Expert Systems with Applications 42 (2015) 6039–6049

Contents lists available at ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

A multi-agent optimization algorithm for resource constrained project scheduling problem

Xiao-long Zheng, Ling Wang [*]

Tsinghua National Laboratory for Information Science and Technology (TNList), Department of Automation, Tsinghua University, Beijing 100084, China

ARTICLE INFO

Article history:
Available online 10 April 2015

Keywords:
Resource constrained project scheduling problem
Multi-agent system
Optimization algorithm

ABSTRACT

In this paper, a multi-agent optimization algorithm (MAOA) is proposed for solving the resource-constrained project scheduling problem (RCPSP). In the MAOA, multiple agents work in a grouped environment where each agent represents a feasible solution. The evolution of agents is achieved by using four main elements in the MAOA, including social behavior, autonomous behavior, self-learning, and environment adjustment. The social behavior includes the global one and the local one for performing exploration. Through the global social behavior, the leader agent in every group is guided by the global best leader. Through the local social behavior, each agent is guided by its own leader agent. Through the autonomous behavior, each agent exploits its own neighborhood. Through the self-learning, the best agent performs an intensified search to further exploit the promising region. Meanwhile, some agents perform migration among groups to adjust the environment dynamically for information sharing. The implementation of the MAOA for solving the RCPSP is presented in detail, and the effect of key parameters of the MAOA is investigated based on the Taguchi method of design of experiment. Numerical testing results are provided by using three sets of benchmarking instances. The comparisons to the existing algorithms demonstrate the effectiveness of the proposed MAOA for solving the RCPSP.

1. Introduction

The resource constrained project scheduling problem (RCPSP) is to schedule project activities over time under a limitation of available resources (Brucker, Drexl, Mohring, Neumann, & Pesch, 1999). The RCPSP is one of the most intractable NP-hard problems in the field of operation research and management science (Mohring, Schulz, Stork, & Uetz, 2003). The RCPSP is very common in a variety of engineering fields, such as medical research (Hartmann, 1997), and software development (Alba & Chicano, 2007). Many scheduling problems can be formulated as the special cases of the RCPSP, such as job-shop scheduling, flow-shop scheduling, open-shop scheduling and project scheduling (Leung, 2004). It is very important for both academic research and engineering applications to develop effective solution algorithms for solving the RCPSP.

Over the past few decades, the RCPSP has attracted increasing attention due to its challenge and extensive engineering background. Early solution methods mainly focus on heuristic priority rules, e.g., minimum slack (MSLK), latest start time (LST) (Davis & Patterson, 1975), earliest start time (EST), shortest processing time (SPT), latest finish time (LFT), most total successors (MTS) (Cooper, 1977), worst case slack (WCS), and resource scheduling method (RSM) (Kolisch, 1996). To obtain the solutions with a higher quality, meta-heuristics have been widely adopted during recent years. Relevant research work in this area includes the following. Hartmann (1998) proposed a permutation-based genetic algorithm (GA) by employing a regret-based sampling method and LFT rule for initializing the population, where the proposed single point crossover that can always generate feasible offspring. Hartmann (2002) further developed a self-adapting GA by adaptively adopting a parallel and serial scheduling generating scheme (SGS). Bouleimen and Lecocq (2003) proposed an activity list based simulating annealing (SA) algorithm based on an alternated activity and time incrementing process. Zhang, Li, and Li (2005) proposed a particle swarm optimization (PSO) algorithm with both the priority-based and the permutation-based representation, showing that the permutation-based PSO outperformed the priority-based PSO. Debels and Vanhoucke (2007) presented an effective decomposition-based GA with a resource-based crossover. Chen, Shi, Teng, Lan, and Hu (2010) proposed an efficient hybrid algorithm by combining ant colony optimization (ACO) and scatter search (SS), and Lam & Li, 2012 proposed a chemical reaction optimization (CRO). Shi, Qu, Chen, et al. (2010) proposed an artificial bee colony (ABC) algorithm, where each solution was represented by a

* Corresponding author. Tel.: +86 10 62783125; fax: +86 10 62786911.
E-mail addresses: zhengxl11@mails.tsinghua.edu.cn (X.-l. Zheng), wangling@mail.tsinghua.edu.cn (L. Wang).

http://dx.doi.org/10.1016/j.eswa.2015.04.009
0957-4174/© 2015 Elsevier Ltd. All rights reserved.

random key. Wu, Wan, Shukla, and Li (2011) proposed a chaos-based improved immune algorithm. Fang and Wang (2012) proposed a shuffled frog-leaping algorithm (SFLA) by using an extended activity list representation, and they presented a speed-up method for evaluating new solutions based on some theoretical analysis. Paraskevopoulos, Tarantilis, and Ioannou (2012) proposed an event list-based evolutionary algorithm. More recently, Zamani (2013) proposed a competitive GA with a magnet-based crossover operator, Yannibelli and Amandi (2013) proposed a hybrid method by integrating the simulated annealing into evolutionary algorithm, Faghihi, Reinschmidt, and Kang (2014) proposed a novel approach of construction project application of the GA based on building information model, and Fahmy, Hassan, and Bassioni (2014) proposed a PSO with stacking justification for solving the RCPSP. Although many methods have been developed, it is still a challenge to solve the large-scale RCPSP effectively.

Multi-agent system (MAS) is an active research topic in artificial intelligence and expert systems. In a reasonable way, multiple agents collaborate with potential advantages for solving complex problems. The MAS based methods have been used to solve the RCPSP (Ren & Wang, 2011) and showed good performance, where an agent represented a resource and the behaviors of agents like collaboration, learning were adopted to handle the RCPSP. Moreover, motivated by swarm intelligence, several population-based algorithms (Liu, Wang, Liu, & Wang, 2011) have been widely applied to solve optimization problems. The intention of this research is to design a new multi-agent optimization algorithm (MAOA) by adopting the key behaviors of agent and the population-based mechanism of swarm intelligence to solve the RCPSP effectively. To be specific, each agent represents a feasible solution, which collaborates with others in a grouped environment via social behavior, autonomous behavior, self-learning, and environment adjustment. Through these elements, agents perform exploration and exploitation in a collaborative way among search space to obtain the promising solutions. We present the detailed implementation of the MAOA and the numerical comparisons with the existing algorithms for solving the RCPSP. The results show that the proposed MAOA is very competitive, especially for solving the large-scale problems.

The rest of the paper is organized as follows: In Section 2 the RCPSP is described; In Section 3, the framework of the multi-agent optimization algorithm is provided after introducing the concepts of agent and multi-agent system. In Section 4, the detailed design of the MAOA for solving the RCPSP is presented. Numerical tests and comparisons as well as the investigation on the effect of parameter setting are given in Section 5. Finally, the paper is ended with some conclusions and future work in Section 6.

