2014 IEEE Congress on Evolutionary Computation (CEC)
July 6-11, 2014, Beijing, China

A Co-evolutionary Teaching-learning-based Optimization Algorithm for Stochastic RCPSP

Huan-yu Zheng, Ling Wang and Sheng-yao Wang

***Abstract*—A co-evolutionary teaching-learning-based optimization (CTLBO) algorithm is proposed in this paper to solve the stochastic resource-constrained project scheduling problem (SRCPSP). The activity list is used for encoding, and resource-based policies are used for decoding. Also, a new competition phase is developed to select the best solution of each class as the teacher. To make two classes evolve cooperatively, both the teacher phase and student phase of the TLBO are modified. Moreover, Taguchi method of design of experiments is used to investigate the effect of parameter setting. Computational results are provided based on the well-known PSPLIB with certain probability distributions. The comparisons between the CTLBO and some state-of-the-art algorithms are provided. It shows that the CTLBO is more effective in solving the problems with medium to large variance.**

I. Introduction

THE resource-constrained project scheduling problem (RCPSP) considers scheduling a set of project activities over time and resource. It needs to determine both the start time and the finish time of every activity of the project as well as the allocation of limited resources for activities in order to optimize certain objectives, such as project makespan. During the past few decades, the RCPSP and its extensions have drawn increasing attention by the researchers in many fields due to their importance in both academic research and engineering applications [1]–[6].

Usually, it assumes in the literature that the durations and starting times of activities are deterministic values. In real world, however, those values cannot be estimated precisely by experts. Consequently, it is more appropriate to consider the uncertain nature of the problem. Several fundamental approaches are introduced by Herroelen and Leus [7]. The stochastic RCPSP (SRCPSP), in which activity durations are regarded as random variables, is much closer to the real conditions than the classical one. Although the SRCPSP is attracting more and more attention by researchers, relevant research work is still very limited. During recent years, only a few exact, heuristic and meta-heuristic methods have been presented. Relevant results in this area include the following. Stork [8] adopted a branch-and-bound algorithm employing three kinds of policies for solving the problem. Ballestín [9] used a stochastic serial schedule generation scheme and presented a genetic algorithm (GA) to solve the SRCPSP. The numerical results showed that the number of scenarios for estimating expected fitness should be reduced so as to generate more schedules. Ballestín and Leus [10] developed a greedy randomized adaptive search procedure (GRASP) to solve the problem. An investigation on the distribution of the possible makespan for a given set of scheduling policies was also provided. Ashtiani, et al. [11] proposed a new set of policies named pre-processor policies for the SRCPSP, which performed better than other algorithms for the instances with large variances. Zheng, et al. [12] presented an ordinal chemical reaction optimization (OCRO) for the SRCPSP employing optimal computing budget allocation to evaluate solutions in uncertain environment efficiently.

The difficulties in solving the SRCPSP mainly lie in the following aspects. With experimental results, Ballestín [9] showed that a huge gap exists between the deterministic makespan and the expected makespan. It was pointed out by Ballestín and Leus [10] that during the execution of a project, a deterministic solution does not include enough information to generate a suitable schedule. Thus, dynamic scheduling policies should be designed to tackle the stochastic factor. In addition, the RCPSP has been proved NP-hard by Blazewicz, et al. [13]. As the SRCPSP is a generalization of the RCPSP, it is NP-hard.

Population-based meta-heuristic algorithms, including GA, particle swarm optimization (PSO), differential evolution, ant colony optimization, and artificial immune system, etc., are widely applied to solve a variety of optimization problems [14]. However, sometimes such algorithms still cannot well solve the complicated problems. One possible reason is that most algorithms are based on a single population, without considering the cooperation or competition behavior of multiple swarms [15]. Recently, co-evolution methods are introduced to some population-based algorithms to achieve better performances [16]–[19].

Co-evolution is concerned with two or more populations evolving simultaneously. In general, co-evolution can be divided into two types, namely cooperative co-evolution and competitive co-evolution. Van den Bergh and Engelbrecht [20] presented a cooperative PSO to enhance search capability by developing cooperative behaviors. Multiple

H. Zheng is with Tsinghua National Laboratory for Information Science and Technology (TNList), Department of Automation, Tsinghua University, Beijing, China (e-mail: zhenghy12@mails.tsinghua.edu.cn).

L. Wang is with Tsinghua National Laboratory for Information Science and Technology (TNList), Department of Automation, Tsinghua University, Beijing, China (corresponding author to provide phone: +86-10-62783125; fax: +86-10-62786911; e-mail: wangling@mail.tsinghua.edu.cn).

S. Wang is with Tsinghua National Laboratory for Information Science and Technology (TNList), Department of Automation, Tsinghua University, Beijing, China (e-mail: wangshengyao10@mails.tsinghua.edu.cn).

This research work is partially supported by the National Key Basic Research and Development Program of China (No. 2013CB329503), the National Science Foundation of China (No. 61174189), and the Doctoral Program Foundation of Institutions of Higher Education of China (No. 20130002110057).

