Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2014, Article ID 302684, 9 pages
http://dx.doi.org/10.1155/2014/302684

Research Article

A Multiagent Evolutionary Algorithm for the Resource-Constrained Project Portfolio Selection and Scheduling Problem

Yongyi Shou, Wenwen Xiang, Ying Li, and Weijian Yao

School of Management, Zhejiang University, Hangzhou 310058, China

Correspondence should be addressed to Yongyi Shou; yshou@zju.edu.cn

Received 5 November 2013; Revised 6 February 2014; Accepted 10 February 2014; Published 22 April 2014

Academic Editor: Jyh-Hong Chou

A multiagent evolutionary algorithm is proposed to solve the resource-constrained project portfolio selection and scheduling problem. The proposed algorithm has a dual level structure. In the upper level a set of agents make decisions to select appropriate project portfolios. Each agent selects its project portfolio independently. The neighborhood competition operator and self-learning operator are designed to improve the agent's energy, that is, the portfolio profit. In the lower level the selected projects are scheduled simultaneously and completion times are computed to estimate the expected portfolio profit. A priority rule-based heuristic is used by each agent to solve the multiproject scheduling problem. A set of instances were generated systematically from the widely used Patterson set. Computational experiments confirmed that the proposed evolutionary algorithm is effective for the resource-constrained project portfolio selection and scheduling problem.

1. Introduction

The project portfolio selection problem (PPSP), together with its various extensions, has been widely studied during the last decade. Given a set of project proposals and constraints, the traditional PPSP is to select a subset of project proposals to optimize the organization's performance objective [1]. Mathematic models have been proposed in the literature. The project portfolio profit is regarded as the natural performance objective and utilized by most models, for example, the zero-one integer programming model [2]. Since the PPSP is an NP-hard problem [3], metaheuristic algorithms such as evolutionary approaches are widely used [4].

Most studies on the PPSP generally dissever the inherent relationship between portfolio selection and project scheduling. The traditional PPSP is based on some assumptions. It is assumed that an individual project has a fixed and unchangeable schedule [5]; hence, only the project selection decision is considered to impact the final portfolio profit. However, project scheduling tends to affect the portfolio feasibility by adjusting the start and completion time of its activities [6]. Especially when resources are constrained, scheduling of project activities helps to better utilize the limited resources, and consequently to increase the portfolio profit [5]. Inclusion of project activity scheduling as a subproblem of project portfolio selection helps improve the overall organization performance even though it increases the complexity of decision making. This combined problem is termed as the resource-constrained project portfolio selection and scheduling problem (RCPPSSP). The RCPPSSP can be described as a problem to select an optimal portfolio of projects and schedule their activities to maximize an organization's stated objectives without exceeding available resources or violating other constraints [7].

The RCPPSSP has attracted increasing attention in recent years as a new research problem. Owing to the dual level structure of the RCPPSSP, most algorithms in the current literature are also composed of two parts. In the upper level, decisions are made to select project portfolios. In the lower level, procedures of multiproject scheduling are adopted to improve the performance of the selected portfolio. Due to the NP-hard nature of the project portfolio selection problem, researchers developed heuristics and metaheuristics to improve the solution quality and computational efficiency.

For example, an implicit enumeration procedure was developed for all possible project priority sequences with high profit [7]. An ant colony optimization (ACO) based on the max-min ant system [8] was proposed, in which solutions were encoded as walks of agents in a construction graph, and transition probabilities were computed to determine the probability of an arc of the graph being chosen by the agents in the next iteration [8]. An iterative multiunit combinatorial auction algorithm [5] was also used to select project portfolios through a distributed bidding mechanism. In the lower level, heuristics such as greedy heuristic [8] and priority rule-based heuristics [9] are widely adopted for multiproject scheduling.

In recent years, agent-based computation has been widely applied in distributed problem solving. An agent is a self-contained problem solving entity [10] which exhibits the properties of autonomy, social ability, responsiveness, and proactiveness [11]. In a multiagent optimization system (MAOS) [12], self-organization agents [13] interact to optimize their own problem solving with limited declarative knowledge and simple procedural knowledge under ecological rationality [12]. Specifically, agents explore in parallel through three types of interactions, namely, cooperation, coordination, and negotiation [14]. Since interactions among the agents contribute to solution diversity and rapid convergence in some cases [15], it is recommended to embed the MAOS in evolutionary algorithms to improve the solution quality [12, 15–18]. Recently, multiagent evolutionary algorithms have been used for single project scheduling [19].

The objective of this paper is to develop a multiagent evolutionary algorithm for the RCPPSSP. The master procedure in the upper decision level is designed by combining the neighborhood competition operator and self-learning operator in the multiagent system. A priority rule-based heuristic is adopted for the subprocedure in the lower decision level. In Section 2, we present the resource-constrained project portfolio selection and scheduling problem and its mathematical model. Section 3 explains the multiagent evolutionary algorithm that we have developed for the RCPPSP. Computational experiments and results are discussed in Section 4. Finally, we conclude this paper in Section 5.

