JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue

JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)

An International Scholarly Open Access, Peer-reviewed, Refereed Journal

A Modified Discrete Teaching Learning Based Optimization Algorithm for the Resource Constrained Project Scheduling Problem

[1]Bidisha Roy, [2]Dr. Asim Kumar Sen

[1]Research Scholar, [2]Research Advisor

[1]Department of Computer Engineering,

[1]St. Francis Institute of Technology, Mumbai, India

Abstract : Many problems in the field of Operations Research and Project Management can be classified as Optimization Problems which are classically NP-Hard. One such important problem is the Resource Constrained Project Scheduling Problem, abbreviated as the RCPSP. It is a Combinatorial Optimization problem involving resource as well as precedence constraints with temporal conditions. But, the RCPSP requires exponential computational resources as the problem complexity increases, as it is intractable. Thus, Soft Computing based approaches provide better results towards solving the problem optimally. This paper presents the usage of a discrete version of the Teaching Learning Based Optimization (TLBO) metaheuristic to solve the problem. Also, a hybrid model using 2-point crossover inspired by the Genetic Algorithm is presented. The proposed models are extensively tested on well-known benchmark test instances and analyzed. The promising results demonstrate the efficiency of the proposed solutions for solving the RCPSP problem of varying magnitudes.

IndexTerms **– Scheduling, Optimization; RCPSP (Resource Constrained Project Scheduling Problem), Soft Computing, Metaheuristics, Swarm Intelligence, Nature Inspired Algorithms, Teaching Learning Based Optimization Algorithm (TLBO), 2-point Crossover.**

I. INTRODUCTION

Project Scheduling is one of the most important tasks in Project Management. It is a decision-making process that aims to find the optimal completion time of the project, also called the makespan, by arranging the activities in the project in such a manner that the requisite constraints are satisfied. The success of a project and hence the business of an organization, thus largely depends on how effectively and efficiently the activities in a project are scheduled. Most of the tools and techniques used to schedule activities in a project, however assume that the availability of resources to be used in a project is unlimited over the time horizon of the project. However, in most practical scenarios, resources such as humans, machines, money etc. are available in limited capacities when several activities are going on concurrently in a project. Also, as industries are progressing towards Industry 4.0, it is becoming more and more imperative to schedule projects such that the limited resources are used more intelligently, along with optimizing the makespan of the project. Hence, a more representative approach is to take into consideration the limited availability of the resources while scheduling the activities in a project. This creates a new variant of the scheduling problem known as the Resource Constrained Project Scheduling Problem, popularly abbreviated as the [1].

The RCPSP is an Optimization Problem with constraints whose objective is to schedule the activities in a project with limited resources in such a way that, both the precedence as well as resource constraints are satisfied and the duration or the makespan is minimized. RCPSP is a key problem in many industries such as construction, Aircraft maintenance etc. as the availability of resources in a constrained manner affects the scheduling of such projects decisively. As the problem is relatively general with multiple application areas, also exhibiting complex, large-scale and non-linear behaviours, it has been a widely studied and researched area amongst engineers and scientists, both in the field of Operations Research (OR) as well as Combinatorial Optimization (CO).

Being such challenging problem both in terms of research and practical applications, RCPSP has received vigorous attention for almost two decades now and is still a very active and continuing area of research. Early research proposed many exact methods based on techniques like Branch and Bound, Dynamic Programming, Mixed Integer and Linear Programming etc. [2] to solve the problem. However, Blazewicz et al. [3] described RCPSP and other constrained scheduling problems as Combinatorial Optimization problems which are NP-Hard in the strongest sense. Hence, though the exact methods did solve the problem optimally for smaller size instances, they were unable to provide a polynomial time solution satisfactorily as the problem size increased. The research focus, then shifted towards the use of greedy technique based approximate techniques to solve the problem. In the last two decades, there has been an ongoing research towards use of Nature Inspired Soft Computing based approaches to find solutions to large size instances of the problem. Unlike exact methods which come with a guarantee of optimal solution, these methods act as a best-effort delivery method finding optimal solution in most cases and suitable feasible solutions almost always as they emulate characteristics found in natural habitat of species like basic nature, adaptability, cooperation etc.

In this research, we investigate the use of a latest metaheuristic viz., the Teaching Learning Based Optimization Algorithm (TLBO) given by Rao et al. [4] towards solving the RCPSP problem. In recent years, the TLBO has emerged as an efficient and prominent metaheuristic to solve a variety of optimization problems with results quite comparable to other metaheuristic algorithms. Along with the classic TLBO, another algorithm which combines the steps of the TLBO with the advantages of the crossover operation inspired by Genetic Algorithm (GA) is also investigated. The results of the investigation are presented by testing them on benchmark RCPSP test instances and comparing them with earlier seminal metaheuristics. The numerous results and comparisons show the competitiveness and superiority of the investigated research.

The remainder of the paper is organized as follows: Section 2 discusses previous related work in the field of RCPSP solutions, both heuristic and metaheuristic. Section 3 formulates the RCPSP problem while Section 4 discusses both the TLBO based approaches proposed in this research. Section 5 provides the results of the conducted experiments on the benchmark instances. Section 6 provides the analysis and conclusion of the results along with the probable future directions in this research.

II. RELATED WORK

Being a class of NP-Hard Optimization Problems, RCPSP is an important intractable problem in the domain of Operations Research, which has attracted researchers to this field. Many researches have proposed many approaches to solve the problem which can be broadly classified into three categories: deterministic or exact methods, heuristic and soft computing-based metaheuristic methods.

Initial researches had proposed exact methods to solve the problem using problem solving techniques like Mixed Integer Programming, Branch and Bound and Dynamic Programming [5]. Though these methods come with the assurance of optimal solution, they can be used to solve only small sized instances of the problem since computational efforts and execution time increase exponentially as the size of the problem increases. This limitation has motivated researchers to look for approximate algorithms either based on heuristics or Artificial Intelligence based metaheuristics to find solutions for large-sized practical problems in an acceptable computational effort.

