International Journal of Civil Engineering

Construction Management

Genetic algorithms to solve resource-constrained project scheduling problems with variable activity durations

M. H. Sebt[1,*], M. H. Fazel Zarandi[2], Y. Alipouri [2]

Received: October 2012, Accepted: January 2013

Abstract

Resource-Constrained Project Scheduling Problem (RCPSP) is one of the most popular problems in the scheduling phase of any project. This paper tackles the RCPSP in which activity durations can vary within their certain ranges such asRCPSP with variable activity durations. In this paper, we have attempted to find the most suitable hybridization of GA variants to solve the mentioned problem.For this reason, three GA variants (Standard GA, Stud GA and Jumping Gene) were utilized for first GA, and two GA variants (Standard GA, Stud GA) for the second one, and their hybridizations were compared. For this purpose, several comparisons of the following hybridizations of GAs are performed: Standard-Standard GA, Standard-Stud GA, Stud-Standard GA, Stud-Stud GA, Jumping Gene-Standard GA, and Jumping Gene-Stud GA. Simulation results show that implementing Stud-Stud GA hybridization to solve thisproblem will cause convergence on the minimum project makespan, faster and more accurate than other hybrids. The robustness of the Stud GA in solvingthe well-known benchmarking RCPSP problems with deterministic activity durations is also analyzed.

Keywords: *Project scheduling, RCPSP with variable activity durations, Standard GA, Stud GA, Jumping gene.*

1. Introduction

One of the most popular problems in the scheduling phase of any project is Resource-Constrained Project Scheduling Problem (RCPSP) that has a wide range of applications in industry, business, and government etc.

Resource-Constrained Project Scheduling has been an interesting research topic for many decades, resulting in a wide variety of optimization procedures. The focus on project lead-time minimization has led to the development of various exact and (meta-) heuristic procedures for scheduling projects with tight resource constraints under a wide variety of assumptions. The basic problem type in project scheduling is the well-known Resource-Constrained Project Scheduling Problem (RCPSP). This problem type aims at minimizing the total duration or makespan of a project subject to precedence relations between activities and limited renewable resource availabilities, and is known to be NP-hard [1].

In literature, some exact algorithms have been used to solve RCPSP. Some of these exact methods are those of Demeulemeester and Herroelen [2]and Sprecher [3]. However, since computation times of exact methods are too large, researchers have used heuristic procedures to obtain reasonable project schedules within short computation times. The following studies presented these procedures: [4, 5].

Some metaheuristic algorithms are also proposed for RCPSP. Genetic Algorithms (GAs), Simulated Annealing (SA), Tabu Search (TS), and Particle Swarm Optimization (PSO) are some of these algorithms. Methods based on genetic algorithms are presented in [6, 7-13]. Simulated annealing is incorporated in [6, 14-18], Tabu Search in [6, 19-24], and Particle Swarm Optimization in [25, 26]. Other studies haveincorporated the Frog Leaping Algorithm (FLA) [27], Evolutionary Programming (EP) [28], and Hybrid Algorithms and so on to solve the RCPSP. In all of these studies, activity durations and resource requirements are assumed deterministic.

This research deals with the RCPSP in which activity durations can vary within their certain ranges (i.e., RCPSP with variable activity durations). Long and Ohsato [29] introduced a hybrid genetic algorithm to solve this problem. However, they only made use of the Standard GA in their

* *Corresponding Author: sebt@aut.ac.ir*
1 Associate Professor of Construction Engineering and Management, Department of Civil Engineering, Amirkabir University of Technology, Tehran, Iran
2 Professor of Industrial Engineering, Department of Industrial Engineering, Amirkabir University of Technology, 15875 Tehran, Iran
3 Ph.D Candidate of Construction Engineering and Management, Department of Civil Engineering, Amirkabir University of Technology, Tehran, Iran

proposed procedure. It will be shown that if other variants of GA are used, the resulting hybridizations will find a better solution (minimum makespan) for the problem and converge to this solution faster with higher accuracy.

Two genetic algorithms are used in the proposed hybridization procedure of Long and Ohsato [29]. The first GA works on the durations of activities, and each chromosome of this first GA is a set of activity durations. The second GA receives a set of activity durations from the first GA to find minimum project makespan for this set, as fitness value. As the main objective of this paper is to find suitable GA variants for the first and second GAs, three variants of GA, namely Standard GA, Stud GA and Jumping Gene are used and the following hybridizations of GAs are compared: Standard-Standard GA, Standard-Stud GA, Stud-Standard GA, Stud-Stud GA, Jumping Gene-Standard GA, and Jumping Gene-Stud GA. The first GA (e.g. Standard GA in Standard-Stud GA) is implemented as a binary genetic algorithm which represents variables as an encoded binary string, and works with the binary strings to minimize the cost.However, genes of each chromosome in the second GA are taken to be integer numbers that show activity numbers.

The remainder of the paper is organized as follows: In the next Section, a background is presented to describe variants of GA. Section 3 describes the general formulation of the RCPSP problem with variable activity durations. Section 4 presents the procedure of using genetic algorithms to solve the problem. In Section 5, results of computational experiments on some example networks are reported, and the comparison between hybridizations of GAs are made. Moreover, Section 5 is devoted to analyzingthe robustness of the best variant of GA to solve the RCPSP with deterministic activity durations by applying it on some well-known benchmark problems. Finally, conclusions and future works are presented in Section 6.

2. Background

The standard genetic algorithm has been introduced by Holland (1975) and attemptsto implement the idea of survival of the fittest in the field of combinatorial optimization [30]. A genetic algorithm starts with a population of n chromosomes. Then, chromosomes are ranked from the lowest cost to the highest by evaluating the cost function. Only the best ones are selected to continue, while the rest are deleted. Two chromosomes are selected from the pool of chromosomes to mate. Mating is the creation of one or more offsprings from the parents selected in the pairing process. Crossover and mutation operators perform this mating. Pairing takes place in the mating population until offsprings are born to replace the discarded chromosomes. This process continues until a number of populations have been created and evaluated, or stopping criteria have been reached.

Khatib and Fleming [31] introduced the Stud GA variant of GA. The difference between this variant of GA and Standard GA is in the selection of father and mother chromosomes. The basic idea behind the Stud GA is to use the best individual in the population (father) to mate with all others (mothers) to produce the new offsprings. No stochastic selection is used here [31].