2. Project Portfolio Selection and Scheduling

The objective of the resource-constrained project portfolio selection and scheduling problem is to maximize the project portfolio profit. The problem can be recognized as a dual level decision problem [8].

The upper level is to select feasible project portfolios under resource constraints. There are a set Ω of N candidate projects from which we select the optimal portfolio. A pool of K types of limited and renewable resources is available for all projects. It is assumed that there is no other relationship among the projects besides resource competition.

The lower level is to solve the multiproject scheduling problem, that is, to determine the start (completion) time of each activity without violating the precedence relations or resource constraints [5]. Given the resource constraints, scheduling project activities within a portfolio helps shorten the project duration and increase the portfolio profit since the project profit is a decreasing function of the project completion time [7].

Two sets of decision variables are designed in this paper: x_i is for project selection and c_{ijt} for project activity scheduling, as shown in the following formulae:

$$x_i = \begin{cases} 1, & \text{if project } i \text{ is included,} \\ 0, & \text{if project } i \text{ is excluded.} \end{cases}$$

$$c_{ijt} = \begin{cases} 1, & \text{if activity } j \text{ of project } i \text{ completes at time } t, \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

The notations used in this paper are listed in Notations for the RCPPSSP section.

The RCPPSSP can be formulated as a 0-1 integer programming model [5, 7]:

$$\max \quad \sum_{i=1}^{N} \sum_{t=EF_{iJ_i}}^{LF_{iJ_i}} b_{it} c_{iJ_i t} \tag{2}$$

$$\text{s.t.} \quad \sum_{t=EF_{iJ_i}}^{LF_{iJ_i}} c_{ijt} = x_i, \quad \forall i, j \tag{3}$$

$$\sum_{t=EF_{iJ_i}}^{LF_{iJ_i}} t c_{iJ_i t \le T_i}, \quad \forall i \tag{4}$$

$$\sum_{t=EF_{ij}}^{LF_{ij}} t c_{ijt} \le \sum_{t=EF_{ih}}^{LF_{ih}} \left(t - d_{ih}\right) c_{iht}, \quad \forall (i,j), \forall (i,h) \in S_{ij} \tag{5}$$

$$\sum_{(i,j) \in A_t} x_i r_{ijk} \le R_k, \quad \forall k, t \tag{6}$$

$$x_i \in \{0,1\}, \quad \forall i \tag{7}$$

$$c_{ijt} \in \{0,1\}, \quad \forall (i,j), \forall t. \tag{8}$$

The objective (2) is to maximize the total profit of the selected project portfolio. Constraint (3) ensures that all activities of a selected project are completed. It also enforces that every activity in unselected projects is not executed. Constraint (4) is to guarantee the selected project is complete before its deadline. Constraint (5) describes the precedence relations among activities which requires an activity to start only after all its predecessors have been completed. Constraint (6) ensures that in each time period the demand on any resource does not exceed its capacity. Formulae (7) and (8) declare the decision variables.

3. Multiagent Evolutionary Algorithm

To solve the RCPPSSP, a multiagent evolutionary algorithm (MAEA) is proposed. Corresponding to the dual level structure of the RCPPSSP, the MAEA has two levels as well.

The master procedure for the upper level is to select project portfolios and a priority rule-based heuristic [20] is designed as the subprocedure for the lower level to do multiproject scheduling.

3.1. Project Portfolio Selection. The multiagent evolutionary algorithm is a combination of two theories: multiagent systems and evolutionary algorithms [17]. Generally, a multiagent system [15] is composed of an environment, a set of objects, a set of agents, a set of relations between objects (agents), and a set of operations. During observation of the environment and interaction with other agents, the fitness value of an agent can be estimated and optimized on the basis of the possessed resources, abilities, and knowledge [17].

In this paper a multiagent evolutionary algorithm was proposed to solve the project selection problem. We designed a multiagent system in which each agent selects project portfolios according to its own preferences and environment. The evolution of the agents is realized by the neighborhood competition and self-learning operators. In neighborhood competition, loser agents will be replaced by new generated agents. In this way, the information and knowledge of individual agents will spread to the whole system. The winner agents will conduct self-learning by applying its own knowledge, for which a simple genetic algorithm is developed. Since most multiagent systems adopt the real-valued representation which is not appropriate for the project selection problem, we designed an agent system based on a discrete representation and modified the operators correspondingly.

3.1.1. Multiagent System. The objective function (2) of the RCPPSSP can be simplified as the following formula:

$$\max f(\mathbf{x}), \quad \mathbf{x} = (x_1, x_2, \ldots, x_N) \in \Theta, \tag{9}$$

where $f(\mathbf{x})$ denotes the portfolio profit which is equal to $\sum_{i=1}^{N} \sum_{t=EF_{ij_i}}^{LF_{ij_i}} b_{it} c_{J_i t}$ and Θ is an N-dimensional search space of the project selection problem. The "$\mathbf{x}$" in boldface represents a vector which is a candidate solution in the search space. The component "x_i" is a 0-1 variable which takes the value of 1 when project i is selected or the value of 0 otherwise.