2. Resource constrained project scheduling problem

Generally, the RCPSP can be described as follows: A project consists of J activities, where each activity j ($j = 1,2,\ldots,J$) can be processed with a duration d_j without preemption. There exists precedence relationship between several activities, which implies that activity j cannot be started until its predecessors are all finished. Moreover, there are K types of renewable resources available for the project. For each resource k ($k = 1,2,\ldots,K$), its availability at any time is R_k. The activity j requires r_{jk} units of resource k during its period of duration. The dummy activities 0 and $J+1$ represent the beginning and the end of the project, respectively. It is assumed that $d_0 = d_{J+1} = 0$ and such dummy activities consume no resource. The objective is to determine the start time of each activity to minimize the maximum complete time (makespan) of the project. For mathematical models of the RCPSP, please refer to Leung (2004).

In Fig. 1, a simple RCPSP with 6 activities is illustrated. For simplicity, only one type of resource is considered with an available amount of 2. In Fig. 2, a feasible schedule for the project in Fig. 1 with a makespan value of 12 is shown.

3. Multi-agent optimization algorithm

3.1. Agent and multi-agent system

Agent is a concept in the field of artificial intelligence (Minsky, 1988). An agent can be viewed as a computer system situated in a certain environment, such as a physical system or internet, with flexible autonomous action (Wooldridge & Jennings, 1995). Agents perceive input information though sensors and then perform resultant actions to affect the environment (Jennings, Sycara, & Wooldridge, 1998), as shown in Fig. 3 (Wooldridge, 2009). Agents are also capable of making decisions of choosing suitable actions to realize the goal independently without external intervention from other agents or human interference. Meanwhile, agent may have a variety of possible actions, including responsiveness, pro-activeness and social behaviors (Wooldridge & Jennings, 1995). Each agent can receive the information from the external

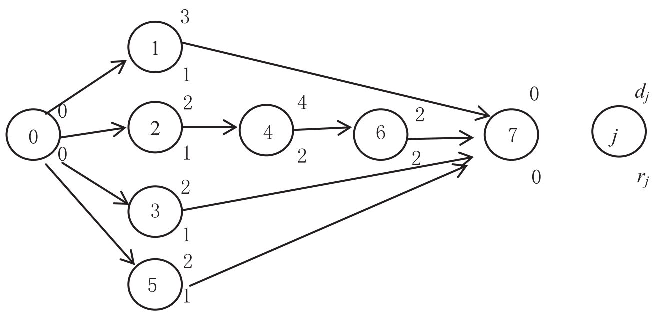


Fig. 1. An example of a project.

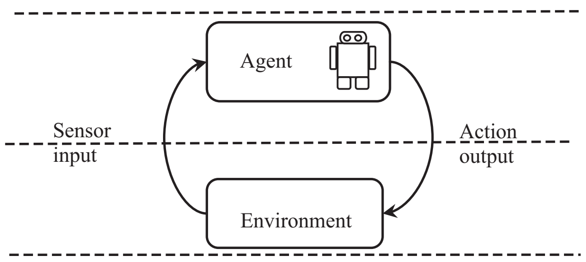


Fig. 3. An abstract view of an agent.

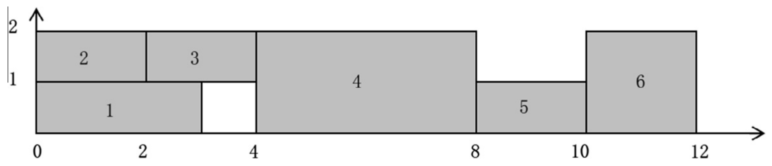


Fig. 2. A feasible schedule.

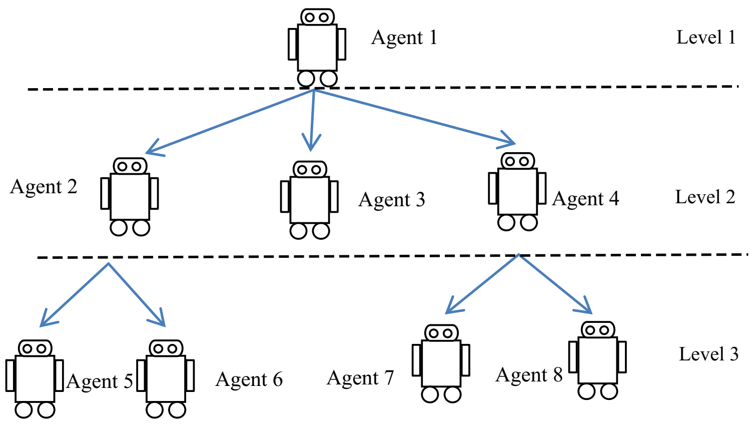


Fig. 4. Hierarchical organization.

environment and react in a timely manner to affect or adapt to the environment. Besides, each agent interacts with the external sources and other agents through social behavior.

Multi-agent system (MAS) is a group of loosely connected autonomous agents that act in a certain environment (Balaji & Srinivasan, 2010). A MAS usually contains three elements: a set of agents, $A = \{a_1, a_2, \ldots, a_n\}$, an environment where agents live and act, and a set of reactive rules that govern the interaction between agents and the environment (Liu, Jing, & Tang, 2002). The agents retain their inherent characteristics and also communicate with other agents through social behavior. Agents are usually well organized in a certain architecture, such as hierarchical organization, holonic organization, coalition and group, as shown in Figs. 4–7, respectively (Balaji & Srinivasan, 2010). To improve the overall performance, effective rules can be used for agents to well interact with the environment and other agents. Next, we will present the MAOA inspired by the mechanism of the multi-agent system.

3.2. Multi-agent optimization algorithm (MAOA)

The key characteristics of a multi-agent system include environment, behaviors of agents, and interactions among agents. Inspired by the mechanism of the MAS including autonomous behavior, social behavior and self-learning behavior of agents, we will design a multi-agent based optimization algorithm. In addition, the interaction between agents and environment is also considered, which is simulated through adjustment of the environment.

3.2.1. Environment and its adjustment

In the MAOA, each agent denotes a solution. The relationships between individual agents are depicted by an organized architecture. All the agents and their relationships form an environment. To solve a specific optimization problem, the representation of an agent and the organized architecture should be defined. In this paper, a grouped structure with N groups is employed, as shown in Fig. 8, where each group consists of S agents. For each group, the best agent of the group is elected as the leader.

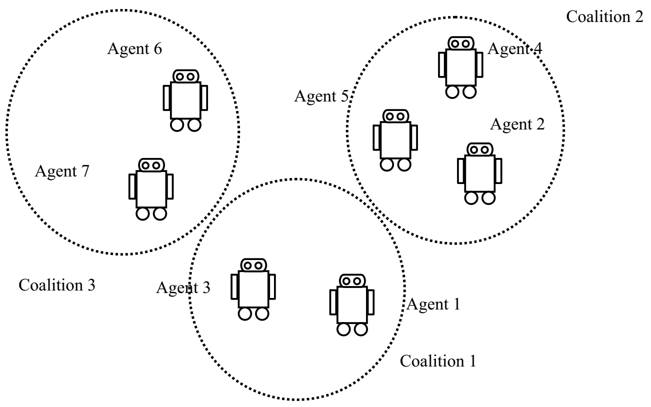


Fig. 6. Coalition organization.