The procedure of Stud GA is as follows [31]:

1. Initialize a random population;
2. Choose the fittest individual for mating (the Stud);
3. Perform crossover between the Stud and the remaining elements, and
4. Repeat until stopping criteria is met.

The Nobel Laureate, McClintock, based on her work on corn plants [32], reported the very first discovery of Jumping Gene (JG). To emulate the analogy for JG computation, a transposition of gene(s) into the same chromosome or even to other chromosomes is devised for computation. As a result, this operation can further enhance genetic operations such ascrossover, mutation, and selection for improving the fitness quality of chromosomes from one generation to the other[33].

Computational JG operations have recently been proposed for the enhancement of searching ability in evolutionary algorithms, in particular, for multi-objective optimization problems. The following issues are consideredin the design and implementation of the JG [33]:

(a) Each chromosome has some consecutive genes thatare randomly selected as the transposon. There may be more than one transposon with different lengths (e.g. one or two bits for binary code).

(b) The jumped locations of the transposons are randomly assigned, and operations can be carried out on the same chromosome, or to another chromosome in the population pool.

(c) Two JG operations, namely cut-and-paste and copy-and-paste, are devised. As seen in Figure 1, these two operations are to be inserted after the parent selection process.

(d) The JG operations are not limited to binary encoded chromosomes, but can also be extended for other coding methods, such as integer or real number.

The Stud GA and Jumping Gene have proven their ability to solve optimization problems, and both of them were better than Standard GA in most cases. This fact inspired us to apply these variants of GA on solving the RCPSP with variable activity durations and to find their suitable hybridization for this problem.

This background will be helpful in understanding the procedure used in this paper to solve the RCPSP with variable activity durations.

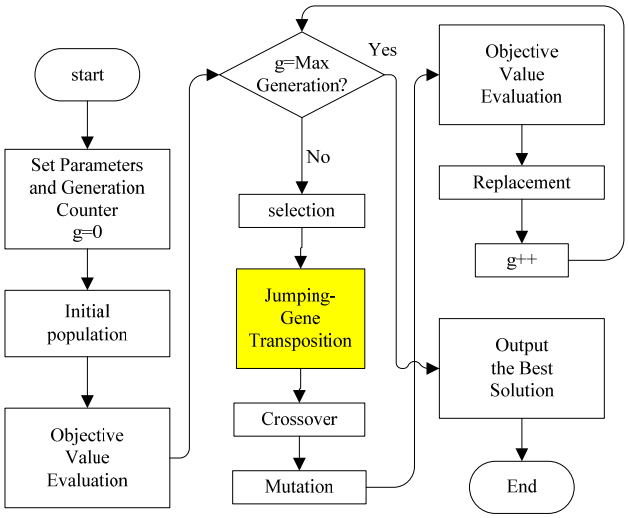

Fig. 1 Genetic cycle of JGEA [33]

3. Resource-constrained project scheduling problem with variable activity durations

In the classical RCPS problem, duration and resource requirements of each activity are considered to be known and deterministic, while in most real world projects, duration of the activities depends on time/resource trade-off (e.g. the total work content of 120 man-days may be performed in 12 days by 10 men, or in 6 days by 20 men) and cannot be determined precisely. Therefore, it seems necessary to determine a suitable duration for each activity of a project as the project duration is minimized. For that reason, this paper deals with the RCPSP with variable activity times.In other words, the problem is to find the deterministic schedule thathas suitable duration and optimal start times of activities, so that the project duration under precedence and resource constraints minimized. Using the classification scheme in Demeulemeester et al. [30], the RCPSP with variable activity durations can be denoted as problem m,1T/cpm,cont,mu/Cmax [29].

The mathematical description of the problem is given as follows [29]:

$$\text{Min } T \tag{1}$$

Subject to

$$T = max\ (ft(j) = max\ (st(j) + t(j)),\ j = 1,\ldots,N \tag{2}$$

$$mint(j) \le t(j) \le maxt(j),\quad j = 1,\ldots,N \tag{3}$$

$$st(i) + t(i) \le st(j)\quad j = 1,\ldots,N \text{ and } \forall i \in P(j) \tag{4}$$

$$\sum i \in S_t r_{ik} \le ak \text{ for } k = 1,\ldots,R \text{ and } t = 1,\ldots,ft(N) \tag{5}$$

$$Est(j) \le st(j) \le Lst(j),\quad j = 1,\ldots,N \tag{6}$$

Variables $st(j)$, $t(j)$, $ft(j)$, $j=1,\ldots,N$ are non-negative integer numbers.

In this formulation, the variables $ft(j)$ and $st(j)$ denote the finish and start times of the different activities, respectively, while $t(j)$ denotes the duration of each activity. The set S_t that is used in equation (5) denotes the set of activities that are in progress at time t. Moreover, a_k denotes the availability of the k^{th} resource type and r_{ik} represents the resource requirement of activities in S_t for resource type k. The interval $[mint(j), maxt(j)]$ is the certain range of duration of activity j, and N is the total number of activities. $P(j)$ is a set of immediate predecessors of activity j and R denotes the number of resources. $Est(j)$ and $Lst(j)$ denote the earliest and latest start of activity j by the standard CPM, respectively.

Objective function (1) minimizes the project duration (T). Equation (2) is used to determine the project duration. Constraints (3) represent that activities have variable durations. Constraints (4) are to implement the precedence constraints, while constraints (5) are to implement resource constraints, and finally Constraints (6) force the start time of activity jto be between the earliest and latest start time of this activity.

4. Solution methodology by using genetic algorithm variants

This section introduces the methodology used in this paper to solve RCPSP with variable activity durations, which is the special case of the Discrete Time/Resource Trade-off Problem (DTRTP).