An agent for the RCPPSSP can be defined as follows.

Definition 1. An agent denoted as $\mathbf{a}$ represents a candidate solution $\mathbf{x}$ to the RCPPSSP. The value of its energy is equal to its value of the objective function in (2):

$$\begin{aligned} \mathbf{a} = \mathbf{x} &= (x_1, x_2, \ldots, x_N) \in \Theta, \\ \text{Energy}(\mathbf{a}) &= f(\mathbf{x}). \end{aligned} \tag{10}$$

The agent $\mathbf{a}$ living in an environment makes decisions autonomously to increase its energy as much as possible. To realize the local perceptivity of agents, the environment is organized as a lattice-like structure which can be defined as follows [16].

Definition 2. All agents live in a lattice-like environment denoted as L. The size of L is $L_{\text{size}} \times L_{\text{size}}$. L_{size} is determined as $L_{\text{size}} = \rho \times N + 1$, where $0 < \rho < 1$.

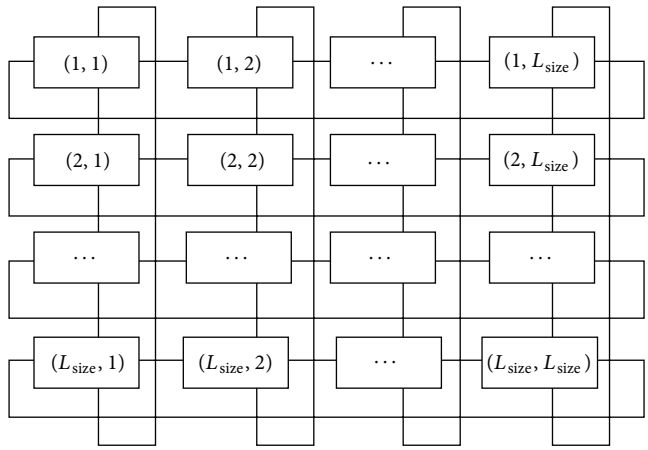

Figure 1: Illustration of an agent lattice.

Figure 1 illustrates an agent lattice. In the agent lattice, an agent denoted as $\mathbf{a}_{uv} = (a_1, a_2, \ldots, a_N)$ is fixed on a lattice point (u, v), where $1 \le u, v \le L_{\text{size}}$.

3.1.2. Neighborhood Competition Operator. In the agent lattice, agents compete with their neighbors to gain more resources so that their purposes, that is, objectives of the RCPPSSP, can be achieved. Reference [16] noted that the neighborhood competition operator facilitates information diffusion to the whole lattice. To describe the neighborhood competition operator, we define the neighborhood as follows.

Definition 3. All agents with a line or diagonal line connecting to agent $\mathbf{a}$ constitute the neighborhood of agent $\mathbf{a}$. The competing neighborhood of $\mathbf{a}_{uv}$ is denoted as N_{uv}^{C}. The perceptive range of an agent's competing neighborhood R_C determines the number of competing neighbors as $(2R_C + 1)^2 - 1$, where $0 \le R_C \le (1/2)L_{\text{size}}$.

For example, when R_C is equal to 1, the number of agents in the neighborhood N_{uv}^{C} is 8.

The basic rule for neighborhood competition operator is defined as follows [16].

Rule 1. If the agent $\mathbf{a}_{uv}$ satisfies (11), it is a loser; otherwise, it is a winner:

$$\text{Energy}(\mathbf{a}_{uv}) < \text{Energy}(\mathbf{a}_{uv}^{*}). \tag{11}$$

The winner survives in the agent lattice, but the loser perishes and is replaced by a new agent $\mathbf{a}'_{uv}$ generated from the local-best agent $\mathbf{a}_{uv}^{*}$ which is defined as

$$\begin{aligned} \mathbf{a}_{uv}^{*} &= (a_1^{*}, a_2^{*}, \ldots, a_N^{*}) \in N_{uv}^{C}, \\ \text{Energy}(\mathbf{a}_{uv}^{*}) &\ge \text{Energy}(\mathbf{a}), \quad \forall \mathbf{a} \in N_{uv}^{C}. \end{aligned} \tag{12}$$

Two alternative strategies [16] are adopted to generate the new agent $\mathbf{a}'_{uv}$.

Strategy 1. A set D is composed of sequence numbers of the positions where the agent $\mathbf{a}_{uv}$ takes different values from

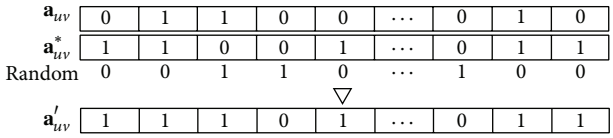

Figure 2: Strategy 1 to generate a new agent.

agent $\mathbf{a}^*_{uv}$; that is, $D = \{i \mid 1 \le i \le N, a_i \ne a^*_i\}$. The new agent $\mathbf{a}'_{uv} = (x'_1, x'_2, \ldots, x'_N)$ is determined by formula (13), in which $1 \le i \le N$ and *Random* takes the values of 0 or 1 randomly

$$a'_i = \begin{cases} a^*_i, & i \notin D \text{ or } (i \in D \text{ and Random} = 0), \\ 1 - a^*_i, & \text{otherwise.} \end{cases} \tag{13}$$

One example of Strategy 1 is presented in Figure 2.