978-1-4799-1488-3/14/$31.00 ©2014 IEEE

swarms were used to optimize different components of the solution vector cooperatively. With such an idea, a large solution space could be decomposed into several small ones. Thus, the convergence rate could get much faster. Competition co-evolution is different from cooperative co-evolution, where several species compete with each other. In [15], a competitive co-evolutionary quantum GA was presented to solve the stochastic job shop scheduling problem.

As a newly developed meta-heuristic, teaching-learning-based optimization (TLBO) algorithm [21] is inspired by the teaching-learning process. In the TLBO, the most brilliant student of the class is selected as the teacher, who provides students with lessons to improve their grades in the teacher phase. Thereafter, students improve grades each other through learning in the student phase. The best grade amongst the population is the output as the final result. The TLBO has been shown to be competitive to other algorithms with the advantage of introducing fewer parameters [22]-[25]. To the best of our knowledge, there is no reported research work to adopt the TLBO for solving the SRCPSP. Thus, a co-evolutionary TLBO-based algorithm (CTLBO) will be designed in this paper to solve the SRCPSP with makespan minimization.

Since the original TLBO [21] is designed for solving the continuous optimization problems, it cannot be applied to combinational optimization problems directly like the SRCPSP. In other words, the TLBO should be modified, by considering the characteristics of the SRCPSP. Specifically, in this paper, the activity list is used for encoding, and resource-based policies are used for decoding. Considering that competition and cooperation of students in different classes can promote the development of students' overall performance, we make some modifications to the framework of the original TLBO. To be specific, a competition phase based on the redefinition of individual performance is embedded in the TLBO; and two classes co-evolve cooperatively in both the teacher phase and the student phase. The effect of parameter setting is investigated, and the testing results and comparisons are provided to demonstrate the effectiveness of the CTLBO in solving the SRCPSP.

The remaining paper is organized as follows: The SRCPSP is described in Section II. Section III introduces the basic TLBO. In Section IV, the CTLBO algorithm for the SRCPSP is designed in details. Computational results and comparisons are provided in Section V. Finally, the paper is ended in Section VI with some conclusions.

II. Definitions and Problem Statement

A. The RCPSP

The RCPSP considers a set of activities with a deterministic duration for each activity. An acyclic graph $G(\boldsymbol{N}, \boldsymbol{A})$ is adopted to show the precedence relationship, in which $\boldsymbol{N} = \{0,1,...,n\}$ denotes project activities and $\boldsymbol{A}$ denotes a set of arcs $(i, j), i, j \in \boldsymbol{N}$ between activities. For an activity $j \in \boldsymbol{N}$, its duration is denoted as d_j.

Suppose there are K resources during the execution process of a project. Let $\boldsymbol{K} = \{1,2,...,K\}$ denote the resource set. For resource $k \in \boldsymbol{K}$, activity j requires r_{jk} renewable resources during its processing process. For each resource k, there are R_k available units throughout the execution process. No interruption is allowed when activities are being processed, and there exist finish-start precedence constraints between some pairs of activities. That is, only after some activities are finished, can their subsequent activities be processed. Let activity $j=0$ and $j=n$ be dummy activities with zero duration and resource usage, which represent the beginning and the finish of the project, respectively. A solution of the RCPSP is named as a schedule, which adopts a vector of starting times $\boldsymbol{s} = (s_0, s_1, ..., s_n)$ to arrange a beginning time s_j for each activity j. The RCPSP can be formulated with the following precedence constraints and resource constraints.

$$\min s_n \tag{1}$$

s.t.

$$s_i + d_i \le s_j, i, j \in \boldsymbol{A} \tag{2}$$

$$\sum_{j \in \boldsymbol{P}(t)} r_{jk} \le R_k, \forall t \in \mathbf{N}^+, \forall k \in K \tag{3}$$

where $\boldsymbol{P}(t)$ denotes the set of processing activities.

For the RCPSP, many heuristic decoding schemes have been presented [1-2], where the schedule generation scheme (SGS) is widely used to transform a certain encoding representation into a schedule. In general, there are two procedures named serial SGS and parallel SGS, which repeatedly assign activity starting times in different procedures [26].

B. The Stochastic RCPSP

For the SRCPSP, the duration D_j of activity $j \in \boldsymbol{N}$ is a random variable with a certain distribution known beforehand. A vector $\boldsymbol{d} = (d_0, d_1, ..., d_n)$, called a scenario, denotes a possible realization of the random variable vector $\boldsymbol{D} = (D_0, D_1, ..., D_n)$. A solution of the SRCPSP is called a scheduling policy, which is the counterpart of a schedule to the traditional RCPSP.