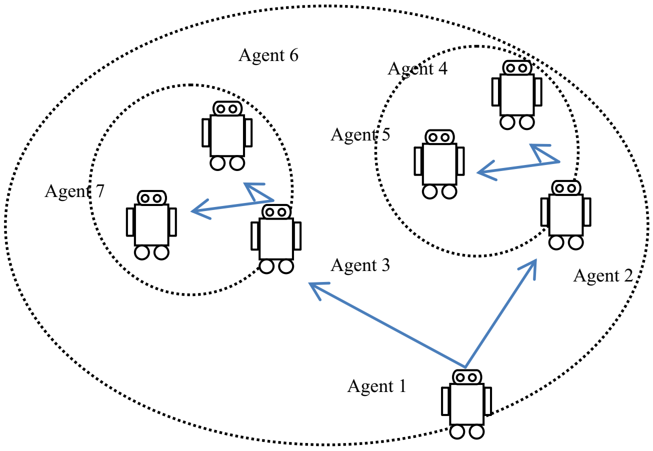


Fig. 5. Holonic organization.

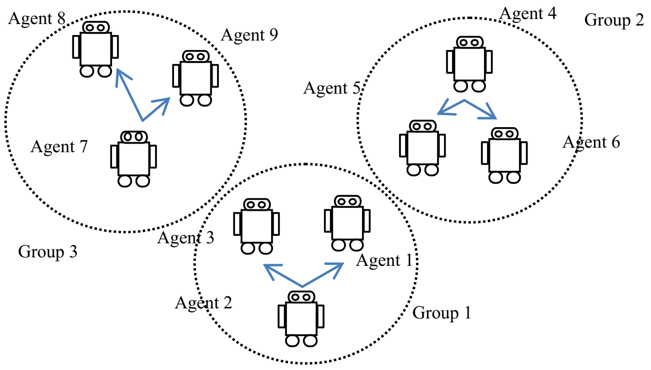


Fig. 7. Group organization.

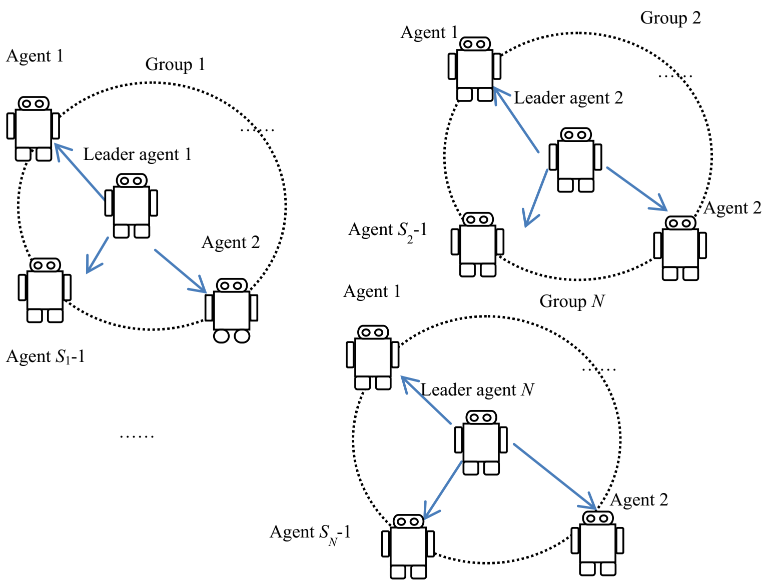


Fig. 8. Leader-group organization.

By adjusting the environment, it is useful for the agents to well search the solution space. In the MAOA, environment is adjusted by re-grouping all the agents. For each group one by one, to be specific, it exchanges the second best agent in each group (namely active agent) with the worst agent in the group with the best leader agent (namely elite group). With such adjustment, information is shared among groups and the search behavior within or among groups can be adjusted accordingly so as to enhance the search variety.

3.2.2. Social behavior, autonomous behavior and self-learning

In a multi-agent system, agents cooperate, coordinate, and negotiate with one another (Wooldridge, 2009). Coordination and cooperation among agents can be viewed as a kind of social behavior. The MAOA adopts two phases of cooperation behaviors, including the collaborative interaction in the whole environment (global social behavior) and the cooperative interaction in the local group (local social behavior). For the global social behavior, the leader agent of the elite group cooperates with all the leader agents in other groups one by one to well explore the whole solution space, as shown in Fig. 9. For the local social behavior, the leader agent of each group interacts with other agents in the same group to further exploit their neighborhoods, as illustrated in Fig. 10.

Apart from the social behavior, each agent is also capable of acting independently without external intervention, i.e., autonomy. In the MAOA, autonomous behavior is realized by random neighborhood exploitation and greedy selection. That is, each agent exploits its neighborhood randomly, and then accepts the new neighbor with a better quality.

Moreover, agent can be improved by learning from the acquired knowledge (Zhong, Liu, Xue, & Jiao, 2004). In the MAOA, a self-learning procedure is used by a special problem-dependent local search. via such a kind of self-learning, the exploitation with some problem characteristics can be enhanced.

3.2.3. MAOA

With the above description, a framework of the MAOA is illustrated as Fig. 11.

In the MAOA, an environment is firstly constructed by dividing all the agents into several groups; then, agents perform evolution through the social behavior as the global exploration as well as through the autonomous behavior and the self-learning as the local exploitation; by moving agents among groups, environment can be adjusted so that information can be shared for further well evolution with the agent-based search behaviors. Next, we will present the detailed implementation of the MAOA for solving the RCPSP.

4. MAOA for RCPSP

4.1. Encoding and the initialization of environment

To solve the RCPSP, the extended activity list (EAL) (Fang & Wang, 2012) is used to represent agent, which contains three lists: (1) an activity list $\pi = \{\pi_1, \pi_2, \ldots, \pi_J\}$; (2) start time of each

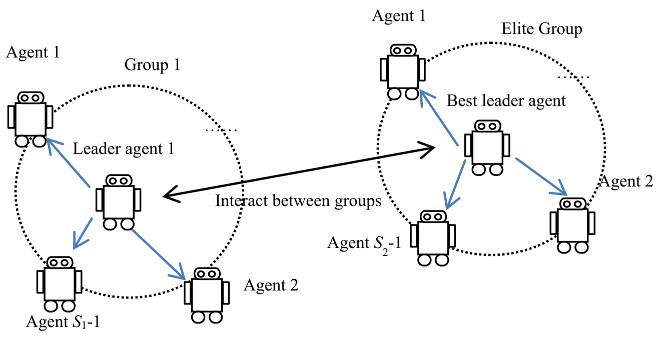


Fig. 9. Global social behavior of agents.

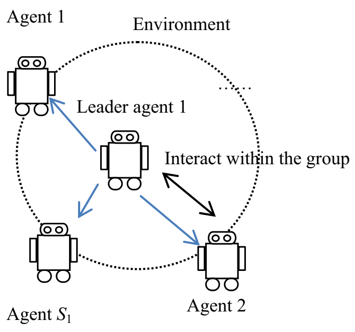


Fig. 10. Local social behavior of agents.