Several heuristic methods have been proposed to obtain near-optimal solutions for large sized instances in a rational amount of time. These are problem specific methods which begin with an initial empty solution set which are subsequently filled up with activities iteratively based on either Priority Rule Based Heuristics or Schedule Generation Schemes (SGS). In Priority based methods, priority values are calculated for an activity based on some rules. The activities are then scheduled such that a good solution would be obtained. On the other hand, SGS encoding generates feasible schedules taking into consideration starting times of the activities based on their precedence. They apply two SGS schemes: Serial SGS based on activity incrementation or Parallel SGS based on time incrementation. Authors in [5][6][7][8] Kolisch and Hartmann, 1999, 2006; Kolisch and Padman, 2001) have put forward different heuristic techniques to solve the RCPSP problem which were based on different Schedule Generation Schemes (both serial and parallel), X-pass methods, Forward Backward Improvement (FBI) and also Priority Rule-Based Heuristics. Though Priority based-methods can solve large-sized RCPSP problem within acceptable time, they are hard to adapt to the constraints of problems dynamically. Hence SGS are more preferred as heuristics to solve large instances of the RCPSP problem.

In the last two decades, many researchers have explored the use of Nature Inspired Soft Computing metaheuristics over heuristics to better the solution provided by the heuristic methods. Metaheuristics are a class of methods used to solve optimization problems. They try to find solutions by emulating successful foraging behaviours and processes found in nature. Starting with an initial set of population constituting of initial feasible solutions, they keep evolving and improving over generations by applying a set of operations that transform current solutions to better solutions. Kolisch and Hartmann [5][7] experimented with metaheuristics like Genetic Algorithms (GA), Simulated Annealing (SA) and Tabu Search (TS) over the initial schedules found using earlier heuristic methods. The tests were carried out on benchmark test problems from the PSPLIB [9]. It was experimentally observed that the average standard deviation from optimal solutions was better using this strategy. These experiments also initiated the idea of first generating feasible solutions using heuristic methods and applying various Soft Computing based metaheuristic methods over them to select schedules which are optimal.

Over the years, several other Nature Inspired Soft Computing based Swarm Intelligence based metaheuristics which maintained a set of solutions in each iteration were proposed. One of the earliest such work was the usage of Ant Colony Optimization (ACO) suggested by Merkle and Middenhorf [2]. Many variations and improvements to this were suggested by various researchers [10][11]. These were based on Max-Min ACO or applying oblivion rate to the pheromone trail after every generation. Another popular metaheuristic that has gained popularity is the Particle Swarm Optimization (PSO) algorithm. Zhang et al. [12] put forward a solution based on the PSO with competitive results especially for the J30 test instances from the PSPLIB [9]. This led to various other researches using PSO with modifications [13][14]. Another popular metaheuristic that has been explored in the study of the RCPSP problem is the Bee Algorithm (BA) which is based on the foraging behaviour of honeybees in nature. Ziarati et al. [15](2011) suggested three different variations of the Bee Algorithms. Some discretized permutation-based Bee Algorithm techniques have been proposed by Nemmich et al. [16][17]. Some researches based on Hybridization of BA and PSO has also been suggested based on experiments carried out in Jia et. al. [18]. Some other prominent works include the usage of Nature Inspired metaheuristics like Cuckoo Search Algorithm [19][20], Flower Pollination Algorithm [21], Brain Storm Algorithm [22], Discrete Firefly Algorithm [23] amongst a few.

Metaheuristics are now often used to solve large-sized instances of NP-Hard problems due to their ability to produce reasonably good results in polynomial time. TLBO is a recent metaheuristic proposed by Rao et al. [4]. This algorithm emulates the teaching learning process of a class. It uses the Teacher and the Learners' as the vital components of the algorithm. The algorithm is designed to use the collective intelligence of the class to obtain optimum results. Many researches recently have proposed the use of the TLBO algorithm for a variety of constrained optimization problems [24][25][26] with competitive results. This has motivated us to apply the TLBO algorithm in our research. We also propose the use of 2-point crossover applied to the TLBO to solve the RCPSP problem. These have been tested on benchmark test instances from PSPLIB [9] and the results have been discussed and analysed.

III. PROBLEM FORMULATION

In this section, we formulate the model for the single-mode RCPSP problem. To represent the RCPSP for a project, the following information is needed [1]:

J	Set of n activities in a project
k	Number of resources.
R_k	Constant amount of R_k units of resource k available at any time
r_{jk}	During processing, activity j occupies r_{jk} units of resource k for k = 1, ..., r
p_j	Duration of an activity j
S_j	Start time an of activity j
C_j	Completion time of activity j given by $S_j + p_j$
$i \rightarrow j$	Set of all precedence between some activities i, j indicating that activity *j* cannot be started before activity *i* is completed

The objective of the RCPSP problem is to determine the Schedule set **S** for the project, in such a way that:

- At each time t the total demand for resource k is not greater than the availability R_k for k = 1, ... , r,
- The precedence constraints are fulfilled, i. e. $S_i + p_i \leq S_j$ if i → j ,
- Some objective functions f(C_1, ... , Cn) are minimized with C denoting Completion time of project

Thus formally RCPSP would be defined by a tuple (V, p, E, R, B, b), where:

$\boldsymbol{V = \{A_0, ..., A_{n+1}\}}$	Activities constituting the project, where n is the number of activities in a project, activities 0 and n+1 being dummy activities, denoting the start point and the end point of a project respectively
P	Duration set, $\boldsymbol{p_0, p_{n+1} = 0}$
E	Set of precedence, given by pairs $(A_i, A_j) \in E$ means that activity A_i precedes activity A_j
R	Renewable resources' set
B	Availabilities of resources such that B_k denotes the availability of R_k
B	Demands of activities for resources, $\boldsymbol{b_{ik}}$ representing the amount of resource R_k used per time period during the execution of A_i

For the above information provided regarding a project, we need to construct an optimal schedule.

A solution S is **feasible** if it is compatible with both the Precedence Constraints as well as the Resource Constraints. Thus, the mathematical model of the RCPSP would be as follows:

Objective Function:

Min (C_{max}) (1)

Subject to constraints:

$$Sj - Si \geq pi \forall (Ai, Aj) \in E \quad (2)$$

$$\sum_{Ai \in At} bik \leq Bk, \forall Rk \in R, \forall t \geq 0 \quad (3)$$

$$C_j \geq 0 \quad (4)$$

Thus, in RCPSP we need to find a Schedule S which has the minimum time considering the constraints of precedence of activities and resources available at hand.

The RCPSP can be demonstrated with a small example as given in Figure 1.

The RCPSP is represented here as a directed acyclic Activity on Node (AoN) graph. Activities are represented as nodes and the precedences as directed paths between activities. There are 6 non-dummy activities represented numbered as nodes 1-6 which have to be scheduled. They would be utilizing K = 2 renewable resources which have capacity of 4 and 2 units respectively. Equation 4 gives the constraint of the completion time decision variable.