During the scheduling process of policies, knowledge about the scheduled activities can be used to arrange the unscheduled activities. In [27], some main sets of scheduling policies were introduced, including earliest-start policies $\mathcal{C}^{ES}$, pre-selective policies, linear pre-selective policies, job-based priority policies, also called activity-based policies $\mathcal{C}^{AB}$, and resource-based policies $\mathcal{C}^{RB}$. Among them, activity-based policies and resource-based policies, which do not dominate each other [27], are the most widely used as they dominate most other policies. Resource-based policies

consider resource constraints at any decision time, allowing unscheduled activities to start as many as possible.

For the problems with large variances, resource-based policies often perform better than activity-based policies [11]. Thus, C^{RB} is employed in this paper to determine the starting times for activities.

III. Basic Teaching-learning-based Optimization

The basic teaching-learning-based optimization [21] is a population-based algorithm originally developed for solving the continuous optimization problems. Inspired by the teaching-learning process, two phases including teacher phase and student phase are used to perform evolution of a population. In the TLBO, an initial population is formed by a group of P_n individuals called students, where each student is represented by a vector.

First, the best student of the population is selected as the teacher *T*, and each of the rest P_n -1 students S_{old} learns from *T* to generate a new student S_{new} as Eq. (4). The new student with better quality will replace the old one.

$$S_{new} = S_{old} + r \cdot (T - T_F \cdot \overline{S}_{old}) \tag{4}$$

where *r* denotes a random real value between 0 and 1, $\overline{S}_{old}$ is the average of all students before teaching, and T_F is a teaching factor as $T_F = round[1 + rand(0,1)]$.

Then, in the student phase, students learn each other by interaction. To be specific, each time two students are selected randomly, where the better one is S_{better} and the worse one is S_{worse}. Using Eq. (5), a new student is generated, which will replace S_{worse} if it is better than S_{worse}.

$$S_{new} = S_{worse} + r \cdot (S_{better} - S_{worse}) \tag{5}$$

The procedure of the basic TLBO is summarized as follows. It repeats the teaching and learning processes, until a stopping criterion is satisfied.

Algorithm 1 The basic TLBO.

```
Generate an initial population;
Evaluate population and identify the best student as teacher;
While the termination criterion is not met;
  For i =1 to P_n
    Student i learn from the teacher;
  End for
  For i =1 to P_n
    Randomly select two students;
    Students learn each other by interaction;
  End for
End while
```

IV. CTLBO for SRCPSP

A. The Framework of the CTLBO

To solve the SRCPSP effectively, a co-evolutionary TLBO is presented with the flowchart straightforwardly shown in Fig. 1. It can be seen that the procedure of the CTLBO mainly includes four phases: initialization, competition phase, teacher phase, and student phase.

First, two classes of students P_A and P_B are generated and evaluated, both with the size of P_n. Then, it sequentially repeats the competition phase, the teacher phase and the student phase until a stopping criterion is satisfied.

In each generation, competition between students of P_A and P_B is performed to calculate competitive ability of each student. The student with the highest competition ability of each class is selected as teachers T_A and T_B. Then, the two-point crossover operator is used to perform cooperation between individuals of two classes to generate new students. By competition and cooperation, the CTLBO could balance exploitation and exploration to achieve better performances for solving the SRCPSP.

Next, we introduce the main elements of the CTLBO in details.

B. Representation and Evaluation

It was concluded by Hartmann and Kolisch [1] that activity list often outperforms other kinds of representations for the RCPSP. Since the SRCPSP shares most characteristics of the traditional one, the activity list is employed to represent student for the SRCPSP. To be specific, a student is represented by an activity list $\pi = (a_0, a_1, ..., a_n)$, in which a_j is the activity on the *j-th* place.

As mentioned in Section II-B, an activity list π is evaluated by using resource-based policies. We refer to [9] for details. Similar to the literature about the SRCPSP [9]-[12], a resource-based policy is approximated by the average makespan of *nscen* scenarios as follows:

$$Makespan_{ave} = \frac{1}{nscen} \sum_{j=1}^{nscen} s_n(d^{(j)}, \pi) \tag{6}$$

where $s_n(d^{(j)}, \pi)$ denotes the makespan of π in the *j-th* scenario.

Same as the existing literature, the output of algorithm S_{best} is evaluated using additional 1000 scenarios to guarantee certain accuracy.

C. Initialization

Two classes are initiated by adopting the regret based random sampling method according to the latest finish time (LFT) rule [21]. First, the dummy activity 0 is selected at the beginning. Then, activities are randomly selected from the feasible sets each time. The feasible set is constituted with the unselected activities whose predecessors are selected. Let η_j denote the probability of choosing activity *j* with a priority value μ_j from the feasible set *SF*.