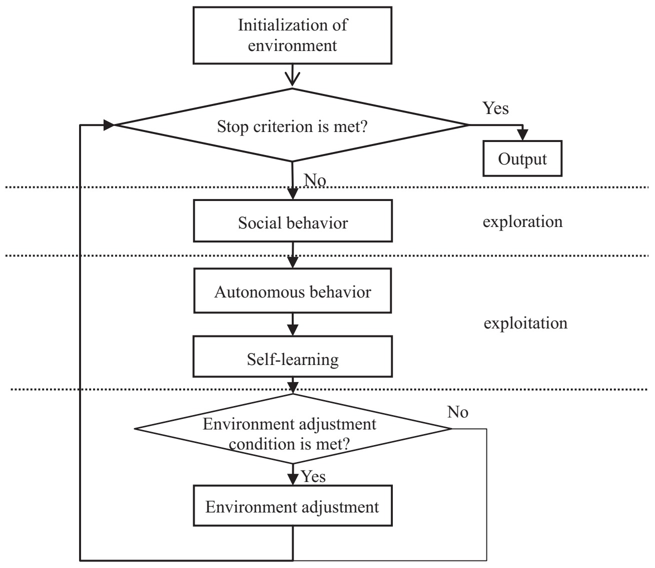


Fig. 11. Flowchart of the MAOA.

activity $\{s_1,s_2,\ldots,s_J,\}$; (3) finish time of each activity $\{f_1,f_2,\ldots,f_J,\}$. Fang and Wang (2012) pointed out that search operators with such an encoding scheme can perform efficiently.

To obtain good initial agents in the MAOA, agents are initialized using the regret-based biased random sample method with the LFT priority rule (Hartmann, 1998). To be specific, the activity list of each agent is constructed by repeatedly selecting an unscheduled activity from the eligible activity set (denoted as D), which contains all the activities whose predecessors have already been scheduled. To determine which unscheduled activity in D is to be selected at each decision stage, a roulette wheel selection strategy is employed and the LFT priority rule is used to generate the following probability η_j of being chosen for eligible activity j:

$$\eta_j = (\mu_j + 1) \Big/ \sum_{i \in D} (\mu_i + 1) \tag{1}$$

where $\mu_j = \max_{i \in D} LFT_i - LFT_j + 1$ is the priority value of activity j and LFT_j is the latest finish time of activity j.

After the activity list of each agent is constructed, the serial scheduling generating scheme (SSGS) (Kolisch & Hartmann, 1999) is adopted to evaluate the activity list and transform it to a schedule. Once a schedule is generated, the start and finish times of each activity are obtained. Since these times are needed when performing search operators, we adopt list (2) and list (3) to store them. In such a way, computational efficiency can be improved, since it takes too much computational time by using SSGS each time to obtain start and finish times.

After agents are initialized, they are randomly divided into N groups, where each group consists of S agents. That is, the size of the whole population is $N \times S$.

4.2. Social behavior

For the RCPSP, Zamani (2013) designed a magnet-based crossover (MBCO) with the advantage that the offspring can inherit partial sequences from both parents. Moreover, a set of offspring can be obtained via MBCO to achieve well exploration. Encouraged by the advantage of the MBCO, it is employed as the global social behavior to perform the interaction between the leader agent in each group and the global best leader agent to generate a set of offspring. Then, the best offspring agent will be adopted to replace the corresponding leader agent with probability ρ. To be specific, the best generated offspring replaces the leader agent if the offspring is better or a random number $rd \in (0,1)$ is larger than ρ; otherwise, the offspring is abandoned. The procedure of the MBCO is as follows:

Step 1: Randomly determine a contiguous block of the activity list, called Block, in the global best leader agent (denoted as π^d).

Step 2: Find the first activity of the leader agent (denoted as π^r) that belongs to the Block. Denote the location index of the found activity in π^r as q_1. Then, copy π^r_k $(k = 1,\ldots,q_1)$ to the offspring directly.

Step 3: Divide the activities π^r_k $(k = q_1 + 1,\ldots,J,\ \pi^r_k \notin \text{Block})$ into three categories: predecessor activity, successor activity and free activity according to whether the activity is a predecessor or successor of a block activity. For the predecessor activities and successor activities, they are located before and after the Block one by one as the same order in π^r, respectively. For each free activity that is neither a predecessor nor a successor of any block activity, it is located before and after the Block respectively to obtain a set of offspring.

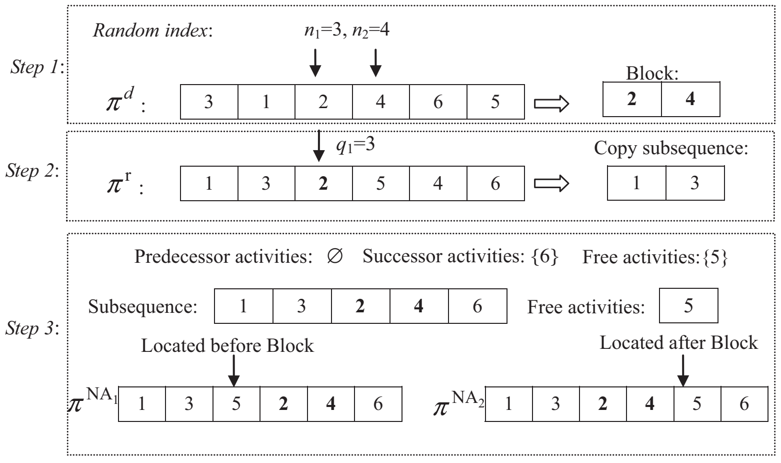
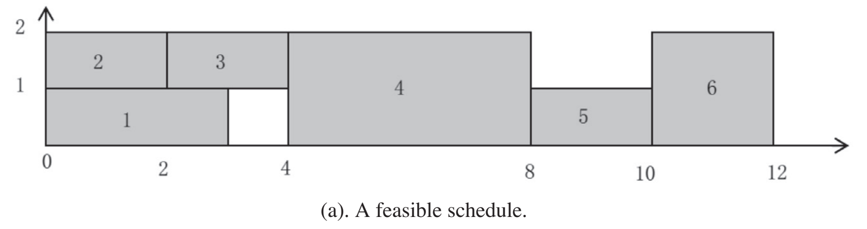


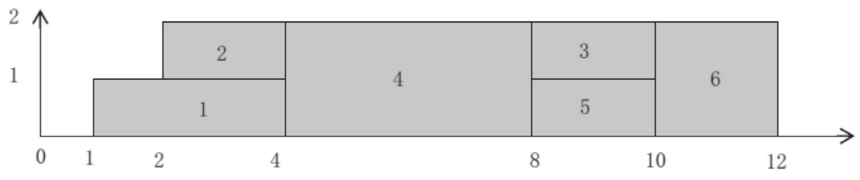
Fig. 12. An example of the MBCO.

```
Procedure of social behavior
$A_{gb}$ is the global best agent and belongs to group $k$, $A_{bj}$ is the leader agent in group $j$.
FOR group $i$=1 to $N$ and $i \neq k$ //global social behavior
    Perform MBCO between $A_{bi}$ and $A_{gb}$
    Select the best generated agent $NA_b$
    IF $NA_b$ is better than $A_{bi}$
        Replace $A_{bi}$ with $NA_b$
    ELSE generate a random number $rd \in (0,1)$
      IF $rd > \rho$
          Replace $A_{bi}$ with $NA_b$
      END IF
    END IF
END FOR
FOR group $i$=1 to $N$ //local social behavior
    FOR agent $A_j$ $j$=1 to $S$ and $A_j \neq A_{bi}$
        Perform RBCO between $A_j$ and $A_{bi}$
        Replace $A_j$ with the generated agent if it is better than $A_j$
    END FOR
END FOR
```

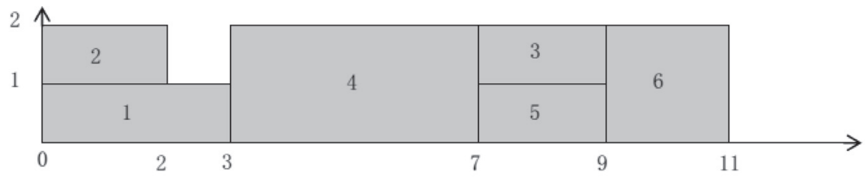
Fig. 13. Procedure of social behavior.



