Hindawi Publishing Corporation
Journal of Control Science and Engineering
Volume 2015, Article ID 923791, 6 pages
http://dx.doi.org/10.1155/2015/923791

Research Article

Hybrid Particle Swarm and Differential Evolution Algorithm for Solving Multimode Resource-Constrained Project Scheduling Problem

Lieping Zhang,[1,2] Yingxiong Luo,[3] and Yu Zhang[1,2]

[1]*Guangxi Key Laboratory of New Energy and Building Energy Saving, Guilin University of Technology, Guilin 541004, China*
[2]*College of Mechanical and Control Engineering, Guilin University of Technology, Guilin 541004, China*
[3]*College of Information Science and Engineering, Guilin University of Technology, Guilin 541004, China*

Correspondence should be addressed to Lieping Zhang; zlp_gx_gl@163.com

Received 14 July 2015; Revised 11 September 2015; Accepted 14 September 2015

Academic Editor: Petko Petkov

In order to find a feasible solution for the multimode resource-constrained project scheduling problem (MRCPSP), a hybrid of particle swarm optimization (PSO) and differential evolution (DE) algorithm is proposed in this paper. The proposed algorithm uses a two-level coding structure. The upper-level structure is coded for scheduling sequence, which is optimized by PSO algorithm. The lower-level structure is coded for project execution mode, and DE algorithm is used to solve the optimal scheduling model. The effectiveness and advantages of the proposed algorithm are illustrated by using the test function of project scheduling problem library (PSPLIB) and comparing with other scheduling methods. The results show that the proposed algorithm can well solve MRCPSP.

1. Introduction

Resource-constrained project scheduling problem (RCPSP) is a project scheduling problem of how to arrange the task start-up time to minimize the total project time reasonably in the conditions of resource constraints and project timing constraints [1]. Multimode resource-constrained project scheduling problem (MRCPSP) is an extension of RCPSP, which is a kind of Nondeterministic Polynomial (NP) problem. On the basis of traditional RCPSP, MRCPSP considers a variety of optional modes of the task and the dependent relationships of resources. It can select task model and attribute dynamically. Comparing with the RCPSP, MRCPSP is closer to the real problem and is of more theoretical as well as practical significance [2]. The optimization solution methods of MRCPSP can be divided into precise algorithm, heuristic algorithm, and intelligent optimization algorithm [3]. Precise algorithm adopts the branch-and-bound method as the main solving method, and it can get the optimal solution. But it is not suitable for solving large-scale scheduling problem. Heuristic algorithm has a strong ability to solve large-scale scheduling problem and the characteristic of the computing speed. But it cannot guarantee obtaining the optimal solution. Intelligent optimization algorithm can get the optimization solution or suboptimal solution by using the algorithm optimization mechanism and evaluation mechanism in a limited set of feasible solutions. Intelligent optimization algorithm is an effective solving method for MRCPSP. Jarboui et al. proposed a combinatorial particle swarm optimization (PSO) algorithm to solve MRCPSP [4]. Van Peteghem and Vanhoucke proposed a scatter search algorithm for MRCPSP, which is executed with different improvement methods [5]. Tseng and Chen presented a two-phase genetic local search algorithm that combines the genetic algorithm and the local search method to solve MRCPSP [6]. Damak et al. proposed a differential evolution (DE) algorithm to solve MRCPSP with multiple execution modes for each activity and minimization of the makespan [7]. Wang and Fang proposed an estimation of distribution algorithm to solve MRCPSP [8]. Li and Zhang presented an ant colony optimization-based methodology for solving the MRCPSP considering both renewable and nonrenewable resources [9]. Liu et al. presented a memetic

algorithm to solve MRCPSP, in which a new fitness function and two very effective local search procedures are used in the proposed algorithm [10].