$$\mu_j = \max_{i \in SF} LFT_i - LFT_j \tag{7}$$

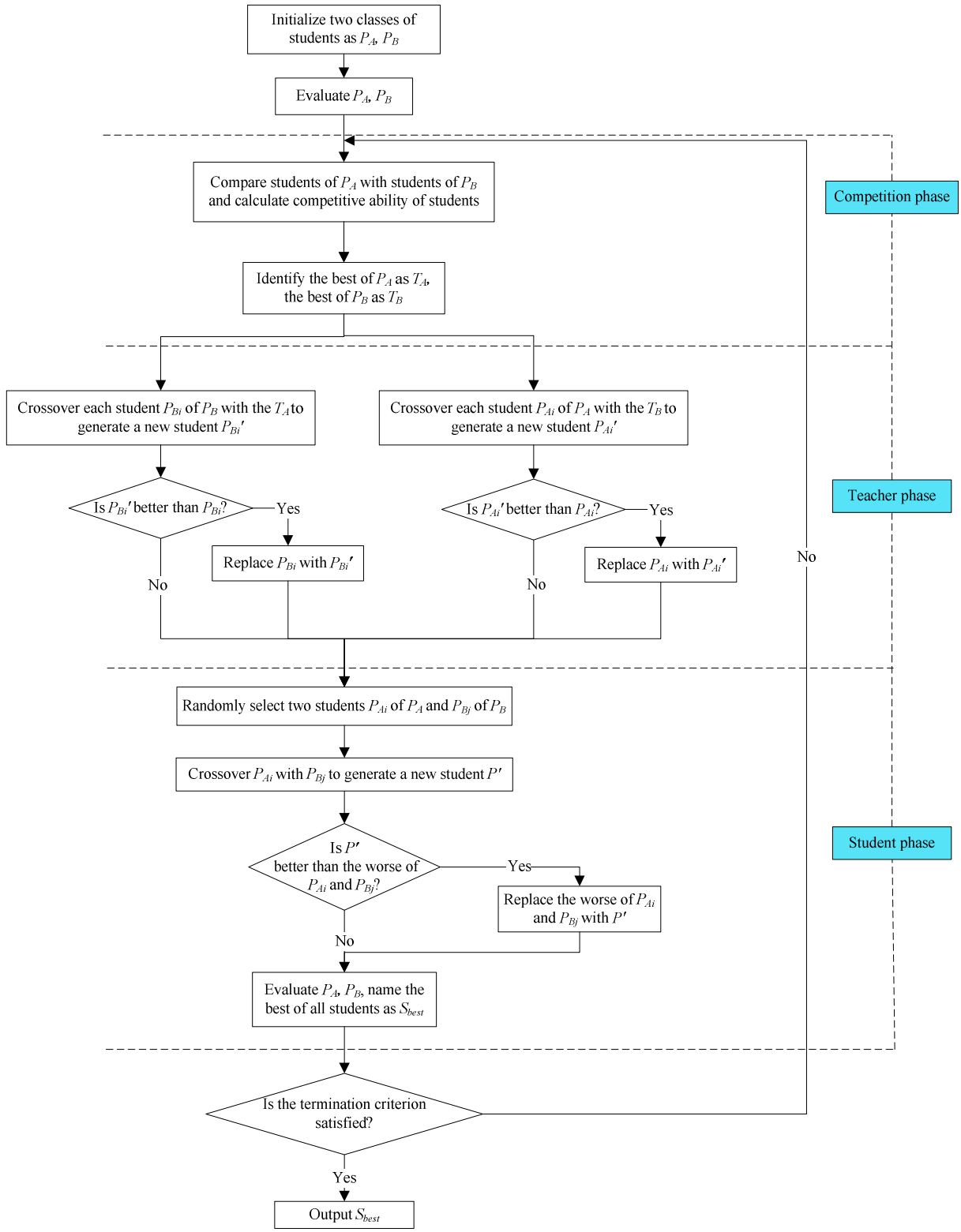

Fig. 1. The flowchart of CTLBO for SRCPSP.

$$\eta_j = (\mu_j + \varepsilon)^{\alpha} / \sum_{i \in SF} (\mu_i + \varepsilon)^{\alpha} \tag{8}$$

where LFT_j denotes the latest finish time of activity j.

Besides, we set $\varepsilon = \alpha = 1$ in the CTLBO as suggested by Kolisch [26].

D. Competition Phase

The competition phase utilizes the mutual effect of competitive students to promote the evolution of the population. The competitive ability is defined as follows.

Definition 1. *Competitive ability*: an evaluation index to

show the ability of students, which is determined by comparing with other students. Students with higher competitive values have better grades than others.

Definition 1 indicates that the competitive ability is calculated through competition with other students. Contrary to the traditional evaluation method, Angeline and Pollack [28] defined a relative fitness, which changes as populations evolve. Similar to the relative fitness, the competitive ability of student j denoted as $cAbility_j$ is calculated as follows:

$$cAbility_j = \sum_{i \in \boldsymbol{CS}} Ability_{ji} \tag{9}$$

$$Ability_{ji} = \begin{cases} 1/N_i & if\ \ j\ defeat\ i \\ 0 & else \end{cases} \tag{10}$$

where N_i is the number of students defeating student i, and $\boldsymbol{CS}$ denotes the competition set, and $Ability_{ji}$ is the competitive ability obtained by student j when competing with student i. The competitive phase is shown in Fig. 2.