Literature on the time/resource trade-off problem is relatively sparse [30], and all of these existing methods solve RCPSP with the Discrete Time/Resource Trade-off Problem (DTRTP) [29]. Long and Ohsato [29] developed a procedure to solve continuous time/resource trade-off, especially to solve the RCPSP with variable activity durations. This procedure has been used to hybridize variants of GA to solve RCPSP with variable activity durations as follows:

The procedure implements a GA on the activity durations and evaluates the fitness of a given set of durations (which amounts to the classical RCPSP problem with single execution activity mode-denoted as m,1/cpm/Cmax [30]) by means of well-known scheduling heuristics [29]. GA is also used as the scheduling heuristic; thus, two genetic algorithms are used in the procedure. The GA thatworks on the activity durations is called first GA, and the GA thatis used as scheduling heuristic is called second GA. The goal is to find the most suitable variants of GA for these first and second GAs; therefore, three GA variants (Standard GA, Stud GA, and Jumping Gene) are utilized for the first GA, and two GA variants (Standard GA and Stud GA) for the second one and their hybridizations are compared. These hybridizations are the following six tries:

- Standard GA with Standard GA
- Standard GA with Stud GA
- Stud GA with Standard GA
- Stud GA with Stud GA
- Jumping Gene with Standard GA
- Jumping Gene with Stud GA

The comparative results of these hybridizations on some example projects exist in the Section 5.

4.1. Functions of the first ga on activity durations

Binary GA is used to implement GA on the activity durations. In binary GA, each gene of the chromosome is a binary string and shows the duration of one activity.Therefore, the genes in each chromosome of this first GA represent durations $t(j)$ of N activities.The encoding and decoding formulas in [34] are used for binary encoding and decoding of mathematical formulas.

In the iterative process of the first GA, new chromosomes are produced by crossover and mutation operators (For JG, also, cut-and-paste and copy-and-paste operators are used). Genes of chromosomes are decoded to determine durations $t(j)$ in the range $[mint(j), maxt(j)]$ and these $t(j)$, $j=1,\ldots,N$ are utilized by second GA to calculate fitness function. In fact, the fitness value of a given set of durations is considered as the project duration and is calculated by the second GA. This process continues until a number of populations have been created and evaluated, or stopping criteria have been reached. Figure 2 shows the aforementioned procedure for the first GA.

4.2. Implementing the second GA in the procedure

The second GA is utilized to find the optimal value of project duration for each set of activity durations. This second GA receives a set of activity durations from first GA and finds the minimum project makespan as fitness valueof this set.

In this research, the evaluation process of fitness function (as

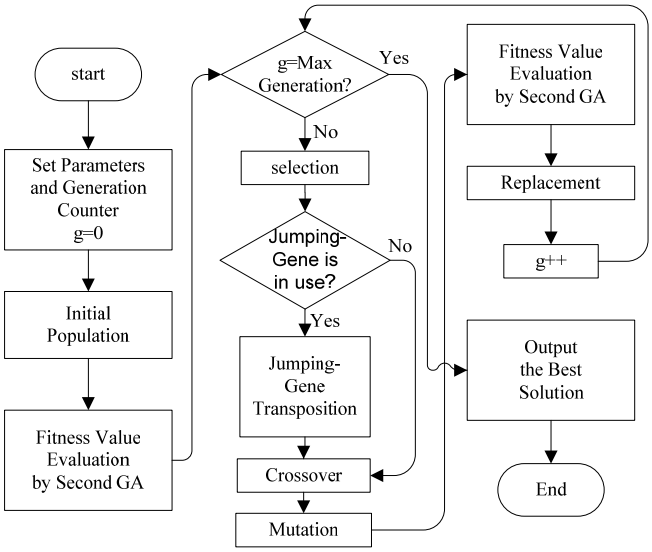

Fig. 2 Flowchart of implementing the first GA

project duration *T*) is implemented by an effective application of well-known heuristic priority rules in the serial scheme. Here, the parallel scheme was not used, because in [35, 36] researchers show that the parallel method does not generally perform better than the serial method.

The serial scheduling scheme sequentially adds activities to the schedule until a feasible complete schedule is obtained. In each iteration, the next activity in the priority list is chosen and for that activity, the first possible starting time is assigned such that no precedence or resource constraint is violated [30].

In this paper, priority list representation is used as the representation scheme for second GA's chromosomes, and the one-point crossover operator is taken to be the binary neighborhood operator implemented together with the adjacent pairwise interchange mutation operator to obtain the near optimal solution in the second GA. Figure 3 shows the aforementioned procedure for the second GA.

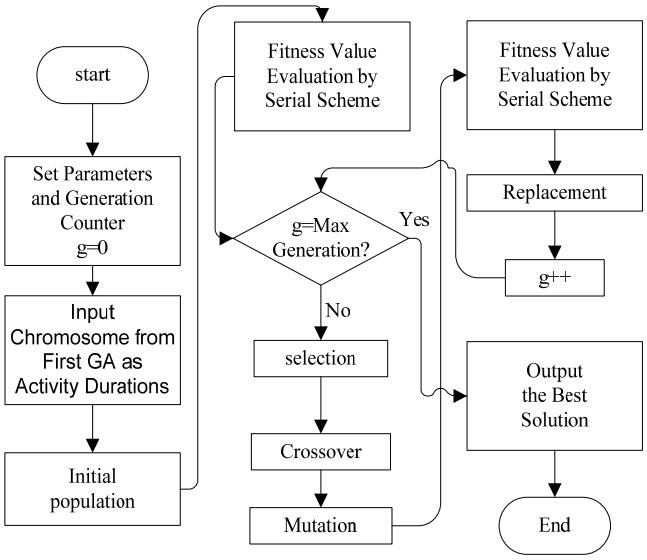

Fig. 3 Flowchart of implementing the second GA

5. Simulation comparisons and results

This section presents the computational analysis for investigating performance of the six proposed hybridizations of GA variants to solve the RCPSP with variable activity durations. The efficacy and robustness of the most suitable variant of GA to solve the well-known benchmarking RCPSP problems with deterministic activity durations is also analyzed. The algorithms were programmed using the MATLABprogramming language, and tests were carried out on a laptop with Windows 7 Operation System, Intel Core 2 Duo CPU at 2.00 GHzclock speed. The following computational experiments were implemented: (1) Performance comparison of six hybridizations of GA variants to find the most suitable hybridization of the GA variants for solvingthe RCPSP with variable activity durations, and (2) Performance comparison of the most suitable variant of GA from part 1 with other metaheuristics to solve the RCPSP with deterministic activity durations.