(a). A feasible schedule.



(b). Forward scheduling.



(c). Backward scheduling.

Fig. 14. A single iteration of FBI.

For the project in Fig. 1, an example of the MBCO is illustrated in Fig. 12. Assume $\pi^d = \{3, 1, 2, 4, 6, 5\}$, $\pi^r = \{1, 3, 2, 5, 4, 6\}$, and the randomly determined Block is $\{\mathbf{2}, \mathbf{4}\}$. The first activity in π^r that belongs to the Block is activity 2, and the corresponding location q_1 is 3. Then, the subsequence {1, 3} is copied to the offspring directly, and activities {5, 6} are classified, where 6 is a successor activity and 5 is a free activity. Thus, 6 should be located after the Block, and 5 can be located before or after the Block. So, it obtains two offspring $\pi^{NA_1} = \{1, 3, 5, \mathbf{2}, \mathbf{4}, 6\}$ and $\pi^{NA_2} = \{1, 3, \mathbf{2}, \mathbf{4}, 5, 6\}$.

The activity sub-sequence with a higher average resource utilization rate can be inherited through the resource-based crossover (RBCO) (Fang & Wang, 2012). It was pointed out that makespan is inversely proportional to the average resource utilization (Fang &

```
Procedure of environment adjustment
$A_{gb}$ is the global best agent and belongs to group $k$, $A_{wk}$ is the worst agent in group $k$.
FOR group $i$=1 to $N$ and $i \neq k$
    Rank the agents in group $i$ and determine the active agent $A_{ai}$
    Determine $A_{wk}$
    $A_{ai}$ moves to group $k$ and $A_{wk}$ moves to group $i$
END FOR
```

Fig. 15. Procedure of environment adjustment.

Wang, 2012). The agent can be improved with the guidance of a better agent via RBCO. So, we adopt the RBCO as the local social behavior for each agent to improve itself with the guidance of the leader agent in its own group. The RBCO is performed between an agent M and the leader agent F as follows:

Step 1: A time interval with length L is randomly selected for agent F between $0.25MS_F$ and $0.75MS_F$, where MS_F is the makespan of F. Then, determine time $t_1 \in [0, MS_F - L]$ such that time interval $[t_1, t_1 + L]$ has a minimal resource utilization rate (RUR) which is calculated as follows:

$$RUR = \frac{1}{K}\sum_{k=1}^{K}\left\{\frac{1}{R_k}\sum_{t\in(t_1,t_1+L)}\sum_{j\in A(t)} r_{jk}\right\} \tag{2}$$

where $A(t)$ is the set of activities being processed at time t.

Step 2: Determine two positions p_1 and p_2 (crossover points), $1 \leqslant p_1 \leqslant p_2 \leqslant J$, where p_1 is the maximal position that satisfies $f_{p_1} \leqslant t_1$ and p_2 is the minimal position that satisfiess$_{p_2} \geqslant t_1 + L$. Then, generate the new agent π^{NA} as follows:

$$\pi_i^{NA} := \pi_i^F, \quad i = \{1,2,\ldots,p_1\} \cup \{p_2,\ldots,J\} \tag{3}$$

$$\pi_i^{NA} := \pi_j^M, \quad j = \min\{\pi_j^M \notin \pi_k^{NA}, \quad k = 1,2,\ldots,i-1\} \tag{4}$$

The procedure of social behavior in the MAOA with the above global behavior and local behavior is shown in Fig. 13.

4.3. Autonomous behavior

The permutation based swap (PBS) is widely used as local search in solving the RCPSP (Chen et al., 2010). To perform the autonomous behavior for each agent, the permutation based swap operator PBS(π_i, π_{i+1}) is employed in the MAOA. To be specific, it randomly selects two adjacent activities (π_i, π_{i+1}) that have no precedence relationship, and then swaps such two activities to generate a new agent. Clearly, the PBS operator does not break the precedence constraint of an EAL. Therefore, the newly generated agent is still feasible.

4.4. Self-learning

According to the characteristics of the RCPSP, Li and Willis (1992) presented a forward–backward improvement (FBI) to adjust a solution by iteratively adopting the SGS forward and backward until the makespan cannot be further reduced. For the forward scheduling, the activities are shifted to right as much as possible in a descending order of finish times; for the backward scheduling, activities are shifted to left in an increasing order of start times. A single iteration of the FBI for the schedule in Fig. 2 is illustrated in Fig. 14.

In the MAOA, the above FBI is used as the self-learning for the best leader agent to perform further exploitation.

4.5. Environment adjustment

With the social behavior, autonomous behavior and self-learning, agents can be improved through the self-evolution and the cooperation within or among groups. To share the information among different groups, environment needs to be adjusted. In the MAOA, environment is adjusted every $T = 10$ generations. The procedure of the environment adjustment is illustrated in Fig. 15.

5. Experimental results

We code the MAOA in C++ and run it on a PC with 2.3 GHz CPU. To compare it with typical existing algorithms, we use three well-known data sets PSPLIB for numerical tests, which were generated by the problem generator ProGen designed by Kolisch and Sprecher (1997). The benchmark data sets include 480 J30 instances (each with 30 activities), 480 J60 instances (each with 60 activities), and 600 J120 instances (each with 120 activities). Generally, the average percentage deviation from the lower bound (AveDevLB) is used as performance metrics for comparison. The optimal solutions for J30 instances and the critical-path length reported by Stinson, Davis, and Khumawala (1978) for J60 and J120 instances are widely used as lower bounds, which can be got from the website of PSPLIB: http://www.om-db.wi.tum.de/psplib/datasm.html. To be specific, the AveDevLB is defined as follows:

$$AveDevLB = \frac{1}{R}\sum_{i=1}^{R}\frac{MS_i - LB_i}{LB_i} \tag{5}$$

where MS_i is the makespan of the ith instance obtained by an algorithm, LB_i is the lower bound known of the ith instance, and R is the number of the used instances. Clearly, the smaller the AveDevLB is, the better the algorithm is.

Table 1
Combination of parameter values.