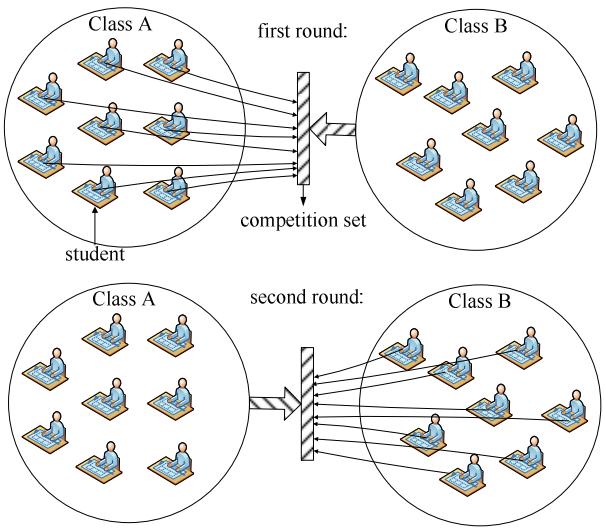

Fig. 2. Competition phase.

E. Teacher Phase and Student Phase

In both the teacher phase and the student phase of the CTLBO, two-point crossover [29] is adopted to generate a new activity list by hybridizing two old ones. With two selected activity lists, it first generates two integers q_1 and q_2 randomly in the range [1, n], and then a new activity list is generated as follows:

$$a_j^{new} := a_j^1, 1 \le j \le q_1 \tag{11}$$

$$a_j^{new} := a_k^2, k = \min\{k \mid a_k^2 \notin \{a_1^{new}, ..., a_{q_1}^{new}\}\}, q_1 + 1 \le j \le q_2 \tag{12}$$

$$a_j^{new} := a_k^1, k = \min\{k \mid a_k^1 \notin \{a_1^{new}, ..., a_{q_2}^{new}\}\}, q_2 + 1 \le j \le n \tag{13}$$

where a_j^{new}, a_j^1, a_j^2 denote the new activity list and the two old ones, respectively.

In both the teacher phase and the student phase, two classes cooperate with each other to enhance the overall performance.

In the teacher phase, different from the basic TLBO, students do not learn from the teacher of their own class, but from the teacher in the other class. In other words, a new student is generated by performing crossover between teacher T_A and student P_{Bi} or between teacher T_B and student P_{Ai}. Then the student is updated if the new one is better.

In the student phase, it selects student i from class A and student j from class B randomly, and they perform crossover to generate a new student P'. The worse student is replaced by P', if the P' is better.

F. Computational complexity analysis

In each generation of the CTLBO, it mainly includes competition phase, teacher phase, and student phase. The computational complexity of each phase can be briefly analyzed as follows:

In the competition phase, the competitive ability of students is calculated with the computational complexity $O(Psize^2)$, where $Psize$ is the size of each class in the CTLBO. In both the teacher phase and the student phase, the two-point crossover operator is performed with the computational complexity $O(n \cdot Psize)$. Once a new student is generated, it should be evaluated. For the RCPSP, Ballestín [9] pointed out that the computational complexity in using the PSGS is $O(K \cdot n^2)$. Then, the computational complexity to evaluate a population is $O(nscen \cdot Psize \cdot K \cdot n^2)$.

Thus, the computational complexity of each generation of the CTLBO is $O(Psize^2 + n \cdot Psize + nscen \cdot Psize \cdot K \cdot n^2)$, which can be simplified as $O[Psize \cdot (Psize + nscen \cdot K \cdot n^2)]$. Suppose the CTLBO evolves a total number of NG generations, the total complexity of the CTLBO is $O[NG \cdot Psize \cdot (Psize + nscen \cdot K \cdot n^2)]$, which depends on parameters NG, $Psize$, K, n, and $nscen$.

V. Computational Results and Comparisons

A. Set up of the Experiment

All tests are conducted on a PC with 2.83 GHz processor / 4.00 GB RAM in Microsoft Windows 7 system. C++ language is used to code the algorithm under Microsoft Visual Studio 2008. The problem set is j120 from PSPLIB [30], including 600 problems. Each problem contains 120 non-dummy activities.

For the SRCPSP, the adopted probability distributions, means and variances are the same as [9]-[12]. To be specific, the deterministic duration d_j^* for activity j from the j120 dataset is taken as the mean value for stochastic duration D_j. Five distributions are tested, including two uniform distributions with support $(d_j^* + \sqrt{d_j^*}, d_j^* - \sqrt{d_j^*})$ and $(0, 2d_j^*)$; one exponential distribution with expectation d_j^*; and two beta distributions with variances $d_j^*/3$ and $d_j^{*2}/3$, both with

support $(d_j^*/2, 2d_j^*)$. For convenience, the above distributions are denoted as U1, U2, Exp, B1 and B2 respectively. Clearly, U1 and B1 have small variances, U2 and B2 have medium variances, and Exp has a large variance.