The methods mentioned above have their different merits and shortcomings. But if we take the combination of several methods into account properly, a better balance between quality and efficiency of solving may be achieved. In this case, we can get the optimum solution and obtain the ultimate solution for different actual demands. Liu et al. proposed a novel hybrid algorithm named PSO-DE, which integrates PSO with DE algorithm to solve constrained numerical and engineering optimization problems [11]. Zhang and Kang proposed a hybrid of ant colony and particle swarm optimization algorithms for solving MRCPSP, which can well solve MRCPSP [12]. Chen and Sandnes proposed a two-PSO algorithm to solve MRCPSP, in which constriction PSO is proposed for the activity priority determination while discrete PSO is employed for mode assignment [13].

A hybrid of particle swarm and differential evolution algorithm is proposed for solving MRCPSP. In that algorithm, the scheduling orders are determined by the upper-level algorithm with PSO algorithm, and the task execution modes are determined by the lower-level algorithm with DE algorithm, which can get faster convergence of the algorithm and avoids falling into local optimum. Based on the verification of many MRCPSP's instances in PSPLIB, the simulation results show that the proposed algorithm can effectively solve the MRCPSP.

2. Descriptions of MRCPSP

The MRCPSP can be described as follows. A project has a series of tasks, and there is a certain logic sequence between them as the technical process and many other reasons. Meanwhile, among several patterns, individual task can choose one of them to complete, and each pattern corresponds to a set of known time limits for project and resource requirements. Solution of the problem is to generate a scheduling scheme that enables one or some of the goals to achieve the optimization, which can meet the work of the precedence constraints relations and resource constraints condition.

In this paper, what we study is to minimize the time limit for a project in MRCPSP. The upper bound of the time limit for a project is defined as $\overline{D}$. The task j ($j = 1, 2, \ldots, J$) must select one from execution models M_j. Furthermore, in the process of implementation, the execution model cannot be interrupted or changed. In m ($1 \leqslant m \leqslant M_j$) kinds of mode to perform the work j requires k kinds of renewable resources, denoted by r^{ρ}_{jmk}, and n kinds of nonrenewable resources, denoted by r^{ν}_{jmn}, and the execution time is denoted by d_{jm}. There is only one execution mode for virtual task 1 and J, which does not consume resources, and the time limit for the project is zero. Among them, renewable resources are denoted by ρ and nonrenewable resources are denoted by ν. In the whole time of the project and each stage of the project, the kth kind of renewable resources can amount to a constant, denoted by R^{ρ}_{k} ($k = 1, 2, \ldots, K$), and the nth kind of nonrenewable total resources, denoted by R^{ν}_{n} ($n = 1, 2, \ldots, N$). If the task J selects the mth kind of modes to execute and complete in the stage of t, then $\chi_{jmt} = 1$; otherwise, $\chi_{jmt} = 0$, wherein χ_{jmt} is a decision variable. The mathematical model for MRCPSP can be described as follows [14]:

$$\min \quad \sum_{t=EF_j}^{LF_j} t \cdot x_{jmt} \tag{1}$$

$$\text{s.t.} \quad \sum_{m=1}^{M_j} \sum_{t=EF_j}^{LF_j} x_{jmt} = 1, \quad j = 1, 2, \ldots, J \tag{2}$$

$$\sum_{m=1}^{M_i} \sum_{t=EF_i}^{LF_i} t \cdot x_{imt} \leq \sum_{m=1}^{M_j} \sum_{t=EF_j}^{LF_j} t \cdot x_{jmt} - d_{jm}, \quad j = 2, \ldots, J, \ i \in P_j \tag{3}$$

$$\sum_{j=1}^{J} \sum_{m=1}^{M_j} r^{\rho}_{jmk} \sum_{q=\max\{t, EF_j\}}^{\min\{t+d_m-1, LF_j\}} x_{jmq} \leq R^{\rho}_{k}, \quad k = 1, \ldots, K, \ t = 1, \ldots, \overline{D} \tag{4}$$

$$\sum_{j=1}^{J} \sum_{m=1}^{M_j} r^{\nu}_{jmn} \sum_{t=EF_j}^{LF_j} x_{jmt} \leq R^{\nu}_{n}, \quad n = 1, \ldots, N. \tag{5}$$