The main objective is to identify a schedule of minimum makespan (Eq. 1), taking into consideration the precedence constraint (Eq. 2) and resource constraint (Eq. 3). The constraint in Eq. 4 depicts the constraint of the completion time decision variable.

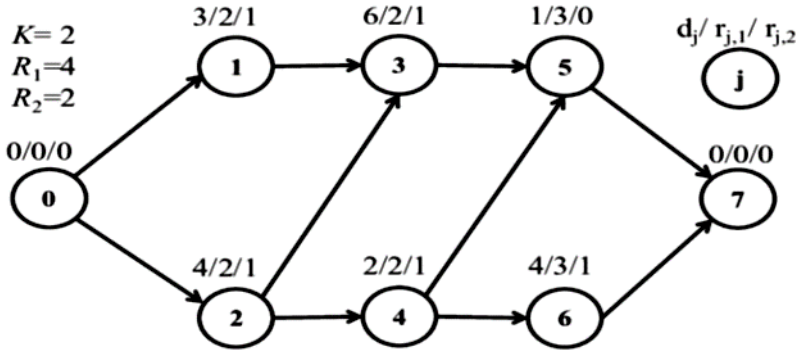

Figure 1: RCPSP Example

A feasible schedule of the above example is shown in Figure 2, with an optimal makespan of 15 units

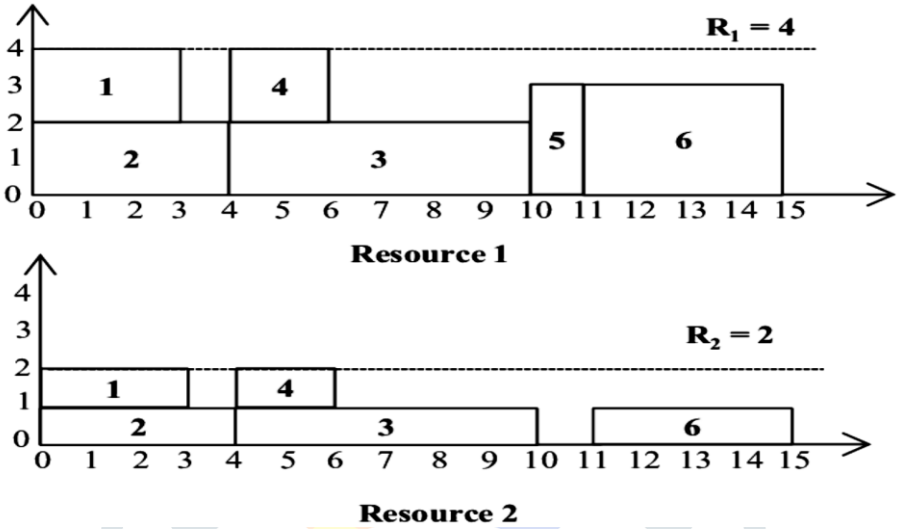

Figure 2: A feasible schedule for the given example

IV. TEACHING LEARNING BASED OPTIMIZATION

Like most evolutionary algorithms in the field of computational intelligence, TLBO [4] also employs an optimization method that is population-based. It approaches the global optima using a population of solutions. It takes inspiration from the fact that teachers as well as peers amongst learners influence the output of learners in the class. The algorithm consists of two vital entities in the system; The teacher and the learners. The working of the algorithm is based on two modes of learning: the interaction between the teacher and the learners, known as the Teacher Phase and the interaction amongst the learners known as the Learner Phase. Both these phases of the basic TLBO are explained below.

4.1 Basic TLBO

4.1.1 Teacher Phase

This is the first phase wherein the learners try to learn from the teacher whereas the teacher through their knowledge attempts to increase the mean result of the class. In this phase, the individual i.e., the solution with the best fitness value is chosen as the Teacher of the class. If we consider an objective function f(x) with d-dimensional variables, the ith learner would be a feasible solution set $X_i = [x_{i1}, x_{i2}, \dots x_{id}]$. In a class of m learners, the mean of the class would be given by $X_{mean} = {}^1/_m\ [\sum_{i=1}^{m} x_{i1}, \sum_{i=1}^{m} x_{i2}, \dots\dots \sum_{i=1}^{m} x_{id}]$. If the teacher i.e., the learner with the best fitness is represented as XTeacher, the position of learners' are updated in each iteration as follows:

$$X_{i,new} = X_{i,old} + r_i(X_{Teacher} - T_F X_{mean}) \quad (5)$$

Where $X_{i,new}$ and $X_{i,old}$ are the updated and initial positions of the i[th] learner. r_i is a random number in the range of [0,1]. T_F is the teaching factor which denotes the mean value to be changed. Calculation of T_F is a heuristic step given by $T_F = round(1 + rand(0,1)\{2-1\})$, to keep the decision random with equal probability. T_F can take values 1 or 2 with 1 corresponding to no increase in the knowledge level and 2 corresponding to transfer of knowledge from teacher to student. The new solution is updated if it is found to be better than the older one.

4.1.2 Learner Phase

In this phase, the learners learn from each other due to peer interaction. If another learner has more knowledge than the current learner, the current learner also benefits through it. An individual learner X_j randomly selects another learner X_k ($j \neq k$), and the learning happens as follows:

$$X_{j,new} = X_{j,old} + r_j(X_j - X_k), \quad if\ f(X_j) < f(X_k) \tag{6}$$

$$X_{j,new} = X_{j,old} + r_j(X_k - X_j), \quad if\ f(X_j) > f(X_k) \tag{7}$$

The new value is accepted if it gives a better fitness value. Since it is a minimization problem, the lesser value of the objective function is the fitter value.

4.2 TLBO for RCPSP

The standard TLBO has been adapted to solve the RCPSP problem in this research. The framework for the proposed approach is given by the flowchart as shown in Figure. 3.

4.2.1 Solution/Population Representation for the RCPSP and Schedule Generation

The representation and encoding of the population in the RCPSP is an essential step for better algorithmic performance. Though the solution can be represented in many forms, Kolisch and Hartmann in their research [6], have mentioned that the RCPSP can be heuristically solved using an Activity List (permutation-based) representation or Random Key List (priority-key) representation. In our investigations, we have used an Activity List (AL) representation to generate the initial solutions, which was proven to be more suitable for single-mode RCPSP problems. An AL list representation is a more suited representation as it is easily implemented and also quickly decoded. Also, there always is an AL schedule that induces an optimal schedule [27].