The quality of an algorithm is evaluated by the average percent of deviation (APD) from the low bound of the deterministic problem, which is calculated as follows.

$$APD = \frac{1}{R}\sum_{j=1}^{R}\frac{E[s_n(D,\pi)]_j - LB_j}{LB_j}\times 100\% \qquad (14)$$

where $R = 600$ is number of problems in j120 dataset, and LB_j is the low bound of *j-th* instance of the problem set.

For a fair comparison, the termination criterion is set as a maximum of 5,000 or 25,000 schedules generated by the algorithm. Note that, it should be counted as one schedule with one scenario of a resource-based policy.

B. Parameters setting

The CTLBO itself contains two key parameters: the size of each class (*Psize*) and the number of scenarios to evaluate an activity list (*nscen*). The Taguchi method of design of experiment (DOE) [31] is used to investigate the effect of parameter setting on the CTLBO. Combinations of different values for the two parameters are listed in Table I.

Since every ten instances in the PSPLIB are generated with the same parameters, we choose the first one of each ten instances. That is, 60 instances are chosen from the j120 data set according to j120*i_1* to carry out the DOE test, $i = 1,2,...,60$. For each instance, U2 distribution with medium level of variance is chosen. A total of 25,000 schedules for each instance are used as the stopping criterion.

According to the number of parameters and the number of factor levels, we choose the orthogonal array $L_{16}(4^2)$. That is, the total number of treatments is 16. For the *j-th* parameter combination, $j = 1,2,...,16$, the CTLBO is run to obtain *APD*. Note that 60 instances are chosen for the DOE test, *R* is 60 in Eq. (14). The orthogonal array and the obtained *APD* values are shown in Table II.

The trend of each factor level is shown in Fig. 3. Also, the response value of each parameter is figured out and the significance rank of each parameter is analyzed in Table III.

From Table III, it can be seen that the number of scenarios to evaluate an activity list (*nscen*) is more significant than the size of each class (*Psize*). It is consistent to the conclusion drawn by Ballestín [9] that for meta-heuristics *nscen* is an important parameter. With the same number of schedules generated, more scenarios are used to accurately estimate the expectation of makespan obtained by each policy results in fewer policies could be used. From Fig. 3, it can be seen that *nscen* should neither be too large nor too small. A large *nscen* makes the algorithm evaluate too few policies, while a small *nscen* leads to an unreliable evaluation of policies. Although *Psize* has the less impact than *nscen*, a suitable one can still yield a good performance. According to the analysis above, we set *nscen*=20 and *Psize*=25 for the following tests.

TABLE I
COMBINATIONS OF PARAMETER VALUES

Parameters	Factor level			
	1	2	3	4
Psize	25	50	75	100
nscen	5	10	20	50

TABLE II
ORTHOGONAL ARRAY AND APD VALUES

Combination number	Factors		*APD*(%)
	Psize	*nscen*	
1	1	1	55.43
2	1	2	54.93
3	1	3	54.67
4	1	4	54.75
5	2	1	55.23
6	2	2	54.89
7	2	3	54.70
8	2	4	54.96
9	3	1	55.44
10	3	2	54.90
11	3	3	54.87
12	3	4	54.96
13	4	1	55.42
14	4	2	54.93
15	4	3	54.96
16	4	4	54.78

TABLE III
RESPONSE VALUE AND SIGNIFICANCE RANK

Level	*Psize*	*nscen*
1	54.95	55.38
2	54.95	54.91
3	55.04	54.80
4	55.02	54.86
Delta	0.10	0.58
Rank	2	1

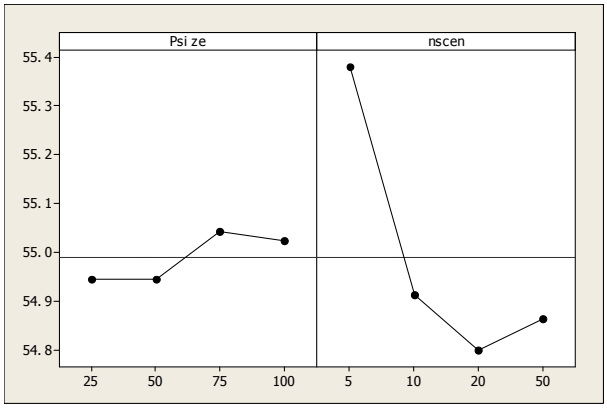

Fig. 3. Factor level trend of the CTLBO.

C. Comparison with Other Heuristics

To test the performances of the CTLBO for the SRCPSP,

comparisons between the CTLBO and the state-of-art algorithms are carried out. The comparative algorithms include GA with activity-based policies [9] denoted as "ABGA", GRASP with activity-based policies [10] denoted as "ABGR", two-phase GA with pre-processing policies [11] denoted as "PPGA" and OCRO with resource-based policies [12] denoted as "OCRO". The APD results using 5,000 and 25,000 schedules as the terminal condition are listed in Table IV-V, respectively, where DEV denotes the deviation showing the difference between the APD of the CTLBO and that of the best known solution.