Among them, $x_{jmt} \in \{0, 1\}$, and $j = 1, 2, \ldots, J$, and $m = 1, 2 \ldots, M_j$, and $t = EF_j, \ldots, LF_j$, EF_j, LF_j, respectively, corresponds to the earliest completion time and finish time at the latest of the task j, and P_j is the direct predecessor task set of j. Equation (1) is the objective function which means the shortest time limit for the total project. Equation (2) means that a task can only be completed once in an execution mode. Equation (3) represents the precedence constraints. According to the constructed network model and network planning methods, the end time of each task is equal to the sum of its start time and the time limit for a task, and the tight task must be carried out at the end of the work. Equation (4) ensures the amount of renewable resources in every stage will not be larger than the amount available. Equation (5) ensures that the amount of nonrenewable resources consumed in the entire project will not be larger than the amount available.

3. Algorithm Design

The upper-level individual coding of the designed algorithm controls the task scheduling orders, and the lower-level individual coding manages the task execution modes. After determining the task scheduling orders, it can find the optimal execution mode in the scheduling order. Due to the fact that the range of execution mode's values is just an integer in the range [1, 3], it is a very small range and it will make the PSO algorithm easily fall into local optimum. Hence, the DE algorithm was introduced into the proposed algorithm to solve the optimal execution mode, which has stronger global

convergence ability and robustness, and then optimize the task scheduling sequence with the PSO algorithm.

3.1. Coding Design. The upper-level individual of algorithm uses the particle coding based on priority rules. The numbers of tasks are represented by the search space dimension of particle swarm, a total of J. At the same time, all dimension values x_{ij} of particle x_i are random real numbers in the range $[0, 1]$. The value represents the priority order of particle j kinds of dimension. The higher the value, the greater the priority. While the virtual start task of the project's first task is the first task to be executed, the last task is the virtual end task of the project, which is the final project task. All the priority values that can set virtual start task are $x_{ij} = 1$, the priority values of the virtual end task are $x_{ij} = 0$, and the other priority values of task are in the range $(0, 1)$. Moreover, the task priority values are different.

In order to let the priority rules satisfy the task's precedence constraints, we need to adjust the task priority value. Specific adjustment method is as follows. Compare the priority values of current task and preceding task; the value will not be adjusted if it is lower than the latter. Otherwise, exchange both of the priority values. And then, consider the preceding task as the current task and compare it with its preceding task. As long as they are exchanged, repeat this step until no values need to be exchanged or there is no preceding task. After this adjustment, we can get the priority rules that met the requirements of the task precedence constraints. The low level of algorithm controls the execution mode coding. Assuming that the execution modes of the task j are M_j kinds, the execution mode of J is in the range $[1, M_j]$; note that the value of the execution mode is integer.

3.2. A Hybrid of PSO and DE Algorithm

3.2.1. PSO Algorithm Based on Inertia Weight. In an n-dimensional search space, population $X = \{x_1, \ldots, x_2, \ldots, x_m\}$ is composed of m particles, and the particle position is $x_i = (x_{i1}, x_{i2}, \ldots, x_{im})^T$, and the particle velocity is $V_i = (v_{i1}, v_{i2}, \ldots, v_{in})^T$. The corresponding individual extreme value is $P_i = (p_{i1}, p_{i2}, \ldots, p_{im})^T$, and the global extreme value of the entire population is $P_g = (p_{g1}, p_{g2}, \ldots, p_{gm})^T$. Taking into account the premature convergence and poor global convergence to the basic PSO algorithm in the practical application, we adopted the PSO algorithm based on inertia weight to promote the global searching ability and the local search ability [15]. According to the principle of following the current optimal particle, x_i particle, which introduced the inertia weight w, will change its speed and position according to