Parameters	Factor level			
	1	2	3	4
N	2	5	7	10
S	5	10	15	20
ρ	0.6	0.7	0.8	0.9

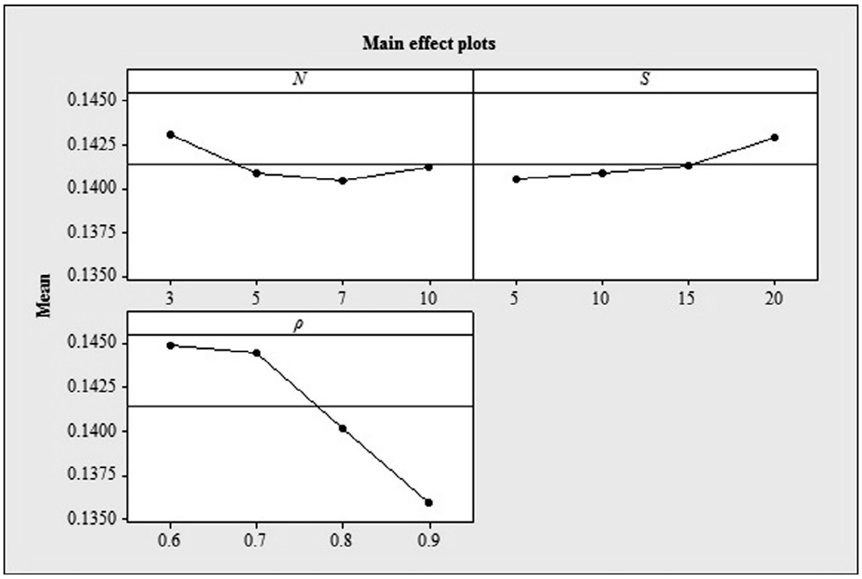


Fig. 16. Factor level trend of the MAOA for J30.

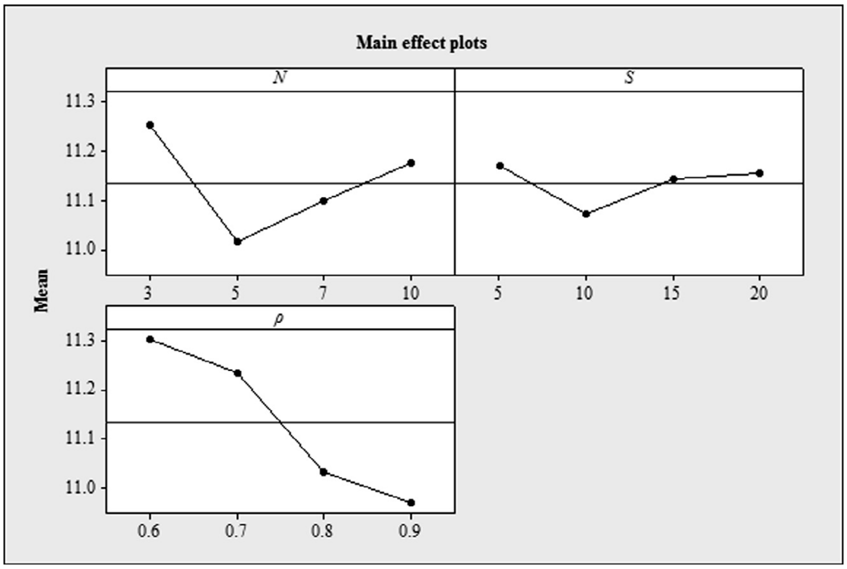


Fig. 17. Factor level trend of the MAOA for J60.

Table 2
Orthogonal table and responsive value for the MAOA.

Experiment number	Factors			ARV (%)		
	N	S	ρ	J30	J60	J120
1	1	1	1	0.1461	11.5239	34.1836
2	1	2	2	0.1454	11.3365	33.7435
3	1	3	3	0.1422	11.1417	33.7883
4	1	4	4	0.1387	11.0108	33.6737
5	2	1	2	0.1423	11.1522	33.7905
6	2	2	1	0.1446	11.0031	33.8101
7	2	3	4	0.1358	10.9135	33.1563
8	2	4	3	0.1407	10.9924	33.4574
9	3	1	3	0.1389	11.0002	33.6371
10	3	2	4	0.1345	10.9477	33.2819
11	3	3	1	0.1426	11.2544	33.8613
12	3	4	2	0.1457	11.1869	33.8757
13	4	1	4	0.1348	11.0054	33.5442
14	4	2	3	0.1388	10.9968	33.6129
15	4	3	2	0.1446	11.2642	33.7921
16	4	4	1	0.1465	11.4329	34.0032

5.1. Parameters setting

The MAOA contains three key parameters: the number of groups (N), the group size (S) and the acceptance probability ρ used in the MBCO. To investigate the effect of these parameters on the performance of the MAOA, we carry out test by using the Taguchi method of design-of-experiment (DOE) (Montgomery, 2005). We set 4 levels for each of the three parameters as Table 1. For each combination of the orthogonal array $L_{16}(4^3)$, we randomly choose 48 instances from J30, J60 and 60 instances from J120 as Fang and Wang (2012) to calculate average response variable (ARV) values as Eq. (5). For each run, the maximum number of the generated schedules is set as 5000. The numerical results are listed in Table 2, and the trend of each parameter for different sets is illustrated in Figs. 16–18.

It can be seen from Figs. 16–18 that medium values are better for N and S, while ρ prefers to a large value. Since the size of the whole population is $N \times S$, small N and S lead to a small population so that the exploration ability at each generation will be weakened,

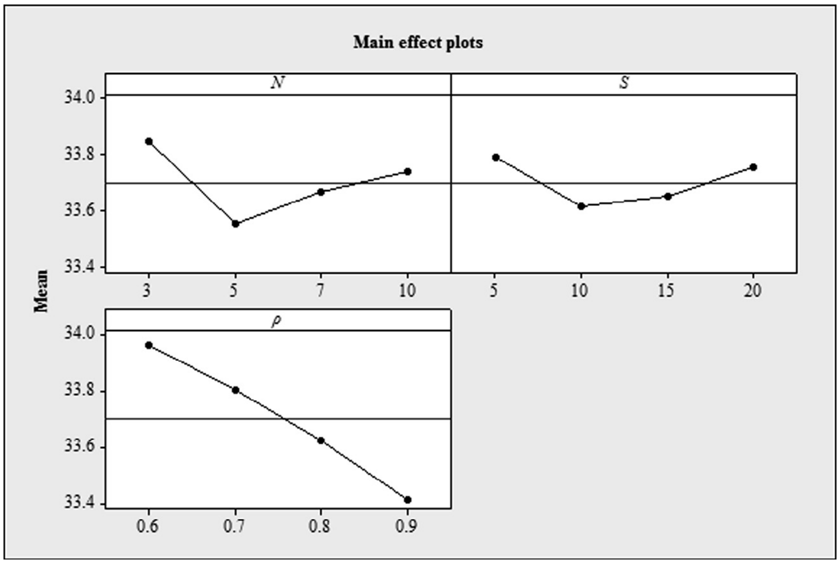


Fig. 18. Factor level trend of the MAOA for J120.

Table 3
The best combination of parameters for the MAOA.

Dataset	N	S	ρ
J30	7	5	0.9
J60	5	10	0.9
J120	5	10	0.9

while large N and S lead to a large population so that the whole evolution of the population will not be sufficient. Medium N and S may tradeoff the population size and the evolution generations so as to balance the exploration and exploitation. As for ρ, a small value will make the algorithm accept inferior new solutions too much. According to the results of DOE tests, the recommended values of three parameters for J30, J60, and J120 are listed in Table 3, respectively.