Strategy 2. Mutation in the evolutionary algorithm is adopted to transform agent $\mathbf{a}^*_{uv}$ to $\mathbf{a}'_{uv}$, which is represented in formula (14). In formula (14), $1 \le i \le N$ and Random is generated randomly in the interval of (0,1)

$$a'_i = \begin{cases} a^*_i, & \text{Random} > \dfrac{1}{N}, \\ 1 - a^*_i, & \text{otherwise.} \end{cases} \tag{14}$$

One example of Strategy 2 is presented in Figure 3.

In this paper a uniform random parameter $D_S \in (0, 1)$ is used to determine which strategy is to be applied to generate the new agent. Firstly, we calculate the similarity between the loser agent $\mathbf{a}_{uv}$ and the local-best agent $\mathbf{a}^*_{uv}$ with maximum energy in the neighborhood by the following formula:

$$|D| = \sum_{i=1}^{N} \eta_i, \tag{15}$$

where $\eta_i = \begin{cases} 1, & a_i = a^*_i \\ 0, & a_i \ne a^*_i \end{cases}$. The higher the value of $|D|$ is, the more similar $\mathbf{a}_{uv}$ to $\mathbf{a}^*_{uv}$ is and the much lower the chance to get a better solution through Strategy 1 is indicated.

Therefore, the rule to select an appropriate strategy is designed as follows.

Rule 2. When the value of $|D|$ satisfies (16), Strategy 1 is adopted to generate new agents. Otherwise, Strategy 2 is adopted:

$$\frac{|D|}{N} < D_S. \tag{16}$$

3.1.3. Self-Learning Operator. In order to survive in competition, agents in the lattice may take actions to increase their energy by using their own knowledge [21]. The self-learning operator [16, 21] is designed to help agents achieve this purpose. It is assumed that only winner agents have the chance to conduct self-learning.

Definition 4. All agents with a line or diagonal line connecting to agent $\mathbf{a}$ constitute the neighborhood of agent $\mathbf{a}$. The self-learning neighborhood of $\mathbf{a}_{uv}$ is denoted as N^L_{uv}.

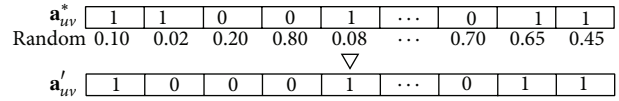

Figure 3: Strategy 2 to generate a new agent.

The perceptive range of an agent's self-learning neighborhood R_L determines the number of self-learning neighbors as $(2R_L + 1)^2 - 1$, where $0 \le R_L \le (1/2)L_{\text{size}}$.

The basic rule for the self-learning operator is defined as follows [16, 21].

Rule 3. If agent $\mathbf{a}_{uv}$ satisfies (17), it has the chance to execute the self-learning operator:

$$\text{Energy}(\mathbf{a}_{uv}) \ge \text{Energy}(\mathbf{a}), \quad \forall \mathbf{a} \in N^L_{uv}. \tag{17}$$

A simple genetic algorithm (SGA) [22] is adopted to realize the self-learning of agent $\mathbf{a}_{uv}$, in which the chromosome takes the same representation as an agent. The hth chromosome in the gth generation is denoted as follows:

$$\text{chromosome}^g_h = \mathbf{x} = (x_1, x_2, \ldots, x_N) \in \Theta, \tag{18}$$

where $0 \le g \le \text{MaxGA}$ and $1 \le h \le \text{PopSize}$. MaxGA denotes the maximum number of iterations and PopSize represents the population size. The fitness function of a chromosome is equal to the energy of its corresponding agent.

In the self-learning process, the initial population is generated as follows. The first chromosome in the initial population is equal to agent $\mathbf{a}_{uv}$ and all other chromosomes are generated by the following formula:

$$\text{chromosome}^0_h = \{y_1, y_2, \ldots, y_N\}, \quad 2 \le h \le \text{PopSize}, \tag{19}$$

where y_i is determined by

$$y_i = \begin{cases} 1 - x_i, & \text{Random} \le D_S, \\ x_i, & \text{Random} > D_S. \end{cases} \tag{20}$$

Three types of operators are applied to generate the new population in each generation, including selection, crossover, and mutation. Binary tournament [23] is used for selection. The elitist strategy [24] is employed to guarantee convergence of the genetic algorithm. One-point crossover is adopted. A crossover point is randomly selected and then the genes in the two parent chromosomes after the crossover point are exchanged to generate two offspring chromosomes [25]. The mutation locus is also selected randomly. The selected gene is then mutated by negation. The probabilities of crossover and mutation are denoted as p_c and p_m respectively.