$$v_{id}^{(t+1)} = w v_{id}^{(t)} + c_1 r_1 \left(p_{id}^{(t)} - x_{id}^{(t)}\right) + c_2 r_2 \left(p_{gd}^{(t)} - x_{id}^{(t)}\right) \quad (6)$$

$$x_{id}^{(t+1)} = x_{id}^{(t)} + v_{id}^{(t+1)}, \quad (7)$$

where $d = 1, 2, \ldots, n$, which means the dth dimension of the particle, and $i = 1, 2, \ldots, m$, which means the particle i. m is the population size. t means the current evolution times. r_1 and r_2 are random numbers distributed in the range $[0, 1]$. c_1 and c_2, respectively, correspond to accelerated constant of the individual particle and the group particle. $v_{id}^{(t+1)}$, $v_{id}^{(t)}$, $x_{id}^{(t+1)}$, and $x_{id}^{(t)}$, respectively, represent the velocity and the position of the ith particle in the $(t + 1)$th generation and the tth generation.

The value of inertia weight w can affect the algorithm's global search ability and local search ability. Related research results show that when w decline linearly from 0.9 to 0.4, the algorithm can converge quickly to the optimal solution [16]. Therefore, this paper adopts the dynamic inertia weight, and the value of w changes in line in the searching process of the particle swarm algorithm, as shown in

$$w = 0.9 - \left(\frac{\text{nc}}{\text{ncmax}}\right) * 0.5, \quad (8)$$

where nc is the current number of cycles and ncmax is the maximum number of cycles.

3.2.2. DE Algorithm. DE algorithm is a simple and efficient parallel search algorithm, which has better robustness and faster convergence capability than other evolution algorithms. And it can be used to solve the MRCPSP. The low-level individual of the designed algorithm uses the DE algorithm to find the optimal execution mode in the scheduling sequence given.

For the individual $X_{r_1,G}$, a new individual $V_{i,G+1}$ can be obtained by

$$V_{i,G+1} = X_{r_1,G} + F \times \left(X_{r_2,G} - X_{r_3,G}\right), \quad (9)$$

where $i = 1, 2, \ldots, N$. r_1 and r_2 as well as r_3 are different random integers in the range $[1, N]$, and they are different from the subscript index i. The variation factor F is a real constant in the range $[0, 2]$, the main function of which is to control the degree of amplification of differential vector $(X_{r_2,G} - X_{r_3,G})$.

The interlace operation is as shown below:

$$U_{i,G+1} = \left(u_{1\text{i},G+1}, u_{2i,G+1}, \ldots, u_{Di,G+1}\right). \quad (10)$$

In the equation

$$u_{ji,G+1} = \begin{cases} V_{ji,G+1}, & \text{if } \left(\text{rand}b\left(j\right) \le \text{CR}\right) \text{ or } \left(j = mbr\left(i\right)\right) \\ X_{ji,G+1}, & \text{if } \left(\text{rand}b\left(j\right) > \text{CR}\right), \left(j \ne mbr\left(i\right)\right), \end{cases} \quad (11)$$

$$j = 1, 2, \ldots, D,$$

where rand$b(j)$ is a uniform distribution probability in the range $[0, 1]$, CR is the crossover probability predefined by the user, CR $\in (0, 1)$, and $mbr(i)$ is a random number generated in the range $[1, D]$.

In order to determine whether vector $U_{i,G+1}$ can be $(G + 1)$th generation of population individuals, compare $U_{i,G+1}$ with $X_{i,G}$; if the former fitness value is better than the latter, replace $X_{i,G}$ with $U_{i,G+1}$ in $(G + 1)$th generation; otherwise, reserve $X_{i,G}$.