5.2. Results of MAOA

Same as the literature, the performances of the MAOA are tested by setting the maximum number of generated schedules as 1000, 5000, 50,000, respectively. For each set, the algorithm is run independently with all the instances. The obtained AveDveLB and the average CPU times (AveTime) for J30, J60 and J120 are shown in Table 4, respectively.

From Table 4, it can be seen that the MAOA has a better performance as more schedules are generated. It implies that the MAOA keeps the potential in finding better solutions if more schedules can be explored. Moreover, the running time of the MAOA is acceptable. Even for the large-scale J120 instances, the average running time is less than 3 min. Besides, the running time increases almost linearly with respect to the total number of the generated schedules.

Table 5
Average deviations (%) for J30.

Algorithms	Maximum number of schedules		
	1000	5000	50,000
Kochetov and Stolyar (2003)	0.10	0.04	0
Mendes, Goncalves, and Resende (2009)	0.06	0.02	0.01
Chen et al. (2010)	0.14	0.06	0.01
MAOA	0.17	0.06	0.01
Debels, Reyck, Leus, and Vanhoucke (2006)	0.27	0.11	0.01
Debels and Vanhoucke (2007)	0.12	0.04	0.02
Fahmy et al. (2014)	0.22	0.05	0.02
Valls, Ballestin, and Quintanilla (2008)	0.27	0.06	0.02
Alcaraz, Maroto, and Ruiz (2004)	0.25	0.06	0.03
Agarwal, Colak, and Erenguc (2011)	0.13	0.1	–
Nonobe and Ibaraki (2002)	0.46	0.16	0.05
Hartmann (2002)	0.38	0.22	0.08
Coelho and Tavares (2003)	0.74	0.33	0.16
Fang and Wang (2012)	0.36	0.21	0.18
Hartmann (1998)	1.03	0.56	0.23

5.3. Comparisons of MAOA with existing algorithms

Next, we compare the MAOA with 14 existing algorithms from the literature. The AveDevLB values of all the comparative algorithms for data sets J30, J60, J120 are listed in Tables 5–7, respectively.

For J30 set, the MAOA ranked the 6th with 1000 schedules, the 5th with 5000 schedules and the 2nd with 50,000 schedules among all the 15 algorithms. The average makespan obtained by the MAOA with 50,000 schedules is much close to the optimal bound with only 0.01% average deviation from the low bound. Therefore, it can be concluded that the MAOA is an effective solver for the RCPSP with small scale. For J60 set, the MAOA ranked the

Table 4
Results for the MAOA.

Data set	1000 schedules		5000 schedules		50000 schedules	
	AveDveLB (%)	AveTime (s)	AveDveLB (%)	AveTime (s)	AveDveLB (%)	AveTime (s)
J30	0.17	0.10	0.06	0.51	0.01	5.02
J60	11.67	0.76	10.84	3.42	10.64	33.54
J120	33.87	3.41	32.64	16.74	31.02	171.33

Table 6
Average deviations (%) for J60.

Algorithms	Maximum number of schedules		
	1000	5000	50,000
MAOA	11.67	10.84	10.64
Fang and Wang (2012)	11.44	10.87	10.66
Chen, Shi, Teng, Lan, and Hu (2010)	11.75	10.98	10.67
Mendes et al. (2009)	11.72	11.04	10.67
Debels and Vanhoucke (2007)	11.31	10.95	10.68
Debels et al. (2006)	11.73	11.10	10.71
Valls et al. (2008)	11.56	11.10	10.73
Kochetov and Stolyar (2003)	11.71	11.17	10.74
Alcaraz et al. (2004)	11.89	11.19	10.84
Fahmy, Hassan, and Bassioni (2014)	11.86	11.19	10.85
Agarwal et al. (2011)	11.51	11.29	–
Hartmann (2002)	12.21	11.70	11.21
Nonobe and Ibaraki (2002)	12.97	12.18	11.58
Hartmann (1998)	13.3	12.74	12.26
Coelho and Tavares (2003)	13.8	13.31	12.83
Kolisch and Drexl (1996)	13.51	13.06	–

Table 7
Average deviations (%) for J120.

Algorithms	Maximum number of schedules		
	1000	5000	50,000
Chen et al. (2010)	35.19	32.48	30.56
Debels and Vanhoucke (2007)	33.55	32.18	30.69
MAOA	33.87	32.64	31.02
Fang and Wang (2012)	34.83	33.20	31.11
Valls et al. (2008)	34.07	32.54	31.24
Fahmy, Hassan, and Bassioni (2014)	35.60	33.78	32.40
Mendes et al. (2009)	35.87	33.03	31.44
Debels et al. (2006)	35.22	33.10	31.57
Alcaraz et al. (2004)	36.53	33.91	31.57
Kochetov and Stolyar (2003)	34.74	33.36	32.06
Agarwal et al. (2011)	34.65	34.15	–
Hartmann (2002)	37.19	35.39	33.21
Nonobe and Ibaraki (2002)	40.86	37.88	35.85
Hartmann (1998)	39.93	38.49	36.51
Coelho and Tavares (2003)	41.36	40.46	39.41

5th with 1000 schedules, the 1st with 5000 schedules and the 1st with 50,000 schedules among all the 15 algorithms. For J120 set, the MAOA ranked the 2nd with 1000 schedules, the 4th with 5000 schedules and the 3rd with 50,000 schedules among all the 15 algorithms. The gap between the MAOA and the best algorithm is less than 0.5%.

So, it can be concluded that the MAOA is effective in solving the RCPSP with medium and large scales.

6. Conclusion

The main contribution of this paper is to present an effective optimization algorithm inspired by multi-agent system and swarm intelligence for solving the RCPSP. Search operators including MBCO, RBCO, PBS and FBI are designed according to the social behavior, autonomous behavior and self-learning inspired from multi-agent system, and agents in the population are dynamically re-grouped to adjust the environment. The effect of key parameters of the MAOA is investigated, and the numerical tests using sets of benchmark instances are provided. The comparisons with 14 existing algorithms show that the proposed MAOA is more effective in solving the RCPSP with medium and large scales. With this research, it shows that expert and artificial intelligent systems like multi-agent system can be well used to develop effective optimization algorithms for solving hard problems.

As for the limitations of this research, MAOA is a simple imitation of multi-agent systems, and more reasonable search operators inspired by the intelligent behaviors should be further studied. Besides, as an optimization algorithm, the convergence of the MAOA and the adaptive adjustment of parameter setting need to be investigated. Future research could focus on the following aspects:

(1) Build the mathematical model of the MAOA and analyze the convergence property.
(2) Design self-adaptive strategies for parameter setting and search behavior.
(3) Develop multi-objective MAOA.
(4) Extend the application of the MAOA to other kinds of complex optimization problems.

Acknowledgments

This research is partially supported by National Key Basic Research and Development Program of China (2013CB329503), National Science Foundation of China (61174189), and Doctoral Program Foundation of Institutions of Higher Education of China (20130002110057).