3.1.4. Repair Mechanism. In the case of resource constraints and multiproject scheduling, it is possible that some agents (solutions) are infeasible. Especially the neighborhood competition operator and self-learning operator may lead to

infeasible agents. Therefore, repair mechanisms are necessary and have been proposed in the relevant literature [8].

In this paper, the infeasible agents are repaired by removing projects in sequence under a certain rule. In an RCPPSSP, the profit of a candidate project depends on its completion time which is unknown in advance, so it is inconvenient to apply other rules except the random rule [8]. Given an infeasible portfolio, a random project is selected and removed from the portfolio. This process continues until feasibility is achieved. Once the repaired portfolio is feasible, the new agent representing the repaired feasible portfolio is used to replace the incumbent infeasible agent.

3.2. Multiproject Scheduling. A priority rule-based heuristic is applied to schedule the selected projects in the lower decision level. Using the minimal slack (MINSLK) priority rule [26] and the serial schedule generation scheme (SSGS) [27], a multiproject schedule is generated for the selected portfolio. In a multiproject environment, the slack (SLK) of an activity is computed as follows:

$$\mathrm{SLK}_{ij} = \mathrm{LF}_{ij} - \mathrm{EF}_{ij}, \tag{21}$$

where LF and EF are the latest and earliest finish times, respectively, which are estimated by the critical path method (CPM).

Suppose a subset Φ is selected in the master procedure. In project $i \in \Phi$, there are J_i activities which are denoted as (i, j). The SSGS heuristic consists of $\sum_{i\in\Phi} J_i$ stages. In each stage g, two disjoint activity sets are identified [27]. The scheduled set S_g includes the activities which are already scheduled; and the decision set D_g contains the unscheduled activities whose predecessors are all in the scheduled set S_g. According to the MINSLK priority rule, the activity with the minimum slack is selected from the decision set and scheduled as early as possible without violating the resource constraints. The activity is then moved from the decision set to the scheduled set. Algorithm 1 shows the pseudocode of the priority rule-based heuristic.

To determine the feasibility of the multiproject schedule, we set a upper bound for each project's completion time, denoted as T_i. Specifically, for project i in the selected portfolio Φ, if its completion time goes beyond its upper bound T_i, the portfolio Φ is recognized as infeasible and shall be repaired.

4. Computational Experiments and Results

This section presents the experiment design and computational analyses for investigating the performances of the proposed multiagent evolutionary algorithm for the RCPPSSP. Based on the design of experiment (DOE) approach, a set of instances was generated systematically. Then parameter configurations of the algorithm were set up through testing of examples. The proposed MAEA was then compared with other algorithms in the literature.

4.1. Experiment Design. An RCPPSSP instance consists of a pool of candidate projects, a set of profit profiles of all projects, and the resources available for all projects. Three project pools with 10 projects in each pool and three other pools with 20 projects in each pool were generated randomly by 72 instances with three types of resources and different networks selected from the widely used Patterson set [28]. The above six project pools are denoted as PAT10_1, PAT10_2, PAT10_3, PAT20_1, PAT20_2, and PAT20_3, respectively.

Table 1: Experiment design.

	Profit decreasing rate (%)	Resource tightness (%)
Cell 1	2	30
Cell 2	2	60
Cell 3	8	30
Cell 4	8	60

It is assumed that a project achieves its base profit if it is complete at its critical path length (CPL) and the profit decreases with a profit decreasing rate λ_1 as its completion time increases. Referring to [7], the base profit B_i and actual profit b_{it} of project i are calculated by formulae (22) and (23), where σ_i denotes the resource utilization coefficient which is subject to a uniform distribution in the interval of [0.5, 1.5]. The upper bound of project completion time T_i is based on the critical path length by formula (24), and the relaxation rate λ_2 has the value of 0.4 in this paper. Resource tightness ω_k is introduced to estimate the capacity of resource k from its maximum resource demand $R_k^{\max}$ in the critical path method:

$$B_i = \sigma_i \sum_{k=1}^{K} \sum_{j=1}^{J_i} r_{ijk} d_{ij}, \tag{22}$$

$$b_{it} = B_i \left(1 - \lambda_1 \left(t - \mathrm{CPL}_i\right)\right), \tag{23}$$

$$T_i = \mathrm{CPL}_i + \min\left\{10, \lambda_2 \mathrm{CPL}_i\right\}, \tag{24}$$

$$R_k = \omega_k R_k^{\max}. \tag{25}$$

Two levels of profit decreasing rate λ_1 (2% or 8%) and resource tightness (30% or 60%) were designed, which forms four experiment cells as shown in Table 1. This 2^2 experimental design crossed with our six project pools yielded 24 instances to test the proposed algorithm.

4.2. Parameter Configurations. Parameters of the proposed MAEA were determined through experiments, including the maximum generations of agents MaxMA, the lattice size L_{size}, the coefficient D_S, and the perceptive ranges R_C and R_L. The parameters of the SGA for self-learning were also assigned, including the maximum generations MaxGA, the populationsize PopSize, the crossover probability p_c, and the mutation probability p_m.

