life problem where the AON network is not static and also some other assumptions have been regarded.

5 Conclusions and Future Work

Various applications of project scheduling can be found in different economic environments such as development projects, construction engineering, software development, and also make-to-order companies. Consequently, in this atmosphere which is extremely competitive, if one can develop efficient algorithms which are able to deal with various execution modes for activities, they'll play an important role in decision making process. Specifically, those which have considered real life conditions in their solutions. Furthermore, other optimization areas

Table 3 Comparison of proficiency in different algorithms

Algorithm	SGS	j10			j12			j16			j18			j20		
		Opt. (%)	Avg. Dev. (%)	Max. Dev. (%)	Opt. (%)	Avg. Dev. (%)	Max. Dev. (%)	Opt. (%)	Avg. Dev. (%)	Max. Dev. (%)	Opt. (%)	Avg. Dev. (%)	Max. Dev. (%)	Opt. (%)	Avg. Dev. (%)	Max. Dev. (%)
CPSO [16]	Serial	99.25	0.03	0.05	98.47	0.09	0.12	85.91	0.44	0.47	79.89	0.89	0.92	74.19	1.10	1.13
HGA [17]	Both	98.51	0.06	0.09	96.53	0.17	0.19	90.00	0.41	0.44	84.96	0.63	0.65	80.32	0.87	0.91
MARLA [5]	Serial	98.70	0.05	0.06	98.17	0.10	0.13	92.18	0.24	0.26	86.23	0.21	0.23	81.59	0.85	0.88
MALA [proposed]	Serial	98.73	0.05	0.06	98.25	0.09	0.11	92.31	0.22	0.24	86.42	0.18	0.20	81.71	0.80	0.83

such as bin packing and knapsack problem can make use of the solutions proposed for this problem [18].

Indeed, the MMRCPSP problem is a really challenging problem and in this study a novel multi-agent learning algorithm has been developed to solve this problem. First of all, we assign an agent to each activity in the AON network. Then we construct a local frame and develop it step by step as the project goes on. Then, the agents make their decisions based on their learning automata (rationality) or their local dispatcher (heuristic-based randomness). The decisions are the next activity to be visited and also the execution mode of the current activity. The partial schedules constructed from the local frame are used to update the action probability vectors of learning automata in all the agents of the frame.

To evaluate our new method, we have applied it on numerous datasets from the PSPLIB which are the most popular datasets for this problem. The experimental outcomes demonstrate that our algorithm works better than the other approaches in the performance of solutions for the MMRCPSP problem. Moreover, it consumes less computational time than the other two heuristic methods we have considered for our simulations.

References

1. Blazewicz, J., Lenstra, J.K., Kan, A.H.G.R.: Scheduling subject to resource constraints: classification and complexity. Discrete Appl. Math. **5**, 11–24 (1983)
2. Mika M, Waligora G, Weeglarz J.: Simulated annealing and tabu search for multi-mode resource-constrained project scheduling with positive discounted cash flows and different payment models. Eur. J. Oper. Res. **164**(3 SPEC), 639–668 (2005)
3. Slowinski, R., Soniewicki, B., Weglarz, J.: DSS for multi objective project scheduling. Eur. J. Oper. Res. **79**, 220–229 (1994)
4. Hartmann, S., Drexl, A.: Project scheduling with multiple modes. A comparison of exact algorithms. Networks **32**(4), 283–297 (1998)
5. Wauters, T., Verbeeck, K.: Vanden Berghe G, De Causmaecher P.: Learning agents for the multi-mode project scheduling problem. J Oper Res Soc **62**, 281–290 (2011)
6. Hartmann, S.: Project scheduling with multiple modes: a genetic algorithm. Ann. Oper. Res. **102**(1–4), 111–135 (2001)
7. Bouleimen, K., Lecocq, H.: A new efficient simulated annealing algorithm for the resource-constrained project scheduling problem and its multiple mode version. Eur. J. Oper. Res. **149**(2), 268–281 (2003)
8. Merkle, D., Middendorf, M., Schmeck, H.: Ant colony optimization for resource-constrained project scheduling. IEEE Trans. Evol. Comput. **6**, 333–346 (2002)
9. Ziarati, K., Akbari, R., Zeighami, V.: On the performance of bee algorithms for resource-constrained project scheduling problem. Appl Soft Comput J **11**(4), 3720–3733 (2011)
10. Valls, V., Quintanilla, M. S., Ballestin, F.: Resource-constrained project scheduling: a critical activity reordering heuristic. Eur. J. Oper. Res. Forthcoming (2004)
11. Debels D, Reyck B. De, Leus R, Vanhoucke M.: A hybrid scatter search/Electromagnetism meta–heuristic for project scheduling. Eur. J. Oper. Res. To appear (2004)
12. Kelley Jr, J.E.: The critical-path method: resources planning and scheduling. In: Industrial Scheduling. Prentice-Hall, New Jersey, pp. 347–365 (1963)

13. Bedworth, D., Bailey, J.: Integrated production control systems management, analysis design. Wiley, New York (1982)
14. Verbeeck, K., Nowe; A., Vrancx, P., Peeters, M.: Reinforcement learning theory and applications. Multi-automata learning, Chapter 9. I-Tech Education and Publishing: Vienna, pp 167–185 (2008)
15. Project Scheduling Problem Library. http://129.187.106.231/psplib/main.html
16. Jarboui, B., Damak, N., Siarry, P., Rebai, A.: A combinatorial particle swarm optimization for solving multi-mode resource-constrained project scheduling problems. Appl. Math. Comput. **195**(1), 299–308 (2008)
17. Lova, A., Tormos, P., Cervantes, M., Barber, F.: An efficient hybrid genetic algorithm for scheduling projects with resource constraints and multiple execution modes. Int. J. Prod. Econ. **117**(2), 302–316 (2009)
18. Hartmann, S.: Packing problems and project scheduling models: an integrating perspective. J Oper Res Soc **51**, 1083–1092 (2000)
19. Jedrzejowicz P, Ratajczak-Ropel E.: Solving the RCPSP/max problem by the team of agents. In: Agent and Multi-Agent Systems: Technologies and Applications, Lecture Notes in Computer Science, Volume 5559/2009, pp. 734–743 (2009)