5.1. Comparison of six hybridizations of GA variants

In order to investigate the performance of the six tries of GA variants mentioned in Section 4, two experimental analyses are presented including their comparison using the same problem example as [29] (part A),and their comparison using twenty instanceproblems from J30 standardinstances set in PSPLIB (part B).

A. Using the same problem example as [29] for comparison

The activity network of the example usedby Long and Ohsato [29] is shown in Figure 4.

Figure 4. Activity network of the example project [29]

As observed in Figure 4, this example is a project with 20 activities. The data foreach activity and the relationship between activities are shown in Table 1. In the last column of Table 1, R_{jk} denotes the total required resource k to perform activity j. The total availability of resource (a_k) is 45 units per day.

The procedure mentioned in the previous section was used with the following input data for the first GA of all the hybridizations: as reference [29], the number of chromosomes (or particles) in each population was taken to be 100; However, the mutation factor (the fraction of the bits thatgets mutated) was chosen to be 0.1 instead of 0.03 in [29] (because increasing the number of mutation increases the algorithm's freedom to search outside the current region of variable space and tends to distract the algorithm from converging on a

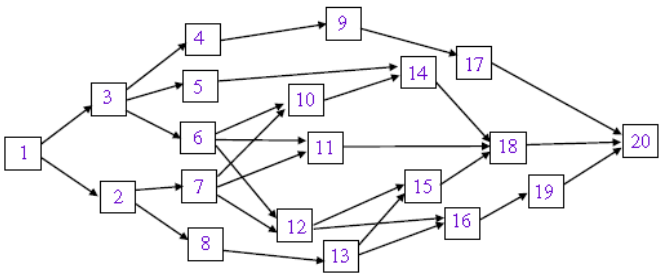

Fig. 4 Activity network of the example project [29]

popular solution [34, pp.43].Furthermore, our considerations showed that 0.1 gives better answers for this problem than other values).The crossover rate was taken to be 0.5 (because, crossover rate is often kept 0.5 in the natural selection process [34, pp.38]).Each gene was coded with 16 bits.Finally, generation number was equaled to 100. These simulation parameters are presented in Table 2.

For the second GA in the hybridization of two GA variants, the number of chromosomes (priority lists) in the population was chosen to be 10, and these chromosomes were produced by the following 9 priority rules: random rule (randomly produced two priority lists), earliest start time rule, earliest finish time rule, latest start time rule, latest finish time rule, minimum total slack rule, minimum free slack rule, minimum safety slack rule, and greatest resource demand rule. The upper bound for project time wasassumed to be 70 days.

Using the aforementioned procedure and input data, six proposed hybridizations of GA variants as mentioned in Section 4 were tested on an example network in Figure 4. All of these six tries were repeated 5 times on the example network and were run until the pre-specified generation 100 was reached. Since the random development of the algorithms causes them to have different answers in different runs, the5-time repetition was carried outfor reducing the effect of chance and increasing reliability in the results. Results are shown in Figures 5-10 and Tables 3-8.

Figures 5-10 show the mean of the results of 5 times running of the algorithms. Each figure shows the minimum value of cost function (second GA answer) versus the number of iterations. The curve in each figure gives a measurement of how fast the corresponding algorithm converges and reaches the global minimum. Moreover, it is a suitable way to use these figures for comparing different algorithms in terms of the final value they reach for a specified number of iterations.

Figure 5 is for Standard-Standard GAs hybridization and its mean value of fivetimes running in the last iteration is 58.2. Likewise, Figures 6, 7, 8, 9 and 10 are for Standard-Stud, JG-Standard, JG-Stud, Stud-Standard, and Stud-Stud GAs hybridizations, respectively; and, with the same order, their mean value of 5 times running in the last iteration are 57.8, 58.2, 58.4, 57.2, 57.2. Since among these mean values 57.2 is

Table 1 Data for activities in the example pr

Activity number	*mint(j)* (days)	*maxt(j)* (days)	predecessors	R_{jk} (units-days)
1	4	6		150
2	4	6	1	140
3	4	5	1	160
4	3	5	3	60
5	2	4	3	30
6	6	9	3	90
7	6	10	2	90
8	5	8	2	40
9	3	5	4	50
10	7	10	6,7	120
11	5	8	6,7	100
12	3	6	6,7	50
13	8	10	8	200
14	4	7	5,10	100
15	3	8	12,13	60
16	5	8	12,13	180
17	3	4	9	60
18	13	15	11,14,15	240
19	5	6	16	150
20	7	8	17,18,19	180

Table 2 Simulation parameters for first GA

Population size	100
Mutation rate	0.1
Crossover rate	0.5
Number of bits in each gene	16
Number of generation	100
Number of repetition	5
Range bound of activity durations	Mentioned in second and third column of Table 1

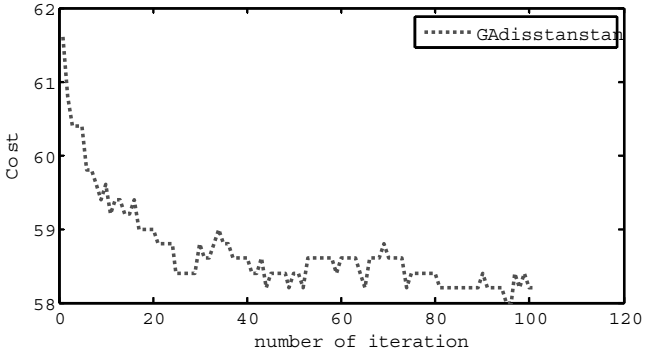

Fig. 5 Plot of the minimum cost as a function of the iteration for Standard-Standard GAs hybridization

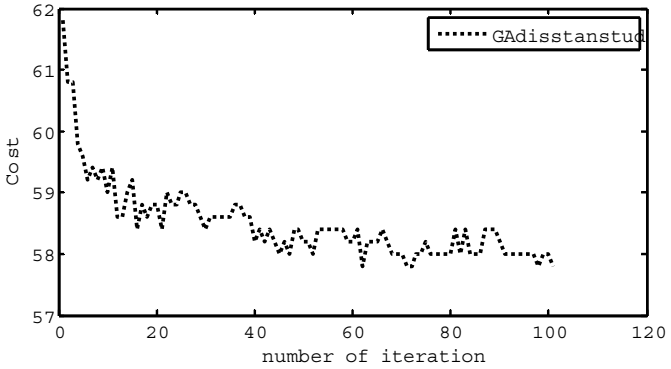

Fig. 6 Plot of the minimum cost as a function of theiteration for Standard-Stud GAs hybridization