TABLE IV
ALGORITHM COMPARISON (5,000 SCHEDULES)

Algorithm	Distribution				
	U1	U2	Exp	B1	B2
ABGA[9]	52.14	78.65	120.22	-	-
ABGR[10]	**46.84**	72.58	114.42	**47.17**	75.97
PPGA[11]	48.86	58.91	76.03	49.01	58.82
OCRO[12]	47.98	58.28	74.16	48.36	60.02
CTLBO	49.89	**58.14**	**73.70**	49.90	**58.15**
DEV	6.51%	0.00%	0.00%	5.79%	0.00%

TABLE V
ALGORITHM COMPARISON (25,000 SCHEDULES)

Algorithm	Distribution				
	U1	U2	Exp	B1	B2
ABGA[9]	49.63	75.38	116.83	-	-
ABGR[10]	**45.21**	70.95	112.37	**45.60**	74.17
PPGA[11]	47.21	58.07	74.56	47.25	57.95
OCRO[12]	47.39	57.83	73.74	47.79	59.60
CTLBO	48.65	**57.22**	**72.98**	48.71	**57.17**
DEV	7.61%	0.00%	0.00%	6.82%	0.00%

From Table IV and V, it can be seen that the CTLBO is superior to other algorithms for the problems with medium and large variances, i.e., U2, B2 and Exp. Compared to other algorithms, the effectiveness of CTLBO mainly owes to the following two aspects. First, the students' overall performance can be improved by the competition process of students in different classes. Second, collaboration in both teacher phase and student phase enhances the search capability. For the other algorithms, competition or collaboration phase is not adopted.

As for the distributions with small variances, like U1 and B1, the ABGR ranks the top one and the CTLBO ranks in the middle place, but the DEV of the CTLBO from that of the ABGR is less than 8%. A possible reason might be the conclusion experimentally shown in [11]. That is, the activity-based policies could perform better for the problems with small variances. In our experiments, the CTLBO employs the resource-based policies, while the ABGR adopts the activity-based policies. So, the ABGR is superior to the CTLBO for U1 and B1; but, for U2, B2 and Exp the CTLBO performs better.

VI. CONCLUSIONS

In this paper, a co-evolutionary TLBO was proposed to solve the stochastic RCPSP. Compared to the original TLBO, the CTLBO adopted competition and cooperation mechanisms to balance exploration and exploitation for population-based search. In addition to the analysis of computational complexity, the effect of parameter setting is investigated by design of experiments. Computational testing results using 600 instances with five different random distributions were presented. The comparisons to the existing algorithms demonstrated the effectiveness of the CTLBO for solving the SRCPSP with medium and large variances. Further work could focus on studying some adaptive CTLBO algorithms by considering the problem-specific characteristics, and the idea of ordinal optimization to handle stochastic optimization could be employed. It might also be interesting to study the CTLBO algorithm for the SRCPSP with multiple objectives or multiple projects.

REFERENCES

[1] S. Hartmann and R. Kolisch, "Experimental evaluation of state-of-the-art heuristics for the resource-constrained project scheduling problem," *Eur. J. Oper. Res.,* vol. 127, pp. 394–407, Dec. 2000.

[2] R. Kolisch and S. Hartmann, "Experimental investigation of heuristics for resource-constrained project scheduling: An update," *Eur. J. Oper. Res.,* vol. 174, pp. 23–37, Oct. 2006.

[3] L. Wang and C. Fang, "A hybrid estimation of distribution algorithm for solving the resource-constrained project scheduling problem," *Expert. Syst. Appl.,* vol. 39, pp. 2451–2460, Feb. 2012.

[4] C. Fang and L. Wang, "An effective shuffled frog-leaping algorithm for resource-constrained project scheduling problem," *Comput. Oper. Res.,* vol. 39, pp. 890–901, May. 2012.

[5] L. Wang and C. Fang, "An effective shuffled frog-leaping algorithm for multi-mode resource-constrained project scheduling problem," *Inform. Sciences,* vol. 181, pp. 4804–4822, Oct. 2011.

[6] L. Wang and C. Fang, "An effective estimation of distribution algorithm for the multi-mode resource-constrained project scheduling problem," *Comput. Oper. Res.,* vol. 39, pp. 449–460, Feb. 2012.

[7] W. Herroelen and R. Leus, "Project scheduling under uncertainty: Survey and research potentials," *Eur. J. Oper. Res.,* vol. 165, pp. 289–306, Sep. 2005.

[8] F. Stork, "Branch-and-bound algorithms for stochastic resource-constrained project scheduling," Technische Universität Berlin, Fachbereich Mathematik, Tech. Rep. Nov. 2000.

[9] F. Ballestín, "When it is worthwhile to work with the stochastic RCPSP," *J. Scheduling,* vol. 10, pp. 153–166, Jun. 2007.