Every population member or learner in the TLBO is an activity vector that represents a possible solution in the n-dimensional parameter space, where ***n*** represents the number of activities in the project. The index of the activity represents the order of the activity in the sequence. The encoding scheme lists the activities such that the precedence constraints have been maintained i.e. the predecessor should be indexed before its successors. As RCPSP is a discrete and deterministic problem, all the activities are integers and are distinct, the search space required by this permutation-based encoding scheme is drastically reduced.

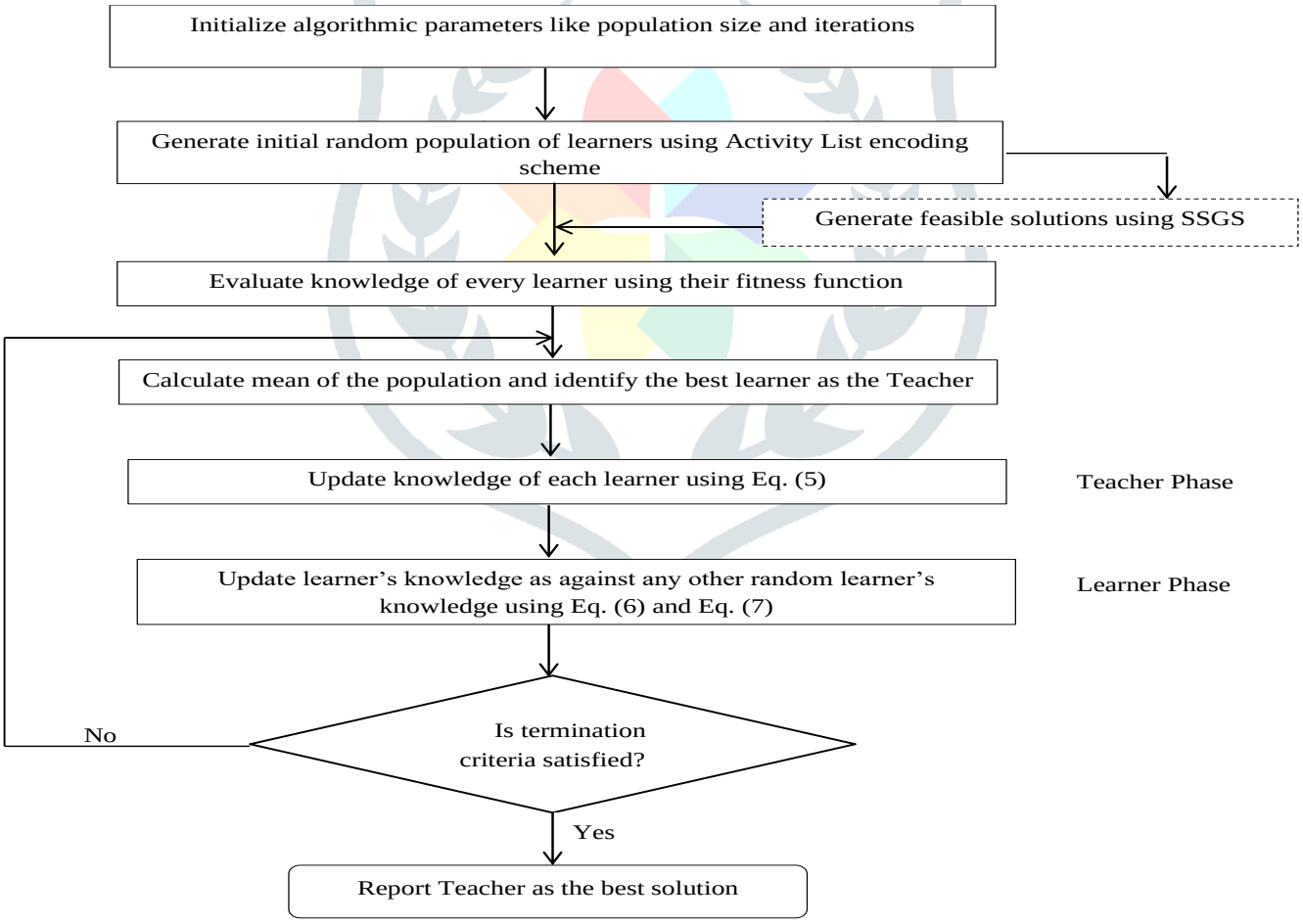

Figure 3. TLBO Algorithm Flowchart for the RCPSP Problem

An example, an Activity List for the example shown in Figure. 1 can be described as shown in Figure. 4. Activities 0 and 7 are the dummy activities to represent the beginning and end of the project respectively.

0	2	1	3	4	5	6	7

Figure 4. Example of AL representation

These ALs need to be made feasible so that they can be evaluated to find the best solution. Hence, Schedule Generation Schemes (SGS) are used to produce active non-delay schedules from the ALs. SGS is the fundamental decoding procedure for the RCPSP that takes into consideration both the precedence constraints as well as the availability of the constrained resources at hand [7]. It is a heuristic that starts from schedule of zero activities and adds activities to be scheduled through iterative improvements. There are two types of SGS viz; Serial SGS (SSGS) that generates schedules using activity incrementation and Parallel SGS (PSGS) which performs time incrementation to add activities iteratively in each time interval. In this research, SSGS decoding has been adopted for generating active schedules. The SSGS is applied by most metaheuristics for the RCPSP [7] as it is found to be most suitable for permutation-based encoding as it always generates active non-delay (feasible) schedules. Hence searching the solution space using a metaheuristic would have greater probability of returning optimal results.

SSGS generates the entire generation of a feasible schedule in ***n*** iterations, ***n*** being the number of activities in project to be scheduled. In each iteration, an activity is taken and is scheduled at its earliest feasible completion time satisfying both the resource and precedence constraints. On the **nth** iteration, the schedule terminates as all the non-dummy activities have been scheduled. The makespan is given by the maximum completion time amongst all the activities preceding dummy activity ***n+1***. The SSGS operates in time $O(n^2R)$, R being the resources available for the project. The makespan determines the fitness value of every learner i.e., every feasible schedule.

4.2.1 Teacher Phase for RCPSP

The schedule with the minimum makespan i.e., maximum fitness in the entire population (generated using SSGS) is selected as the Teacher. The mean for the entire population is calculated. Every learner is updated based on Eq. (5) if the updated value has a better fitness (calculated using SSGS). However, the equations for classic TLBO are mostly designed for continuous optimization problems. RCPSP, though, is a discrete optimization problem with every activity in the schedule list being distinct. Hence, the equations for the mean and updating the learner are modified by rounding of the results. Also, since we need distinct activities from 1 to n in a project, the Eq. (5) is modified as follows:

$$X_{i,new} = round(X_{i,old} + abs(\frac{2*r_i(X_{Teacher} - T_F X_{mean})}{100})) \quad (8)$$

Thus, algorithm is modified to generate a Discrete TLBO formula for updating the learner in the Teacher Phase. A similar modification is done in the Learner Phase which is explained in the next subsection.