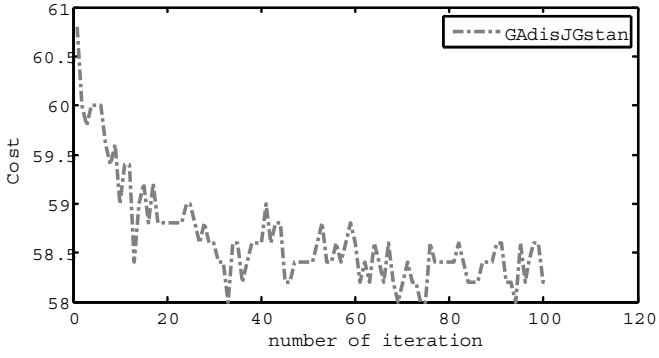

Fig. 7 Plot of the minimum cost as a function of the iteration for JG-Standard GAs hybridization

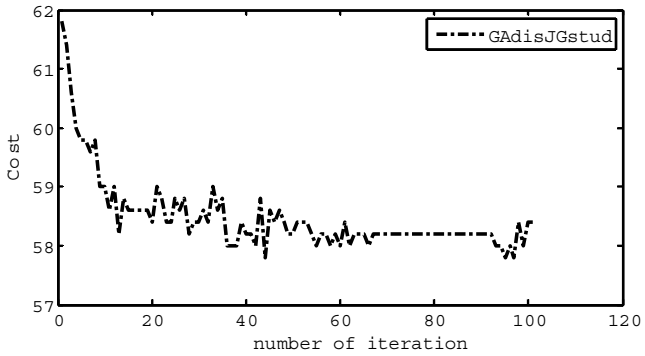

Fig. 8 Plot of the minimum cost as a function of theiteration for JG-Stud GAs hybridization

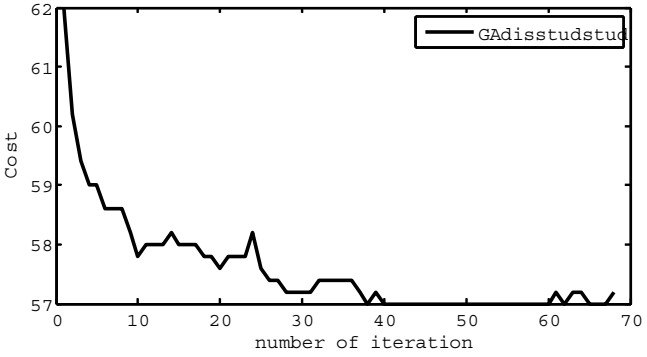

Fig. 10 Plot of the minimum cost as a function of theiteration for Stud-Stud GAs hybridization

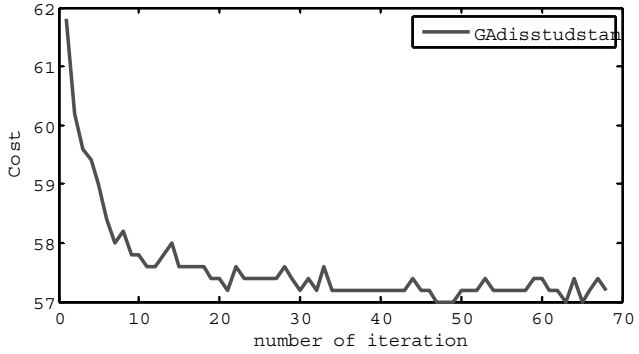

Fig. 9 Plot of the minimum cost as a function of the iteration for Stud-Standard GAs hybridization

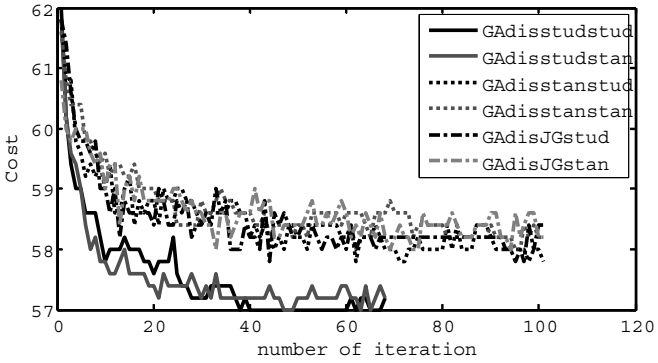

Fig. 11 Plot of the minimum cost as a function of the iteration for all the hybridizations

Table 3 Results for activities using Standard-Standard GAs

Activity number	*t(j)* (days)	*st(j)* (days)	*ft(j)* (days)	r_{jk} (units/day)
1	4	0	4	37.5
2	5	8	13	28
3	4	4	8	40
4	5	11	16	12
5	3	8	11	10
6	7	13	20	12.86
7	7	13	20	12.86
8	6	13	18	6.67
9	4	16	20	12.5
10	8	20	28	15
11	8	28	36	12.5
12	5	20	25	10
13	8	25	33	25
14	5	33	38	20
15	5	33	38	12
16	7	38	45	25.7
17	3	20	23	20
18	13	38	51	18.46
19	6	45	51	25
20	7	51	**58**	25.7

Table 4 Results for activities using Standard-Stud GAs

Activity number	*t(j)* (days)	*st(j)* (days)	*ft(j)* (days)	r_{jk} (units/day)
1	4	0	4	37.5
2	5	8	13	28
3	4	4	8	40
4	5	13	18	12
5	2	8	10	15
6	7	10	17	12.86
7	9	13	22	10
8	5	13	18	8
9	5	18	23	10
10	7	26	33	17.15
11	8	29	37	12.5
12	3	23	26	16.67
13	8	18	26	25
14	4	33	37	25
15	4	29	33	15
16	7	37	44	25.7
17	3	26	29	20
18	13	37	50	18.46
19	6	44	50	25
20	7	50	**57**	25.7

the lowest value, the convergence speed of the Stud-Standard and Stud-Stud GAs hybridizations are more than the other tries.