3.3. Scheduling Generation Strategy. Scheduling generation strategy based on priority rules includes serial scheduling scheme and parallel scheduling scheme. In this paper, we use serial scheduling scheme to solve the problem. For an upper individual x_i, we can get x_i' according to sorting the order of x_i from big to small. And the serial number of original individual x_i is stored in the order of x_i' to y_i. For example, if the original individuals are $[1, 0.3, 0.5, 0.2, 0.4, 0.7, 0]$, then the sorted individuals are $[1, 0.7, 0.5, 0.4, 0.3, 0.2, 0]$, and the value of y_i is $[1, 6, 3, 5, 2, 4, 7]$. Select an element y_{ij} in order from y_i according to the value of y_{ij} to determine the start time of scheduling task x_i and y_{ij}. Choose a larger one from the current time and all the preceding task finish time of the current task as the current task start time, and the end time of the current task is determined by the lower layer determined time limit for project plus the current task start time. Update the current time in accordance with the use of resources. When the available resources run out, update the current time. And when the end time of a task is greater than the current time, release its consumed resources in turn. Meanwhile, make the current time equal to the end time of the task that releases resources, until available resources meet the requirements of the current task. The end time of virtual end of task is the project completion time, namely, the value of algorithm fitness function.

3.4. Algorithm Implementation Processes. The major steps of the proposed algorithm can be outlined as follows.

Step 1. Initialize the upper-level populations to generate m individuals position (priority rules) $(x_1, x_2, \ldots, x_m)$ and velocity $(v_1, v_2, \ldots, v_m)$.

Step 2. Adjust the priority rules to meet the task precedence constraints represented in (3).

Step 3. Initialize the lower-level populations to generate n individuals (the task execution modes).

Step 4. The lower-level populations, respectively, execute mutation and interlace operation according to (9) and (10).

Step 5. The lower-level populations execute selection operation and optimize the execution mode.

Step 6. If the lower-level populations reached the maximum numbers of iterations, then go to Step 7, or go to Step 4 to continue to optimize the execution mode.

Step 7. Respectively, update the velocity and position of the upper populations according to (6) and (7).

Step 8. If the upper-level populations reached the maximum numbers of iterations, the algorithm ends, or go to Step 2.

4. Simulation Analyses

In order to test the effectiveness of the algorithm, the standard MRCPSP instances of PSLIB are used to validate the effectiveness of the algorithm. The selected task is J18.

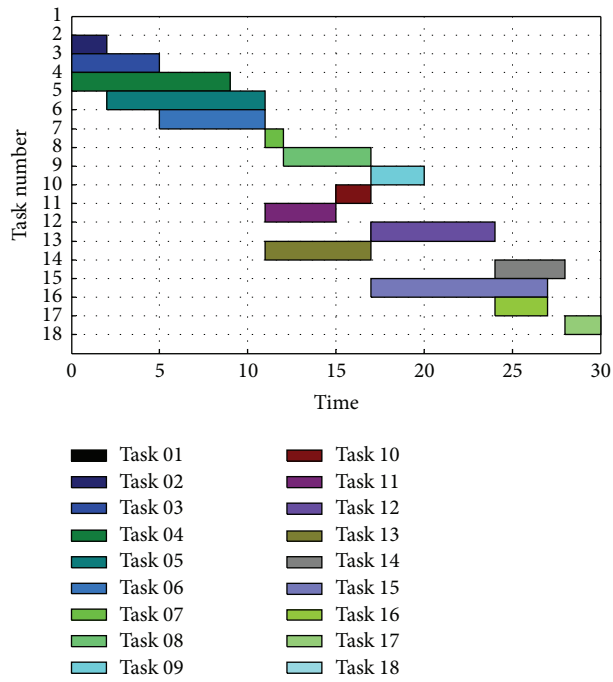

Figure 1: Gantt chart of the optimal solution.

Table 1: Computing results vary from different number of iterations in the upper level.