4.2.3 Learner Phase for RCPSP

As mentioned earlier in the Learner Phase, the learners' enhance their knowledge by peer interaction with other learners. Hence, the current learner *j* is compared with another learner *k* ($j \neq k$) as per the formulae given in Eq. (6) and Eq. (7). However, to make it suitable to a discrete problem, the equations are modified as follows:

$$X_{j,new} = round\left(X_{j,old} + abs\left(\frac{2*r_j(X_j - X_k)}{100}\right)\right), if\ f(X_j) < f(X_k) \quad (9)$$

$$X_{j,new} = round\left(X_{j,old} + abs\left(\frac{2*r_j(X_j - X_k)}{100}\right)\right), if\ f(X_j) > f(X_k) \quad (10)$$

In this way, a Discrete TLBO (DTLBO) variant of the classic TLBO has been implemented to solve the discrete RCPSP problem.

The Teacher and the learner phase are iteratively applied over the number of generations. Once the termination criteria is satisfied, the Teacher is returned as the best solution for the project.

4.3 2-point Crossover

In the previous subsection, a DTLBO algorithm to solve the RCPSP problem that was adopted has been discussed. Hartmann [28] has mentioned few competitive techniques using concepts of Genetic Algorithm to solve this problem.

Hence, we adopt a novel approach to solve the RCPSP problem, wherein the updating of the Learner in both the phases is done using a 2-point crossover as in Genetic Algorithm. The steps for the TLBO, however remain the same. The framework for the same is given by the following flowchart in Fig. 4.

As seen in the Fig 4, the initial encoding and decoding process is the same as the DLTBO implementation. Once, the initial schedules are generated, they are converted into feasible solutions using SSGS as before. Also, the solution with the best fitness i.e., makespan is selected as the teacher and the mean is for the class is calculated.

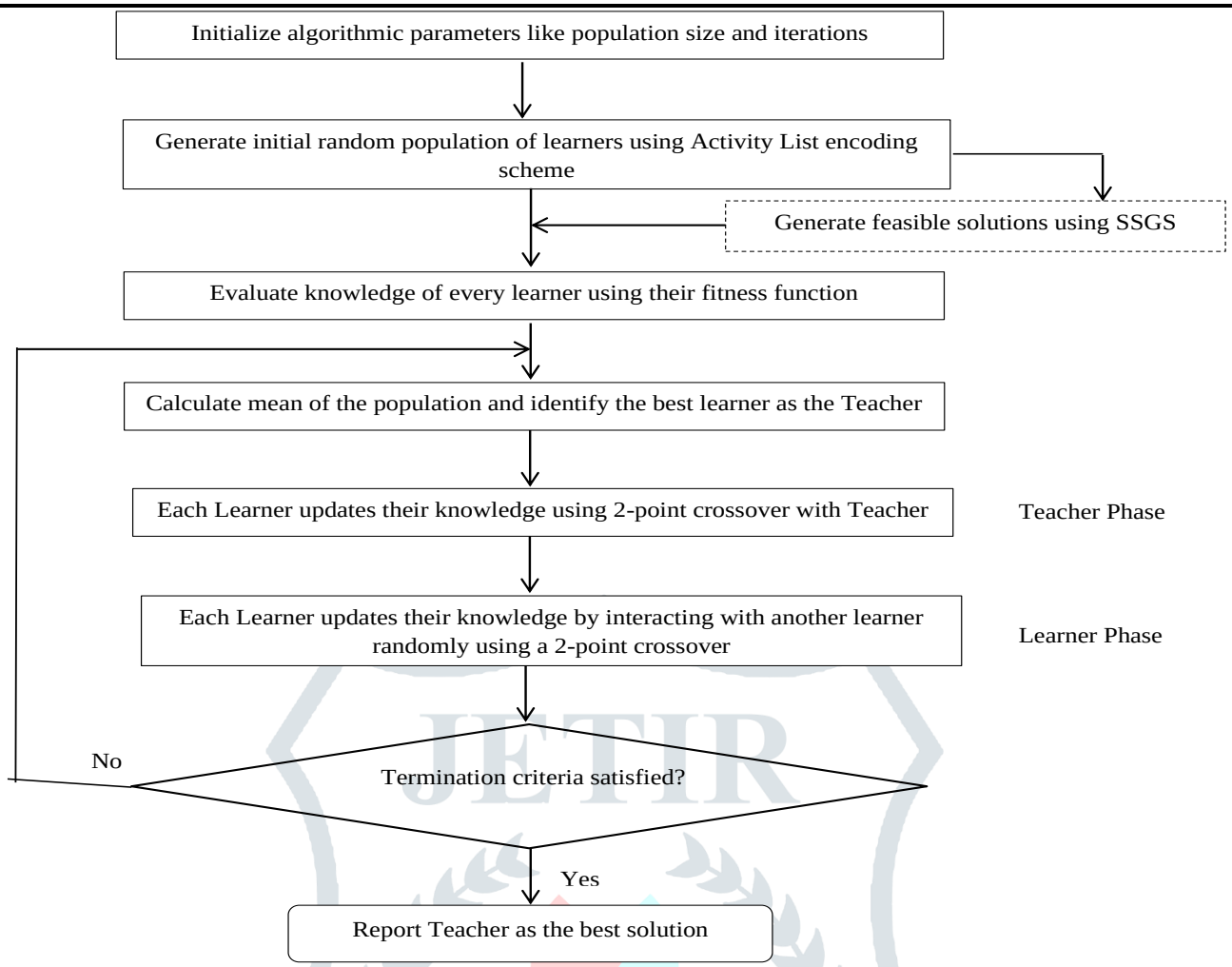

Figure 4. TLBO Algorithm using 2-point crossover Flowchart for the RCPSP Problem

4.3.1 Teacher and Learner Phases

As has been mentioned in the previous subsections, the Teacher i.e., the individual with the best fitness transfers his/her knowledge to the other individuals in the class i.e., the Learners. Here, we do this using a two-point crossover [28]. As an example, consider two individuals, I^1 and I^2 as the Teacher and a Learner Respectively. Let their feasible schedules be depicted as shown in Figure. 5, with 0 and 7 being the dummy activities.