The number of generations and the lattice size were determined firstly. With the increasing size of the agent lattice L and more generations of evolution, the multiagent algorithm is more likely to find the optimal solution in a longer computation time. Through testing on examples, MaxMA was set at 100 in this paper. The suitable lattice size is proportional to the number of candidate projects. We

Procedure of the priority rule-based heuristic
BEGIN
INIT: $S_0 := \{\}$
FOR $g := 1$ **TO** $\sum_{i \in \Phi} J_i$
COMPUTE D_g
SELECT (i^*, j^*) from D_g /∗ according to the MINSLK rule ∗/
ASSIGN $ST_{i^* j^*}$ /∗ as early as possible ∗/
$S_g := S_{g-1} \cup \{(i^*, j^*)\}$
ENDFOR
END

Algorithm 1: Pseudo-code of the priority rule-based heuristic.

Table 2: Profit data for problem instances.

Number	Cell	Project pool	Problem size	Algorithm		
				Ranking	MKP_ranking	MAEA
1	1	PAT10_1	10	7076.77	7076.77	7754.63
2	1	PAT10_2	10	6343.67	6343.67	7128.61
3	1	PAT10_3	10	7700.29	7700.29	8109.66
4	1	PAT20_1	20	13512.07	13512.07	14987.3
5	1	PAT20_2	20	10510.92	10510.92	11556.38
6	1	PAT20_3	20	11672.9	11672.9	12520.37
7	2	PAT10_1	10	6988.4	6988.4	7289.51
8	2	PAT10_2	10	6199.9	6199.9	6391.59
9	2	PAT10_3	10	7700.29	7700.29	7920.52
10	2	PAT20_1	20	12829.7	12829.7	13703.7
11	2	PAT20_2	20	10033.22	10033.22	10715.38
12	2	PAT20_3	20	11580.28	11580.28	11922.46
13	3	PAT10_1	10	4316.8	4316.8	4871.05
14	3	PAT10_2	10	3850.56	3850.56	4300.49
15	3	PAT10_3	10	4320.24	4320.24	5381.88
16	3	PAT20_1	20	7522.15	7558.27	9288.56
17	3	PAT20_2	20	6323.93	6323.93	7086.58
18	3	PAT20_3	20	7902.9	7902.9	8383.34
19	4	PAT10_1	10	3942.85	3942.85	4441.45
20	4	PAT10_2	10	3850.56	3850.56	3850.56
21	4	PAT10_3	10	4320.24	4320.24	4601.84
22	4	PAT20_1	20	7464.19	7485.53	8472.36
23	4	PAT20_2	20	6176.77	6176.77	6273.44
24	4	PAT20_3	20	7544.8	7544.8	7945.27
Average				7486.85	7489.24	8120.70

estimated $L_{\text{size}} = \rho \times N + 1$ by a coefficient $\rho = 0.4$ in this paper. Since the designed 24 instances have 10 or 20 projects, the lattice size L_{size} takes the values of 5 or 9, respectively.

According to Definitions 3 and 4, when $N = 10$, both perceptive ranges R_C and R_L belong to the set $\{1, 2\}$. Similarly, when $N = 20$, both R_C and R_L belong to the set $\{1, 2, 3, 4\}$. Our testing showed that the proposed MAEA has a good performance when $R_C = R_L = 1$.

The coefficient D_S is used to select the strategy of generating new agents to replace loser agents. When D_S is larger, Strategy 1 is more likely to be applied, which means the new agent will be more similar to the best agent in its neighborhood. Consequently, the multiagent algorithm may be easily trapped in a local optimum. However, the stability of the algorithm will be affected when D_S is much smaller. To tradeoff between the convergence and stability, we set D_S to 0.25 according to our experiments.

In summary, the parameters for the proposed multiagent evolutionary algorithm were determined as follows:

Table 3: Average computation times (in seconds).

	Ranking	MKP-Ranking	MAEA
Cell 1	0.018	0.061	18.155
Cell 2	0.019	0.064	19.060
Cell 3	0.025	0.082	262.155
Cell 4	0.026	0.085	250.638
Average	0.022	0.073	137.502

Table 4: Test statistics.

Algorithm	Ranking—MKP_Ranking	Ranking—MAEA	MKP_Ranking—MAEA
Z	−1.342[a]	−4.197[a]	−4.197[a]
Asymp. Sig. (2-tailed)	0.180	0.000	0.000

[a] Based on negative ranks.

MaxMA = 100, $\rho = 0.4$, $R_C = R_L = 1$, $D_S = 0.25$, MaxGA = 10, PopSize = 10, $p_c = 0.6$, and $p_m = 0.2$.

4.3. Computational Results and Comparison. For the above 24 instances, the profits achieved by the proposed MAEA are shown in Table 2, as well as the results of two other benchmarking algorithms in [29], namely, Ranking and MKP_ranking methods.