By placingFigures 5-10 in a single figure as shown in Figure 11, it can be observed schematically that the above claim is true. As can be seenin Figure 11, in all the iterations, curves related to hybridizations of Stud-Standard and Stud-Stud GAs have fewer mean values than others; therefore, these two tries are faster than others to converge and reach the minimum point searched as the minimum project duration.

The suitable duration ($t(j)$), the daily required resource k (r_{jk}), and the start and finish times ($st(j)$ and $ft(j)$) of activity j, as the test results of six mentioned tries are shown in Tables 3-8. Table 3 is for Standard-Standard GAs hybridization, and its minimum makespan is equal to 58 (bold number in the last row

Table 5 Results for activities using JG-Standard GAs

Activity number	*t(j)* (days)	*st(j)* (days)	*ft(j)* (days)	r_{jk} (units/day)
1	4	0	4	37.5
2	4	8	12	35
3	4	4	8	40
4	5	12	17	12
5	3	17	20	10
6	8	12	20	11.25
7	6	12	18	15
8	7	12	19	5.71
9	4	18	22	12.5
10	7	22	29	17.15
11	6	28	34	16.67
12	5	28	33	10
13	8	20	28	25
14	4	33	37	25
15	3	34	37	20
16	7	37	44	25.7
17	4	29	33	15
18	13	37	50	18.46
19	6	44	50	25
20	7	50	**57**	25.7

Table 6 Results for activities using JG-Stud GAs

Activity number	*t(j)* (days)	*st(j)* (days)	*ft(j)* (days)	r_{jk} (units/day)
1	4	0	4	37.5
2	5	8	13	28
3	4	4	8	40
4	4	8	12	15
5	4	12	16	7.5
6	7	13	20	12.86
7	6	13	19	15
8	7	13	20	5.71
9	5	16	21	10
10	8	20	28	15
11	6	20	26	16.67
12	4	21	25	12.5
13	8	26	34	25
14	5	31	36	20
15	3	34	37	20
16	7	37	44	25.7
17	3	28	31	20
18	13	37	50	18.46
19	6	44	50	25
20	7	50	**57**	25.7

Table 7 Results for activities using Stud-Standard Gas

Activity number	*t(j)* (days)	*st(j)* (days)	*ft(j)* (days)	r_{jk} (units/day)
1	4	0	4	37.5
2	5	8	13	28
3	4	4	8	40
4	4	8	12	15
5	4	12	16	7.5
6	7	13	20	12.86
7	7	13	20	12.86
8	6	13	19	6.67
9	4	16	20	12.5
10	7	20	27	17.15
11	6	27	33	16.67
12	4	28	32	12.5
13	8	20	28	25
14	4	32	36	25
15	3	33	36	20
16	7	36	43	25.7
17	4	28	32	15
18	13	36	49	18.46
19	6	43	49	25
20	7	49	**56**	25.7

Table 8 Results for activities using Stud-Stud Gas

Activity number	*t(j)* (days)	*st(j)* (days)	*ft(j)* (days)	r_{jk} (units/day)
1	4	0	4	37.5
2	6	8	14	23.3
3	4	4	8	40
4	3	10	13	20
5	2	8	10	15
6	8	13	21	11.25
7	7	14	21	12.86
8	7	14	21	5.7
9	5	13	18	10
10	8	22	30	15
11	7	30	37	14.29
12	5	22	27	10
13	10	21	31	20
14	6	31	37	16.67
15	5	31	36	12
16	7	37	44	25.7
17	4	18	22	15
18	13	37	50	18.46
19	6	44	50	25
20	7	50	**57**	25.7

of column 4). Likewise, Tables 4, 5, 6, 7 and 8 are for Standard-Stud, JG-Standard, JG-Stud, Stud-Standard, and Stud-Stud GAs hybridizations, respectively; and, with the same order, their minimum makespans are 57, 57, 57, 56 and 57. These minimum makespans with the standard deviations of all the six tries are shown in Table 9.

Table 9 Minimum makespan and standard deviation of all six tries

Parameter	Algorithm type					
	Standard-Standard	Standard-Stud	JG-Standard	JG-Stud	Stud-Standard	Stud-Stud
Minimum makespan (days)	58	57	57	57	56	57
Standard deviation	0.43	0.49	0.49	0.40	0.40	0.00

Using Table 9, the accuracy of the six mentioned tries can be discussed. As it is observed in this table, the lower standard deviation in the table is 0.00 and is for the Stud-Stud GAs hybridization. The lower standard deviation corresponds to higher accuracy. The lower minimum makespan is for Stud-Standard GAs hybridization, but because of its lower accuracy (standard deviation = 0.40), repetition of its minimum makespan cannot be expected. For Stud-Stud GAs hybridization, the accuracy is high (standard deviation = 0.00), and the minimum makespan is reasonable.

Now it can be claimed that the Stud-Stud GAs hybridization is the most suitable hybridization of GAs variants for solving the RCPSP with variable activity durations; because, on one hand, as it was proven using Figures 5-11, the convergence speed of the Stud-Standard and Stud-Stud GAs hybridizations are more than the other tries, and on the other hand, as it was shown in Table 9, Stud-Stud GAs hybridization is more accurate than Stud-Standard GAs hybridization. Comparing six proposed hybridizations of GAs onlyby applying them on the example of Long and Ohsato [29] is not enough to make a strong decision about the most suitable hybridization of GAs, therefore, the analysis of part B is required.

B. Using twenty instance problems from PSPLIB for comparison

There are benchmark problems for RCPSP with deterministic activity times, but there is no benchmark problem set for RCPSP with variable activity times. Thus, twenty instance problems are derived from J30 standard instances set in PSPLIB benchmark problems as the base and problems with variable activity times is generated as follows: for each selected instance from PSPLIB, considering b equal to deterministic estimate, the most optimistic time (a) is drawn randomly from interval [b-0.3b, b], and the most pessimistic time (c) is drawn randomly from interval [b, b+0.4b]. The floor function is used to come up with an integer for the optimistic estimate, and ceiling function to come up with an integer for the pessimistic estimate. The lower value (a) is bounded by 1, which means that the most optimistic time required to complete an activity is at least 1 time unit.