This is for the example illustrated in Figure. 2

0	2	1	3	4	5	6	7

I^1

0	1	2	4	3	6	5	7

I^2

Figure 5. Feasible Schedules of Teacher and a Learner

We randomly generate two integers u_1 and u_2 in the range [1, n]. Let the values generated by random number generation be u_1=2 and u_2=4. A new individual I^{new} is generated using the following crossover operations:

$$I_j^{new} = I_j^2, \quad 1 \le j \le u_1 \tag{11}$$

$$I_j^{new} = I_k^1, \quad k = \min\{k | I_k^2 \notin \{I_k^{new}, \dots, I_{u1}^{new}\},\ u_1 + 1 \le j \le u_2 \tag{12}$$

$$I_j^{new} = I_k^2, \quad k = \min\{k | I_k^2 \notin \{I_k^{new}, \dots, I_{u2}^{new}\},\ u_2 + 1 \le j \le n \tag{13}$$

Hence, the new Individual based on I^1 and I^2 would be as illustrated in Figure 6. The new individual is replaced if it has a better fitness value than the fitness of the current learner.

0	1	2	3	4	6	5	7

u1	u2	I^{new}

Figure 6. Individual generated after two-point crossover

Similarly, in the Learner Phase, where the learners learn from each other, we randomly select another learner and perform a two-point crossover [28] on the Activity Lists to generate the new Individual Learner. If the new individual is better, it replaces the older learner.

This process is carried out for all learners till the termination criterion is satisfied, post which the algorithm terminates by returning the updated Teacher as the best solution.

V. EXPERIMENTAL RESULTS AND COMPARISONS

In this section we report and discuss the results of the extensive computational experiments carried out to inspect the performances of the proposed algorithms mentioned in the previous section. Both the proposed models are investigated using the standard benchmark data sets from the PSPLIB [9]. The PSPLIB consists of 4 different types of datasets viz; J30, J60, J90 and J120. J30, J60 and J90 are sets of projects having 30, 60 and 90 activities (non-dummy) respectively. These sets contain 480 such instances that can be comprehensively tested. Similarly, the J120 set consists of 600 project instances of 120 non-dummy activities each. They are generated using the ProGen Generator. Each instance is generated using 3 main parameters, viz; Resource Strength (RS), Resource Factor (RF) and Network Complexity (NC). More details about the instances and their generation can be found in Sprecher and Kolisch (1997). Table 1 illustrates an example of the 30 activities instance (J301_10) in the PSPLIB. As seen in the table, this project instance consists of 30 activities and 4 types of resources, with every activity having a maximum of 3 successors. Since the RCPSP is such a complex problem, optimal solutions are known only for the J30 instances. For the other instances, critical path lower bound solutions are provided. To be able to provide a reasonable comparison with other seminal models, the instances are tested over 1000 and 5000 schedules. The solutions are measured on the basis of average deviation *(Dev_Avg)* from the optimal schedules for the J30 instances and average deviation from best solutions obtained from lower bound critical path methods for the other instances. *Dev_Avg* is the deviation of the solution given by the proposed model and the best solutions stored in the data sets.

Table 1. Instance J301_10 with precedence and resource requirements

Activity#	Successors			Duration	Required Resources			
					R 1	R 2	R 3	R 4
1 (dummy)	2	3	4	0	0	0	0	0
2	5	7	20	4	1	0	0	0
3	22			3	0	0	0	2
4	18			10	0	0	0	4
5	6	9	10	10	0	0	1	0
6	30			5	0	10	0	0
7	8	11	21	1	0	3	0	0
8	30			7	3	0	0	0
9	13	14	16	4	0	0	0	1
10	19			5	0	0	0	6
11	12			7	0	0	0	6
12	17	25		10	0	3	0	0
13	21	23		2	0	7	0	0
14	15	27		4	0	0	2	0
15	28			5	0	5	0	0
16	29			5	0	0	0	1
17	24			4	0	0	0	4
18	19	23		3	0	0	7	0
19	31			7	0	0	0	3
20	23			6	8	0	0	0
21	24			2	0	0	2	0
22	26	27		5	0	0	8	0
23	26	31		8	10	0	0	0
24	28			3	0	0	7	0
25	27	28		3	5	0	0	0
26	30			2	8	0	0	0
27	31			10	0	0	5	0
28	29			3	0	0	0	9
29	32			5	8	0	0	0
30	32			5	7	0	0	0
31	32			1	5	0	0	0
32 (dummy)				0	0	0	0	0
Resource Availabilities →					11	12	9	9

The *Dev_Avg* in percentages is as given in Eq. (14):

$$Dev\_Avg = \frac{\sum_{i \in BI} \frac{(Obtained_i - Best_i)}{Best_i} * 100}{instances} \quad (14)$$

Where *BI* denotes Benchmark Instances.

5.1 Parameter Settings

The TLBO algorithm does not require setting of parameters. The only factor that is calculated i.e., the TF takes value between 1 and 2 and are calculated randomly with every population member. The TLBO is a fast-converging algorithm. Hence, even if the population size is large, it does not slow down the algorithm. By experimentation, the population size i.e., the number of learners is kept at 100.

The models have been implemented using Matlab R2014a on a CORE i5 10th Generation 16 GHz machine. Initially, we have conducted 10 independent runs of both the DTLBO and the TLBO-2point crossover for 10% of the test instances of all the data sets without any repetition. The results of the independent results are as shown in Table 2.

Table 2. Average Deviation on training instances for DTLBO and TLBO-2pt crossover

Expt. No.	Average Deviation							
	J30 *Dev_Avg*		J60 *Dev_Avg*		J90 *Dev_Avg*		J120 *Dev_Avg*	
	DTLBO	TLBO-2pt	DTLBO	TLBO-2pt	DTLBO	TLBO-2pt	DTLBO	TLBO-2pt
1	2.45	0.99	12.77	11.42	15.74	14.68	35.73	33.602
2	2.22	0.64	13.32	11.78	16.13	15.15	35.20	33.989
3	2.58	1.01	12.54	12.28	15.50	15.31	35.695	32.567
4	2.81	0.72	13.26	11.98	15.24	14.58	36.542	33.092
5	2.87	0.94	12.91	12.13	16.04	15.09	35.826	33.613
6	2.61	1.05	12.98	11.48	14.70	13.98	35.553	33.134
7	2.67	1.07	13.10	11.87	15.17	14.22	35.03	32.706
8	2.41	0.65	12.59	12.18	15.80	14.94	35.70	32.41
9	2.97	1.23	13.26	11.45	16.11	15.15	35.07	33.198
10	2.82	0.98	12.83	11.60	15.44	14.43	35.31	32.561

From the results shown in the Table, it is evident that the hybrid TLBO with a 2-point crossover performs better than DTLBO for all the benchmark test instances. The DTLBO requires rounding off continuous values generated so that the sequence can be discretized. This is totally avoided in the 2-point crossover technique inspired by the GA [28]. Hence, the sequences generated are always discrete and distinct. Hence, a hybrid TLBO-2pt crossover model works better than the standard TLBO model which is discretized to solve the RCPSP problem. Figure. 7 shows the comparison of average deviation of all the training schedules for all the four test instances for both DTLBO and TLBO-2point crossover.