In the Ranking method, all candidate projects are ranked by a certain priority rule. Projects are then scheduled one by one according to their priorities until the next project makes the portfolio infeasible. It was reported that the greedy "max profit" rule performs best [29] and hence was used in this paper.

The MKP_ranking method also involves two stages [29]. In the first stage, the project selection problem is solved as a multidimensional 0-1 knapsack problem (MKP) and then all selected projects are prioritized by the "max profit" rule. In the second stage, all projects selected in the first stage are scheduled sequentially. In case a selected project cannot be scheduled before its deadline due to resource constraints or it has a negative profit, the project is removed from the portfolio.

The MAEA and benchmarking algorithms were implemented in C language on a PC with a CPU at 2.0 GHz and 2 GB physical memory. The average computation times are shown in Table 3. It is obvious that the profit decreasing rate has a significant role in determining the computation time of the multiagent evolutionary algorithm. If the profit decreases faster after the project's critical path length as in Cell 3 and Cell 4, the algorithm takes a much longer time to search for portfolios with projects complete in time.

The average profits of 24 instances achieved by the three algorithms are 7486.85, 7489.24, and 8120.70, respectively. It is observed that the MAEA has a higher average profit than the other two methods.

To investigate the performance of the proposed MAEA, the Wilcoxon Signed Ranks Test was applied to analyze the data in Table 2. Table 4 shows the paired comparison outcomes of statistical analysis. The profits obtained by the MAEA are significantly higher than the other two methods at a significance level of 0.001.

5. Conclusions

In this paper, the resource-constrained project portfolio selection and scheduling problem is formulated as a 0-1 integer programming model. The problem has a dual level structure. Project scheduling in the lower level helps increase the portfolio profit by improving the resource allocation among selected projects and rescheduling their activities. A multiagent evolutionary algorithm is proposed to solve the RCPPSSP. The algorithm adopts a dual level structure owing to the nature of the RCPPSSP. In the upper level, agents in an agent lattice are designed to search for feasible portfolios automatically. The neighborhood competition operator and self-learning operator are integrated to accelerate the evolution of agents. In the lower level, each agent adopts a priority rule-based heuristic to conduct multiproject scheduling to better utilize the scarce resources. We conducted an experiment to test the performance of the proposed algorithm. A set of 24 instances were generated from the Patterson set systematically. Computational results show that the proposed multiagent evolutionary algorithm has an outstanding performance.

Notations

Ω: Set of candidate projects
i: Project index, $i = 1, 2, \ldots, N$, where N denotes the number of candidate projects
j: Activity index, $j = 1, 2, \ldots, J_i$, where J_i is the number of activities in project i
t: Time index, $t = 0, 1, \ldots, T$, where T is the upper bound of project completion time
k: Resource index, $k = 1, 2, \ldots, K$, where K is the number of resource types
Φ: Set of selected projects, $\Phi = \{i \mid x_i = 1, i \in \Omega\}$
(i, j): Activity j in project i
d_{ij}: Duration of activity (i, j)

S_{ij}: Set of immediate successors of activity (i, j)
ST_{ij}: Start time of activity (i, j)
CT_{ij}: Completion time of activity (i, j), $CT_{ij} = ST_{ij} + d_{ij}$
EF_{ij}: Earliest finish time of activity (i, j)
LF_{ij}: Latest finish time of activity (i, j)
A_t: Set of activities in process at time t
R_k: Capacity of renewable resource k
r_{ijk}: Quantity of resource k required by activity (i, j) for its execution
b_{it}: Profit of project i when it is complete at time t
T_i: Upper bound of the completion time of project i.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

The authors are grateful to the anonymous reviewers for their constructive comments. The paper is supported by National Natural Science Foundation of China (Grant no. 71072119) and Zhejiang Provincial Natural Science Foundation of China (Grant no. R7100297).

References

[1] F. Ghasemzadeh and N. P. Archer, "Project portfolio selection through decision support," *Decision Support Systems*, vol. 29, no. 1, pp. 73–88, 2000.

[2] R. L. Schmidt, "A model for R & D project selection with combined benefit, outcome and resource interactions," *IEEE Transactions on Engineering Management*, vol. 40, no. 4, pp. 403–410, 1993.

[3] K. F. Doerner, W. J. Gutjahr, R. F. Hartl, C. Strauss, and C. Stummer, "Pareto ant colony optimization with ILP preprocessing in multiobjective project portfolio selection," *European Journal of Operational Research*, vol. 171, no. 3, pp. 830–841, 2006.

[4] A. L. Medaglia, S. B. Graves, and J. L. Ringuest, "A multiobjective evolutionary approach for linearly constrained project selection under uncertainty," *European Journal of Operational Research*, vol. 179, no. 3, pp. 869–894, 2007.

[5] Y.-Y. Shou and Y.-L. Huang, "Combinatorial auction algorithm for project portfolio selection and scheduling to maximize the net present value," *Journal of Zhejiang University C*, vol. 11, no. 7, pp. 562–574, 2010.