Upper-level number of iterations	Average deviation (%)	Optimal solution ratio (%)
10	0.78	83.33
15	0.44	93.33

The task contains 2 kinds of renewable resources and 2 kinds of nonrenewable resources, in which the renewable resources are $R_1 = 26$, $R_2 = 27$, and the nonrenewable resources are $N_1 = 48$, $N_2 = 51$. We compare the proposed algorithm with different iterations. The upper-level PSO algorithm parameters of the proposed algorithm are described as follows: the individual particle acceleration constant is selected as $c_1 = 2$ and the group of particle acceleration constants is selected as $c_2 = 2$, the number of populations is selected as 10, and the number of iterations is taken as 10. The parameters of lower-level DE algorithm are described as follows [17]: the interlace probability is selected as CR = 0.4, the variation factor is selected as $F = 0.7$, and both of the numbers of populations and iterations are selected as 10. A Gantt chart of optimal solution is shown in Figure 1.

When the upper-level iteration numbers are, respectively, 10 and 15, the solved results of the proposed algorithm are shown in Table 1. In accordance with Table 1, we can know that the proposed algorithm is effective for MRCPSP, and, with the increase of the number of iterations, the probability of obtaining the optimal solution increased.

In order to test the feasibility of the proposed algorithm, we compare the proposed algorithm with two layers' PSO algorithm based on inertia weight. The parameters of two

Table 2: Results comparison of PSODE algorithm and PSO algorithm.

Algorithm	Average deviation (%)	Optimum ratio (%)
PSODE	0.78	83.33
PSO	5.67	40

Table 3: Results comparison of the proposed algorithm and other methods.

Project	Population size	J10		J20	
		Average deviation	Optimal solution rate	Average deviation	Optimal solution rate
This paper	10-10	0%	100%	1.82%	76.67%
Reference [12]	10-10	0%	100%	1.28%	71.5%

layers' PSO algorithm based on inertia weight are described as follows: the individual particle acceleration constants c_1 and c_2 are selected as 2, the numbers of populations are selected as 10, and the numbers of iterations are selected as 10. The parameters of the proposed algorithm are described as before, in which the number of iterations of PSO algorithm is selected as 10. The corresponding results are shown in Table 2. In accordance with Table 2, we can know that the average deviation and optimum ratio of PSODE algorithm are higher than those of PSO algorithm.

In order to further test the feasibility and the effectiveness of the proposed algorithm, we compare it with the solving results of [12]. Projects J10 and J20 have 536 and 554 optimal solutions, respectively, and projects J10 and J20 have two kinds of renewable resources and two kinds of nonrenewable resources, respectively, and every instance has three modes. Project J10 has a total of 12 tasks, and project J20 has a total of 22 tasks, and both the first task and the last task are virtual tasks, which do not consume the time and the resources. The parameters of the proposed algorithm are described as before. The results comparison of the proposed algorithm and the method of [12] is shown in Table 3. According to the results in Table 3, the proposed algorithm is better than the method of [12] in average deviation and optimal solution rate.

5. Conclusions

A new algorithm for MRCPSP combined PSO algorithm and DE algorithm is proposed in this paper. The optimal project execution modes are obtained by DE algorithm after the determination of the task scheduling sequence; then the task scheduling sequences are optimized by PSO algorithm. In order to confirm the validity of the algorithm, we take an experiment based on the standard MRCPSP instances of PSLIB and other scheduling methods. The experimental and comparison results show that the proposed algorithm has a better performance in average deviation and optimal solution rate.

Conflict of Interests

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

Acknowledgments

This research was supported by Guangxi Natural Science Foundation (no. 2014GXNSFAA118371) and the research fund of Guangxi Key Laboratory of New Energy and Building Energy Saving (no. 12-03-21-3).

References

[1] F. Ballestín and R. Blanco, "Theoretical and practical fundamentals for multi-objective optimisation in resource-constrained project scheduling problems," *Computers and Operations Research*, vol. 38, no. 1, pp. 51–62, 2011.