[10] F. Ballestín and R. Leus, "Resource-constrained project scheduling for timely project completion with stochastic activity durations," *Prod. Oper. Manag.,* vol. 18, pp. 459–474, Jul. 2009.

[11] B. Ashtiani, R. Leus, and M. B. Aryanezhad, "New competitive results for the stochastic resource-constrained project scheduling problem: exploring the benefits of pre-processing," *J. Scheduling,* vol. 14, pp. 157–171, Apr. 2011.

[12] H. Zheng, L. Wang, S. Wang, and C. Fang, "Ordinal chemical reaction optimization for stochastic resource constrained project scheduling problem," in *Proc. 32nd Chin. Control Conf.*, Xi'an, China, 2013, pp. 2437–2442.

[13] J. Blazewicz, J. K. Lenstra, and A. H. G. Kan, "Scheduling subject to resource constraints: classification and complexity," *Discrete. Appl. Math.,* vol. 5, pp. 11–24, Jan. 1983.

[14] A. Gogna and A. Tayal, "Metaheuristics: review and application," *J. Exp. Theor. Artif. In.,* pp. 1–24, May. 2013.

[15] J. Gu, M. Gu, C. Cao, and X. Gu, "A novel competitive co-evolutionary quantum genetic algorithm for stochastic job shop scheduling problem," *Comput. Oper. Res.,* vol. 37, pp. 927–937, May. 2010.

[16] N. García-Pedrajas, J. A. R. Del Castillo, and D. Ortiz-Boyer, "A cooperative coevolutionary algorithm for instance selection for instance-based learning," *Mach. Learn.,* vol. 78, pp. 381─420, Mar. 2010.

[17] H. F. Teng, Y. Chen, W. Zeng, Y. J. Shi, and Q. H. Hu, "A dual-system variable-grain cooperative coevolutionary algorithm: satellite-module layout design," *IEEE Trans. Evolut. Comput.,* vol. 14, pp. 438─455, Jun. 2010.

[18] M. Li and Z. Wang, "A hybrid coevolutionary algorithm for designing fuzzy classifiers," *Inform. Sciences,* vol. 179, pp. 1970─1983, May. 2009.

[19] H. F. Wang and Y. Y. Chen, "A coevolutionary algorithm for the flexible delivery and pickup problem with time windows," *Int. J. Prod. Econ.,* vol. 141, pp. 4─13, Jan. 2013.

[20] F. Van den Bergh and A. P. Engelbrecht, "A cooperative approach to particle swarm optimization," *IEEE Trans. Evolut. Comput.,* vol. 8, pp. 225─239, Jun. 2004.

[21] R. V. Rao, V. J. Savsani, and D. P. Vakharia, "Teaching-learning-based optimization: A novel method for constrained mechanical design optimization problems," *Comput. Aided. Design,* vol. 43, pp. 303─315, Mar. 2011.

[22] K. R. Krishnanand, B. K. Panigrahi, P. K. Rout, and A. Mohapatra, "Application of multi-objective teaching-learning-based algorithm to an economic load dispatch problem with incommensurable objectives," in *Swarm, Evolutionary, and Memetic Computing*, Ed, Springer, 2011, pp. 697─705.

[23] V. Toğan, "Design of planar steel frames using teaching-learning based optimization," *Eng. Struct.,* vol. 34, pp. 225─232, Jan. 2012.

[24] R. V. Rao and V. Patel, "Multi-objective optimization of heat exchangers using a modified teaching-learning-based optimization algorithm," *Appl. Math. Model.,* vol. 37, pp. 1147─1162, Feb. 2013.

[25] R. V. Rao, V. J. Savsani, and J. Balic, "Teaching-learning-based optimization algorithm for unconstrained and constrained real-parameter optimization problems," *Eng. Optimiz.,* vol. 44, pp. 1447─1462, Mar. 2012.

[26] R. Kolisch, "Serial and parallel resource-constrained project scheduling methods revisited: Theory and computation," *Eur. J. Oper. Res.,* vol. 90, pp. 320─333, Apr. 1996.

[27] F. Stork, "Stochastic resource-constrained project scheduling," Ph.D. dissertation, Technische Universität Berlin, 2001.

[28] P. J. Angeline and J. B. Pollack, "Competitive environments evolve better solutions for complex tasks," in *Proc. 5th Int. Conf. Genet. Algorithms*, San Mateo, CA, 1993, pp. 264─270.

[29] S. Hartmann, "A competitive genetic algorithm for resource‐constrained project scheduling," *Nav. Res. Log.,* vol. 45, pp. 733─750, Oct. 1998.

[30] R. Kolisch and A. Sprecher, "PSPLIB-a project scheduling problem library: OR software-ORSEP operations research software exchange program," *Eur. J. Oper. Res.,* vol. 96, pp. 205─216, Jan. 1997.

[31] D.C. Montgomery, *Design and analysis of experiments*. Arizona: John Wiley and Sons, 2005.