[6] M. A. Coffin and B. W. Taylor III, "Multiple criteria R&D project selection and scheduling using fuzzy logic," *Computers and Operations Research*, vol. 23, no. 3, pp. 207–220, 1996.

[7] J. Chen and R. G. Askin, "Project selection, scheduling and resource allocation with time dependent returns," *European Journal of Operational Research*, vol. 193, no. 1, pp. 23–34, 2009.

[8] W. J. Gutjahr, S. Katzensteiner, P. Reiter, C. Stummer, and M. Denk, "Competence-driven project portfolio selection, scheduling and staff assignment," *Central European Journal of Operations Research*, vol. 16, no. 3, pp. 281–306, 2008.

[9] T. R. Browning and A. A. Yassine, "Resource-constrained multi-project scheduling: priority rule performance revisited," *International Journal of Production Economics*, vol. 126, no. 2, pp. 212–228, 2010.

[10] N. R. Jennings and M. Wooldridge, "Applying agent technology," *Applied Artificial Intelligence*, vol. 9, no. 4, pp. 357–369, 1995.

[11] M. J. Wooldridge and N. R. Jennings, "Intelligent agents: theory and practice," *Knowledge Engineering Review*, vol. 10, no. 2, pp. 115–152, 1995.

[12] X.-F. Xie and J. Liu, "Multiagent optimization system for solving the traveling salesman problem (TSP)," *IEEE Transactions on Systems, Man, and Cybernetics B*, vol. 39, no. 2, pp. 489–502, 2009.

[13] G. Di Marzo Serugendo, M.-P. Gleizes, and A. Karageorgos, "Selforganisation and emergence in MAS: an overview," *Informatica*, vol. 30, no. 1, pp. 45–54, 2006.

[14] N. R. Jennings, K. Sycara, and M. Wooldridge, "A roadmap of agent research and development," *Autonomous Agents and Multi-Agent Systems*, vol. 1, no. 1, pp. 7–38, 1998.

[15] R. Drezewski and L. Siwik, "Co-evolutionary multi-agent system for portfolio optimization," *Studies in Computational Intelligence*, vol. 100, pp. 271–299, 2008.

[16] W. Zhong, J. Liu, M. Xue, and L. Jiao, "A multiagent genetic algorithm for global numerical optimization," *IEEE Transactions on Systems, Man, and Cybernetics B*, vol. 34, no. 2, pp. 1128–1141, 2004.

[17] R. Drezewski and L. Siwik, "A review of agent-based co-evolutionary algorithms for multi-objective optimization," *Adaptation, Learning, and Optimization*, vol. 7, pp. 177–209, 2010.

[18] J. A. Araúzo, J. Pajares, and A. Lopez-Paredes, "Simulating the dynamic scheduling of project portfolios," *Simulation Modelling Practice and Theory*, vol. 18, no. 10, pp. 1428–1441, 2010.

[19] X. Yuan, C. Xiao, X. Lv, and J. Liu, "A multi-agent genetic algorithm for resource constrained project scheduling problems," in *Proceeding of Genetic and Evolutionary Computation Conference*, pp. 195–196, 2013.

[20] I. S. Kurtulus and S. C. Narula, "Multi-project scheduling: analysis of project performance," *IIE Transactions*, vol. 17, no. 1, pp. 58–66, 1985.

[21] W.-C. Zhong, J. Liu, F. Liu, and L.-C. Jiao, "Combinatorial optimization using multi-agent evolutionary algorithm," *Chinese Journal of Computers*, vol. 27, no. 10, pp. 1341–1353, 2004.

[22] M. Srinivas and L. M. Patnaik, "Genetic algorithms: a survey," *Computer*, vol. 27, no. 6, pp. 17–26, 1994.

[23] J. Alcaraz and C. Maroto, "A robust genetic algorithm for resource allocation in project scheduling," *Annals of Operations Research*, vol. 102, no. 1–4, pp. 83–109, 2001.

[24] J. F. Gonçalves, J. J. M. Mendes, and M. G. C. Resende, "A genetic algorithm for the resource constrained multi-project scheduling problem," *European Journal of Operational Research*, vol. 189, no. 3, pp. 1171–1190, 2008.

[25] Y. Shou and W. Wang, "Robust optimization-based genetic algorithm for project scheduling with stochastic activity durations," *Information*, vol. 15, no. 10, pp. 4049–4064, 2012.

[26] E. M. Davies, "An experimental investigation of resource allocation in multiactivity projects," *Operational Research Quarterly*, vol. 24, no. 4, pp. 587–591, 1973.

[27] R. Kolisch, "Serial and parallel resource-constrained project scheduling methods revisited: theory and computation," *European Journal of Operational Research*, vol. 90, no. 2, pp. 320–333, 1996.

[28] J. H. Patterson, "Comparison of exact approaches for solving the multiple constrained resource, project scheduling problem," *Management Science*, vol. 30, no. 7, pp. 854–867, 1984.

[29] J. Chen, *Project selection, scheduling and resource allocation for engineering design groups, [PhD dissertation]*, University of Arizona, Tucson, Ariz, USA, 2005.