[2] P. Ghoddousi, E. Eshtehardian, S. Jooybanpour, and A. Javanmardi, "Multi-mode resource-constrained discrete time-cost-resource optimization in project scheduling using non-dominated sorting genetic algorithm," *Automation in Construction*, vol. 30, pp. 216–227, 2013.

[3] P. Brucker, A. Drexl, R. Möhring, K. Neumann, and E. Pesch, "Resource-constrained project scheduling: notation, classification, models, and methods," *European Journal of Operational Research*, vol. 112, no. 1, pp. 3–41, 1999.

[4] B. Jarboui, N. Damak, P. Siarry, and A. Rebai, "A combinatorial particle swarm optimization for solving multi-mode resource-constrained project scheduling problems," *Applied Mathematics and Computation*, vol. 195, no. 1, pp. 299–308, 2008.

[5] V. Van Peteghem and M. Vanhoucke, "Using resource scarceness characteristics to solve the multi-mode resource-constrained project scheduling problem," *Journal of Heuristics*, vol. 17, no. 6, pp. 705–728, 2011.

[6] L.-Y. Tseng and S.-C. Chen, "Two-phase genetic local search algorithm for the multimode resource-constrained project scheduling problem," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 4, pp. 848–857, 2009.

[7] N. Damak, B. Jarboui, P. Siarry, and T. Loukil, "Differential evolution for solving multi-mode resource-constrained project scheduling problems," *Computers and Operations Research*, vol. 36, no. 9, pp. 2653–2659, 2009.

[8] L. Wang and C. Fang, "An effective estimation of distribution algorithm for the multi-mode resource-constrained project scheduling problem," *Computers & Operations Research*, vol. 39, no. 2, pp. 449–460, 2012.

[9] H. Li and H. Zhang, "Ant colony optimization-based multi-mode scheduling under renewable and nonrenewable resource constraints," *Automation in Construction*, vol. 35, pp. 431–438, 2013.

[10] S. Liu, D. Chen, and Y. Wang, "Memetic algorithm for multi-mode resource-constrained project scheduling problems," *Journal of Systems Engineering and Electronics*, vol. 25, no. 4, pp. 609–617, 2014.

[11] H. Liu, Z. Cai, and Y. Wang, "Hybridizing particle swarm optimization with differential evolution for constrained numerical and engineering optimization," *Applied Soft Computing Journal*, vol. 10, no. 2, pp. 629–640, 2010.

[12] W. Zhang and K. Kang, "Ant colony and particle swarm optimization algorithm-based solution to multi-mode resource-constrained project scheduling problem," *Computer Engineering and Applications*, vol. 43, no. 34, pp. 213–216, 2007.

[13] R.-M. Chen and F. E. Sandnes, "An efficient particle swarm optimizer with application to man-day project scheduling problems," *Mathematical Problems in Engineering*, vol. 2014, Article ID 519414, 9 pages, 2014.

[14] R. Kolisch and A. Sprecher, "PSPLIB—a project scheduling problem library," *European Journal of Operational Research*, vol. 96, no. 1, pp. 205–216, 1997.

[15] A. El-Gallad, M. El-Hawary, A. Sallam, and A. Kalas, "Enhancing the particle swarm optimizer via proper parameters selection," in *Proceedings of the IEEE Canadian Conference on Electrical & Computer Engineering*, pp. 792–797, Winnipeg, Canada, May 2002.

[16] Y. H. Shi and R. C. Eberhart, "Empirical study of particle swarm optimization," in *Proceedings of the Congress on Evolutionary Computation (CEC '99)*, pp. 1945–1950, Piscataway, NJ, USA, July 1999.

[17] Z. Huang and Y. Chen, "An improved differential evolution algorithm based on adaptive parameter," *Journal of Control Science and Engineering*, vol. 2013, Article ID 462706, 5 pages, 2013.