(a) (b)

(c) (d)

Figure 7. Comparison of DTLBO and TLBO-2pt over benchmark test instances

An exhaustive comparison of our model has been done with the best available metaheuristics in literature. These include models involving Genetic Algorithms [28], ACO [2][10][11], PSO [13][18], BA [[15][16][17]. To be able to present a fair comparison, the TLBO model too has been tested over 1000 and 5000 schedules for 30 independent runs for lower and medium size test instances and 20 independent runs for the J120 data set for its computational complexity.

Table. 3 presents the average standard deviation for the J120 test instances where all the instances have been tested for 1000 as well as 5000 schedules respectively.

Algorithm/Model	SGS	Schedules	
		1000	5000
GA [28]	Serial	39.37	36.74
GA [28]	Priority Rule	39.93	38.49
ACO [2]	Serial	35.56	35.56
MMAS [10]	Serial	--	31.55
ACO-DOR [11]	--	26.58	26.57
A-PSO [13]	Random Key	34.93	32.49
ABC [15]	Serial	43.24	39.87
BSO [15]	Serial	41.18	37.86
BA [15]	Serial	40.38	38.12
PBA [16]	Serial	42.85	38.45
ABC-PSO [18]	Serial	36.15	35.28
IDCS [19]	Serial	33.43	32.69
TLBO-2pt – This study	Serial	**33.35**	**32.08**

Table 4 and 5 show the average standard deviations from the lower bound critical path solutions for the J60 and the J30 case studies. There have not been many tests conducted on the J90 datasets to provide a substantial comparison. From the tables, we find that the proposed model using TLBO is very much competitive and comparable algorithm to solve the RCPSP problem. The deviations for J120 and J60 which are more indicative of medium and large sized projects show a great improvement in the standard deviation due to the TLBO algorithm. The deviation of the J30 test instances is also comparable. The usage of the 2-point crossover enhances the exploration search as well as and helps generate more potential learners for the next iteration/generation.

Algorithm/Model	SGS	Schedules	
		1000	5000
GA [28]	Serial	12.68	11.89
GA [28]	Priority Rule	13.30	12.74
ACO-DOR [11]	--	11.51	11.51
A-PSO [13]	Random Key	11.94	11.12
PSO-ICA [14]	--	12.58	12.36
ABC [15]	Serial	14.57	13.12
BSO [15]	Serial	13.67	12.70
BA [15]	Serial	13.35	12.83
PBA [16]	Serial	13.39	12.10
ABC-PSO [18]	Serial	12.14	11.90
IDCS [19]	Serial	11.78	10.99
TLBO-2pt – This study	Serial	11.82	11.02

The crossover method used in both the phases generates the new population based on present and historical random learner, allowing learners' positions in the new population modified with better probability. Both the Teacher and the Learner Phase are adapted suitably to accommodate the discrete and deterministic nature of the RCPSP problem for single-mode variant, thus enhancing the capability to find the global optima balancing both the exploration and exploitation capability. The crossover operation randomly modifies the learners thereby enhancing the searchability in the global space. This variation helps improve the explorability of the standard algorithm which can otherwise get stuck in the local search space.

VI. CONCLUSION

It is a well-proven fact that Constrained Optimization Problems like the scheduling problems are the few of the most complicated Combinatorial Optimization Problems in literature, RCPSP being one of them. Also, due to its essential use in industry, an optimal solution to the problem is evidently needed. In this research, two variants of the TLBO i.e., the Discrete TLBO and the TLBO-2 point crossover have been applied to solve the RCPSP problem for its various benchmark test instances from the PSPLIB library. An exhaustive study and comparison of these techniques' vis a vis other prominent researches have also been presented. The computational experiments show that both the variants produce consistently good solutions with the 2-point crossover variant outperforming the DTLBO variant. The 2-point crossover operator introduced in the second variant improves the performance of the classic TLBO algorithm demonstrating that hybrid metaheuristics show better promise in reaching global optima for the RCPSP problem. These improved algorithms also require less parameters, the only important ones being the population size and the number of generations. Thus, hybrids with less parameters to be set can be future direction of investigation towards solving this problem optimally.

Algorithm/Model	SGS	Schedules	
		1000	5000
GA [28]	Serial	1.03	0.56
GA [28]	Priority Rule	1.38	1.12
A-PSO [13]	Random Key	0.28	0.06
PSO-ICA [14]	--	0.57	0.46
ABC [15]	Serial	0.98	0.57
BSO [15]	Serial	0.65	0.36
BA [15]	Serial	0.63	0.33
PBA [16]	Serial	0.62	0.30
ABC-PSO	Serial	0.28	0.15
IDCS [19]	Serial	0.44	0.25
TLBO-2pt–This study	Serial	0.86	0.54

The results also show that a hybrid approach consisting of the best features of various soft computing and nature inspired algorithms provides a better standard deviation from the optimal values. Hence, a hybrid approach with well-known Nature Inspired Swarm Intelligence techniques can be a way of approaching this problem in the future.

This approach can be extended towards other variants of the RCPSP problem like multi-mode or multi-skill situations, pre-emptive precedence or stochastic time durations by way of future study. These variants of TLBO can also be used to solve other hard Combinatorial Optimization Problems. Hybrid approaches combining the ease of TLBO with other operations of Genetic Algorithms like mutation and elitism can also be investigated in future.

REFERENCES

[1] Vanhoucke, M. 2012. Project Management with Dynamic Scheduling, Springer-Verlag Berlin Heidelberg.