Standard instances of PSPLIB are available at http://129.187.106.231/datasm.html considering a single mode resource constrained project scheduling problem, and are generated by the problem generator ProGen using three parameters: Network complexity (average number of non-redundant arcs per node including dummy activity), Resource Factor (average portion of resource of a particular type used and consumed) and Resource Strength (availability of a particular type of resource). In this paper, all20 problems have Network Complexity as 1.5, Resource Factor as 0.25 and Resource Strength as 0.2 for the first 10 instances, and 0.5 for the next 10 instances. In all of these 20 selected problems, there are 30 non-dummy activities and 2 dummy activities.

Results of applying six proposed hybridizations of GA variants using the procedure of Section 4 with the input data of part A to solve the 20 generated problems with variable activity times are shown in Table 10.

Despite in J30 the optimal makespans for all instances are known, it would not be useful to compare the solution obtained for problems with variable activity times to only the optimal solutions for problems with deterministic activity times. Therefore, the lower bound for the 20 selected problems is calculated and the results obtained by the six proposed GA hybridizations are compared to the lower bounds. The lower bounds are calculated by solving the problems by removing their resource constraints. The numbers in the third column of Table 10 show the lower bounds of the problems. Columns 4-9 represent deviation of the results obtained through this research from the lower bound. In Table 10, the best result obtained for each of the twenty generated problems has been

Table 10 Results of six hybridizations of GA variants from solving 20 generated problems

Project No.	Optimal makespans	Lower bound	Difference from l.b.					
			Standard-Standard	Standard-Stud	JG-Standard	JG-Stud	Stud-Standard	Stud-Stud
1	43	30	14	13	13	13	12	**11**
2	47	35	6	5	5	5	**4**	**4**
3	47	36	7	6	6	6	**5**	**5**
4	62	47	9	8	9	8	**7**	**7**
5	39	26	8	8	8	**7**	**7**	**7**
6	48	31	**9**	**9**	**9**	**9**	**9**	**9**
7	60	50	**0**	**0**	**0**	**0**	**0**	**0**
8	53	43	3	3	3	3	**2**	**2**
9	49	35	11	11	11	9	8	**7**
10	45	29	10	9	9	9	**8**	**8**
11	38	28	**4**	**4**	**4**	**4**	**4**	**4**
12	51	38	7	6	6	6	**5**	**5**
13	43	34	2	2	2	2	1	**0**
14	43	36	**1**	**1**	**1**	**1**	**1**	**1**
15	51	41	2	2	1	1	**0**	**0**
16	47	41	2	1	2	1	1	**0**
17	47	39	**0**	**0**	**0**	**0**	**0**	**0**
18	54	40	9	9	8	8	8	**6**
19	54	44	**3**	**3**	**3**	**3**	**3**	**3**
20	43	33	5	4	5	4	4	**3**
Avg. CPU time(s)			7.274	6.056	7.503	6.342	5.973	5.214

highlighted. These results show that, in all of the cases, the Stud-Stud GAs hybridization can perform better than other hybridizations.

Some methods are very capable of reaching their goals but are so time-consuming that they cannot be used in real-world applications. Considering this fact, an attempt is made to compare six proposed GA hybridizations consideringthe aspect of time consumption. The last row of Table 10 shows the average CPU times required for running an iteration of each hybridization with 100 individuals for first GA and 10 individuals for second GA to solve the 20 generated problems with 30 activities. As it can be observed, Stud-Stud GA hybridization requires less CPU time than others do; therefore, it can now be strongly claimed that in RCPSP with variable activity times, Stud-Stud GA hybridization can reach minimums far from other hybridizations in less iterations, thus, it can be considered as possessingmore accuracy, more speed, and less CPU time.

5.2. Comparison of the Stud GA with other methods

In this subsection, Stud GA as the best variant of GA to solve the RCPSP with deterministic activity times is compared with other metaheuristic methods. To illustrate the effectiveness of the Stud GA, J30 project instances from PSPLIB, which consists of 480 projects with 30 activities and 4 resource types, were used. In J30, the optimal makespans for all instances are known. The comparison is carried out in view of the average deviation from the optimal makespan.

The parameters in Table 11 are used for Stud GA. Table 12 displays the results obtained by Stud GA and other methods with 1000 and 5000 evaluated schedules. In Table 12, we present the type of algorithms, the type of schedule-generation scheme used, the authors of each algorithm, and the average deviation from the optimal solution for 1000 and 5000 schedules, respectively. The algorithms are sorted according to descending performance regarding 1000 schedules.

As it is observed in Table 12, Stud GA's average deviation from the optimal solution is 1.14% and 0.89% for 1000 and 5000 schedules, respectively. Stud GA could solve 328 out of 480 problems optimally, while in 93 instances the solution was close to optimum. This means that the Stud GA worked well on 421 out of 480 instances of the J30 set. Meanwhile, the mean CPU-times of the Stud GA for 1000 and 5000 schedules are 3.9 and 18.4 seconds, respectively.

Considering all the results, we can conclude that the Stud GA is competitive with the best ones currently known, having the additional interest of its ease of coding and low computational requirements.

Table 11 Simulation parameters for Stud GA

Population size	10
Mutation rate	0.1
Crossover rate	0.5
Number of generation	100 and 500
Number of repetition	5

6. Conclusions

In this paper, six hybridizations of GA variants were proposed to solve RCPSP with variable activity durations. These six tries were as follows: Standard GA with Standard GA, Standard GA with Stud GA, Stud GA with Standard GA, Stud GA with Stud GA, Jumping Gene with Standard GA, and Jumping Gene with Stud GA.

These hybridizations were compared with each other by applying them on some example projects. Simulation results showed that the hybridization of Stud GA with Stud GA was the most suitable solutionfor solving the RCPSP with variable activity durations; because, this hybridization converged to minimum makespan faster and with higher accuracy than other hybridizations. In other words, implementation of Stud GA as first GA to work on the durations of activities and, also, using Stud GA to find minimum project makespan for each set of activity durations from first GA will cause convergence and reaching the minimum project duration faster than other hybrids. Comparison of the Stud GA with other approaches using theJ30 standard instances in PSPLIB was also implemented.The computational results validate the effectiveness of the Stud GA in solving the RCPSP with deterministic activity times.