References

Agarwal, A., Colak, S., & Erenguc, S. (2011). A neurogenetic approach for the resource constrained project scheduling problem. *Computer and Operations Research, 38*(1), 44–50.
Alba, E., & Chicano, J. F. (2007). Software project management with GAs. *Information Science, 177*(10), 2380–2401.
Alcaraz, J., Maroto, C., & Ruiz, R. (2004). Improving the performance of genetic algorithms for the RCPS problem. In *Proceedings of the 9th international workshop on project management and scheduling* (pp. 40–43).
Balaji, P. G., & Srinivasan, D. (2010). An introduction to multi-agent systems. In *Innovations in Multi-Agent Systems and Applications*. New York, Berlin Heidelberg: Springer.
Bouleimen, K., & 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), 268–821.
Brucker, P., Drexl, A., Mohring, R., Neumann, K., & Pesch, E. (1999). Resource-constrained project scheduling: Notation, classification, models, and methods. *European Journal of Operational Research, 112*(1), 3–41.
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*(5), 1031–1039.
Coelho, J., & Tavares, L. (2003). Comparative analysis of meta-heuristics for the resource constrained project scheduling problem. Technical Report, Department of Civil Engineering, Instituto Superior Tecnico.
Cooper, D. F. (1977). A note on serial and parallel heuristics for resource-constrained project scheduling. *Foundations of Control Engineering, 2*(4), 131–133.
Davis, E. W., & Patterson, J. H. (1975). A comparison of heuristic and optimum solutions in resource-constrained project scheduling. *Management Science, 21*(7), 944–955.
Debels, D., Reyck, B., Leus, R., & Vanhoucke, M. (2006). A hybrid scatter-search electromagnetism meta-heuristic for the resource-constrained project scheduling problem. *European Journal of Operational Research, 169*(3), 638–653.
Debels, D., & Vanhoucke, M. (2007). A decomposition-based genetic algorithm for the resource-constrained project scheduling problem. *Operations Research, 55*(3), 457–469.
Faghihi, V., Reinschmidt, K. F., & Kang, J. H. (2014). Construction scheduling using genetic algorithm based on building information model. *Expert Systems with Applications, 41*(16), 7565–7578.
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.
Fang, C., & Wang, L. (2012). An effective shuffled frog-leaping algorithm for resource-constrained project scheduling problem. *Computers & Operations Research, 39*(5), 890–901.
Hartmann, S. (1997). Scheduling medical research experiments: An application of project scheduling methods. Technical Report, University Kiel, Germany.
Hartmann, S. (1998). A competitive genetic algorithm for resource-constrained project scheduling. *Naval Research Logistics, 45*(6), 733–750.
Hartmann, S. (2002). A self-adapting genetic algorithm for project scheduling under resource constraints. *Naval Research Logistics, 49*(5), 433–448.

Jennings, N. R., Sycara, K., & Wooldridge, M. (1998). A roadmap of agent research and development. *Autonomous agents and multi-agent systems, 1*(1), 7–38.

Kochetov, Y., & Stolyar, A. (2003). Evolutionary local search with variable neighborhood for the resource constrained project scheduling problem. In *Proceedings of the 3rd international workshop of computer science and information technologies*. Russia.

Kolisch, R. (1996). Efficient priority rules for the resource-constrained project scheduling problem. *Journal of Operations Management, 14*(3), 179–192.

Kolisch, R., & Drexl, A. (1996). Adaptive search for solving hard project scheduling problems. *Naval Research Logistics, 43*, 23–40.

Kolisch, R., & Hartmann, S. (1999). Heuristic algorithms for solving the resource constrained project scheduling problem: Classification, computational, analysis. In *Project scheduling: Recent models, algorithms and applications* (pp. 147–178). Berlin: Kluwer.

Kolisch, R., & Sprecher, A. (1997). PSPLIB – a project scheduling problem library. *European Journal of Operational Research, 96*(1), 205–216.

Lam, A. Y. S., & Li, V. O. K. (2012). Chemical-reaction-inspired metaheuristic for optimization. *IEEE Transactions on Evolutionary Computation, 14*(3), 381–399.

Leung, J. Y. (2004). *Handbook of scheduling: Algorithms, models, and performance analysis*. Boca Raton, FL: CRC.

Li, K. Y., & Willis, R. J. (1992). An iterative scheduling technique for resource constrained project scheduling. *European Journal of Operational Research, 56*(3), 370–379.

Liu, J. M., Jing, H., & Tang, Y. Y. (2002). Multi-agent oriented constraint satisfaction. *Artificial Intelligence, 136*(1), 101–144.

Liu, B., Wang, L., Liu, Y., & Wang, S. Y. (2011). A unified framework for population-based metaheuristics. *Annals of Operations Research, 186*(1), 231–262.

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

Minsky, M. (1988). *Society of mind*. New York: Simon and Schuster.

Mohring, R. H., Schulz, A. S., Stork, F., & Uetz, M. (2003). Solving project scheduling problems by minimum cut computations. *Management Science, 49*(3), 330–350.

Montgomery, D. C. (2005). *Design and analysis of experiments*. Arizona: John Wiley & Sons.

Nonobe, K., & Ibaraki, T. (2002). Formulation and tabu search algorithm for the resource constrained project scheduling problem (RCPSP). In *Essays and surveys in meta-heuristics* (pp. 557–588). Boston: Kluwer.

Paraskevopoulos, D. C., Tarantilis, C. D., & Ioannou, G. (2012). Solving project scheduling problems with resource constraints via an event list-based evolutionary algorithm. *Expert Systems with Applications, 39*(4), 3983–3994.

Ren, H., & Wang, Y. (2011). A survey of multi-agent methods for solving resource constrained project scheduling problems. In *Proceedings of international conference on management and service science* (pp. 1–4).

Shi, Y., Qu, F. Z., Chen, W., et al. (2010). An artificial bee colony with random key for resource-constrained project scheduling. In *Life system modeling and intelligent computing* (pp. 148–157). New York, Berlin Heidelberg: Springer.

Stinson, J. P., Davis, E. W., & Khumawala, B. M. (1978). Multiple resource-constrained scheduling using branch and bound. *IIE Transactions, 10*(3), 23–40.

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.

Wooldridge, M. (2009). *An introduction to multiagent systems*. Arizona: John Wiley & Sons.

Wooldridge, M., & Jennings, N. R. (1995). Intelligent agents: Theory and practice. *The Knowledge Engineering Review, 10*(2), 115–152.

Wu, S. S., Wan, H. D., Shukla, S. K., & Li, B. Z. (2011). Chaos-based improved immune algorithm (CBIIA) for resource-constrained project scheduling problems. *Expert Systems with Applications, 38*(4), 3387–3395.

Yannibelli, V., & Amandi, A. (2013). Hybridizing a multi-objective simulated annealing algorithm with a multi-objective evolutionary algorithm to solve a multi-objective project scheduling problem. *Expert Systems with Applications, 40*(7), 2421–2434.

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. (2005). Particle swarm optimization-based schemes for resource constrained project scheduling. *Automation in Construction, 14*(3), 393–404.

Zhong, W. C., Liu, J., Xue, M. Z., & Jiao, L. C. (2004). A multi-agent genetic algorithm for global numerical optimization. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, 34*(2), 229–244.