[2] Merkle, D., Schmeck, H., and Middendorf, M. 2002. Ant Colony Optimization for Resource-Constrained Project Scheduling, *IEEE Trans. Evol. Comp*. 6(4). https://doi.org/10.1109/TEVC.2002.802450

[3] Blazewicz, J., Lenstra, J.K., and Rinnooy, K.A.H.G. 1983. Scheduling subject to resource constraints: classification and complexity. *Discret. Appl. Math*. 5(1): 11–24

[4] Rao R.K., Savsani V.J., and Vakharia D.P. 2011. Teaching-learning-based optimization: A novel method for constrained mechanical design optimization problems. *CAD Computer Aided Design*, 43(3): 303-315. https://doi.org/10.1016/j.cad.2010.12.015

[5] Hartmann, S., and Kolisch, R. 2000. Experimental evaluation of state-of-the-art heuristics for the resource-constrained project scheduling problem. *Euro. Jour. Opr. Research*, 127(2): 394-407.

[6] Kolisch, R., and Hartmann, S. 1999. Heuristic algorithms for solving the resource- constrained project scheduling problem: classification and computational analysis, in: J. Weglarz (Ed.), Project Scheduling: Recent Models, Algorithms and Applications, Kluwer Academic Publishers, Berlin, 147–178.

[7] Kolisch, R., and Hartmann, S. 2006. Experimental investigation of heuristics for resource- constrained project scheduling: an update. *Euro. Jour. Opr. Research*, 174: 23–37.

[8] Kolisch, R., and Padman, R. 2001. An integrated survey of deterministic project scheduling. *Omega*, 29: 249–272.

[9] Sprecher, A., and Kolisch, R. 1997. PSPLIB-a project scheduling problem library: OR software-ORSEP operations research software exchange program. *Euro. Jour. Opr. Research*. 96(1): 205-216.

[10] Zhou, Y., Guo, Q., and Gan, R. 2009. Improved ACO algorithm for resource-constrained project scheduling problem. International Conference on Artificial Intelligence and Computational Intelligence. 3: 358-365. https://doi.org/10.1109/AICI.2009.461

[11] Gonzalez-Pardo, A., Ser, J.D., and Camacho, D. 2017. Comparative study of pheromone control heuristics in ACO algorithms for solving RCPSP problems. *Applied Soft Computing Journal*. 60: 241-255. *http://dx.doi.org/10.1016/j.asoc.2017.06.042*

[12] Zhang, H., Li, X., Li, H., and Huang, F. 2006. Particle swarm optimization-based schemes for resource-constrained project scheduling. *Automation in Construction.* 14(3): 393-404.

[13] Kumar, N., and Vidyarthi, D.P. 2015. A model for resource-constrained project scheduling using adaptive PSO. *Soft Computing: Methodologies and Application*. 20(4): 1565-1580.

[14] Kasravi, M., Mahmoudi, A., and Feylizadeh, M.R. 2019. A novel algorithm for solving resource-constrained project scheduling problems: a case study. *Journal of Advances in Management Research*. 16(2): 194-215.

[15] Ziarati, K.,Akbari, R., and Zeighami, V. 2011. On the performance of bee algorithms for resource-constrained project scheduling problem.*Applied Soft Computing.* 11(4): 3720-3733. *http://dx.doi.org/10.1016/j.asoc.2011.02.002*

[16] Nemmich, M., Debbat, F., and Slimane, M. 2019. A permutation-based bees algorithm for solving resource-constrained project scheduling problem. *International Journal of Swarm Intelligence Research*. 10(4): 01-24.

[17] Nemmich, M., Debbat, F., and Slimane, M. 2019. An enhanced discrete bees algorithms for resource constrained optimization problems. *Inteligencia Artificial*. 22(64): 123-134.

[18] Jia, Q., and Guo, Y. 2016. Hybridization of ABC and PSO algorithms for improved solutions of RCPSP. *Journal of the Chinese Institute of Engineers*. 39(6): 727-734.

[19] Bibiks, K., Hu, Y., Li, J., Pillai, P., and Smith, A. 2018. Improved discrete cuckoo search for the resource-constrained project scheduling problem. *Applied Soft Computing, Elsevier*. 69: 493-503. *https://doi.org/10.1016/j.asoc.2018.04.047*.

[20] Dang Quoc, H., The, L.N., Doan, C.N., and Thanh, T.P. 2018. New Cuckoo Search Algorithm for the Resource Constrained Project Scheduling Problem. Proceedings - 2020 RIVF International Conference on Computing and Communication Technologies. 16-18. *10.1109/RIVF48685.2020.9140728*

[21] Bibiks, K., Li, J., and Hu, F. 2015. Discrete flower pollination algorithm for resource constrained project scheduling problem. *International Journal of Computer Science and Information Security*. 13(7): 08-19. *10.1016/j.ins.2014.02.056*

[22] Wu, Y., Wang,X., Li, G., and Lu, A. (2019). Brain Storm Optimization Algorithm based on adaptive inertial selection strategy for the RCPSP. *InProc* - 2019 Chinese Automation Congress, CAC 2019. 2610-2615. *10.1109/CAC48633.2019.8996409*

[23] Roy B., and Sen A.K. 2020. A Novel Metaheuristic Approach for Resource Constrained Project Scheduling Problem. In: Pant M., Kumar Sharma T., Arya R., Sahana B., Zolfagharinia H. (eds) Soft Computing: Theories and Applications. Advances in Intelligent Systems and Computing, vol 1154. Springer, Singapore. https://doi.org/10.1007/978-981-15-4032-5_49

[24] Baykasoglu A, Hamzadayi A, and Kose S.Y. 2014. Testing the performance of teaching learning-based optimization (TLBO) algorithm on combinatorial problems: Flow shop and job shop scheduling cases. *Information Sciences*, 276: 204-218.

[25] Ji, X., Zhou, J., Yin, Y., and Shen, X. 2017. An improved teaching-learning-based optimization algorithm and its application to a combinatorial optimization problem in foundry industry. *Applied Soft Computing.* 57: 504-516. *10.1016/j.asoc.2017.04.029*

[26] Saharan, S., Lather J.S., & Radhakrishnan, R. 2017. Combinatorial problem optimization using TLBO. *InProc:* IEEE International Conference on Signal Processing, Computing and Control. 559-563. *10.1109/ISPCC.2017.8269741*

[27] Kolisch, R. 1996. Serial and parallel resource-constrained project scheduling methods revisited: Theory and computation. *European Journal of Operations Research.* 90(2): 320-333.

[28] Hartmann, S. 1998. A competitive genetic algorithm for resource-constrained project scheduling problem. *Naval Research Logistics.* 45(7): 733-750.