Table 12 Average deviations (%) from optimal makespan—ProGen set J=30

Algorithm	SGS	Reference	Schedules	
			1000	5000
GAPS—random key	Param. active	Mendes et al. [13]	0.06	0.02
GA—forw.–backw.	Both	Alcaraz et al. [11]	0.25	0.06
GA—forw.–backw.	Serial	Alcaraz and Maroto [12]	0.33	0.12
GA—FBI	Serial	Valls et al. [7]	0.34	0.20
SFLA-activity list	Serial	Fang and Wang [27]	0.36	0.21
SA—activity list	Serial	Bouleimen and Lecocq [17]	0.38	0.23
TS—activity list	Serial	Nonobe and Ibaraki [19]	0.46	0.16
GA—self-adapting	Serial	Hartmann [10]	0.38	0.22
GA—activity list	Serial	Hartmann [9]	0.54	0.25
PSO—activity list	Serial	Zhang et al. [25]	0.69	0.42
TS—schedule scheme	Related	Baar et al. [24]	0.86	0.44
PSO—activity priorities	Serial	Zhang et al. [25]	0.92	0.61
MCEP—random key	Serial	Sebt et al. [28]	1.02	0.75
GA—random key	Serial	Hartmann [9]	1.03	0.56
Stud GA—priority rule	**Serial**	**This paper**	**1.14**	**0.89**
GA—priority rule	Serial	Hartmann [9]	1.38	1.12
GA—problem space	Mod. par.	Leon and Ramamoorthy [8]	2.08	1.59

This paper has some potential future works:

One of the future prospects of the approaches proposed here is to use other variants of EAs (Evolutionary Algorithms) such asEPs (Evolutionary Programming) variants, and PSO variants etc. to solve the RCPSP with variable activity durations.

Implementation of evolutionary algorithms variants on the other families of RCPSP, such as the Multi-mode RCPSP and Preemptive RCPSP etc. to find the most powerful algorithm as their solution can be another prospect for future studies.

References

[1] Blazewicz, J., Lenstra, J.K. and Rinooy Kan, A.H.G. "Scheduling subject to resource constraints: Classification and complexity", Discrete Applied Mathematics, 5, pp 11–24 (1983).

[2] Demeulemeester, E.L. and Herroelen, W.S. "New benchmark results for the resource-constrained project scheduling problem", Management Science, 43, pp 1485–1492 (1997).

[3] Sprecher, A. "Scheduling resource-constrained projects competitively at modest memory requirements", Management Science, 46, pp 710–723 (2000).

[4] Tormos, P. and Lova, A. "A competitive heuristic solution technique for resource-constrained project scheduling", Annals of Operations Research, 102, pp 65–81 (2001).

[5] Möhring, R., Schulz, A., Stork, F. and Uetz, M. "Solving project scheduling problems by minimum cut computations", Management Science, 49(3), pp 330–350 (2003).

[6] Lee, J.K. and Kim, Y.D. "Search heuristics for resource constrained project scheduling", Journal of the Operational Research Society, 47, pp 678–689 (1996).

[7] Valls, V., Ballestin, F., Quintanilla, M.S. "Justification and RCPSP: a technique that pays', European Journal of Operational Research,165, pp 375–86 (2005).

[8] Leon, V.J. and Ramamoorthy B. "Strength and adaptability of problem-space based neighborhoods for resource-constrained scheduling", OR Spektrum 17, pp 173–182 (1995).

[9] Hartmann, S. "A competitive genetic algorithm for resource-constrained project scheduling", Naval Research Logistics, 45, pp 733–750 (1998).

[10] Hartmann, S. "A self-adapting genetic algorithm for project scheduling under resource constraints", Naval Research Logistics, 49, pp 433–448 (2002).

[11] Alcaraz, J., Maroto, C. and Ruiz, R. "Improving the performance of genetic algorithms for the RCPS problem". Proceedings of the ninth international workshop on project management and scheduling, Nancy France, pp 40–43 (2004).

[12] Alcaraz, J. and Maroto, C. "A robust genetic algorithm for resource allocation in project scheduling", Annals of Operations Research, 102, pp 83–109 (2001).

[13] Mendes, J.J.M., Gonçalves, J.F. and Resende, M.G.C. "A random key based genetic algorithm for the resource constrained project scheduling problem", Computers & Operations Research, 36, pp 92–109 (2009).

[14] Boctor, F.F. "An adaptation of the simulated annealing algorithm for solving resource-constrained project scheduling problem", International Journal of Production Research, 34, pp 2335–2351 (1996).

[15] Cho, J.H. and Kim, Y.D. "A simulated annealing algorithm for resource constrained project scheduling problems", Journal of the Operational Research Society, 48, pp 736–744 (1997).

[16] Bouleimen, K. and Lecocq, H. "A new efficient simulated annealing algorithm for the resource-constrained project scheduling problem", In: G. Barbarosoglu, S. Karabati, L. Özdamar, G. Ulusoy, Eds., Proceedings of the Sixth International Workshop on Project Management and Scheduling, Bogazici University Printing Office, pp 19–22 (1998).

[17] Bouleimen, K. and Lecocq, H. "A new efficient simulated annealing algorithm for the resource-constrained project scheduling problem and its multiple mode version", European Journal of Operational Research, 149, pp 268–281 (2003).

[18] Slowinski, R., Soniewicki, B. and Weglarz, J. "DSS for multiobjective project scheduling", European Journal of Operational Research, 79, pp 220–229 (1994).

[19] Nonobe, K. and Ibaraki, T. "Formulation and tabu search algorithm for the resource constrained project scheduling problem", In: C.C. Ribeiro, P. Hansen, Eds., Essays and Surveys in Metaheuristics, Kluwer Academic Publishers, pp 557–588 (2001).

[20] Valls, V., Quintanilla, S. and Ballestin, F. "Resource-constrained project scheduling—a critical activity reordering heuristic", European Journal of Operational Research, 149, pp 282–301 (2003).

[21] Artigues, C., Michelon, P. and Reusser, S. "Insertion techniques for static and dynamic resource-constrained project scheduling", European Journal of Operational Research, 149, pp 249–267 (2003).

[22] Pinson, E., Prins, C. and Rullier, F. "Using tabu search for solving the resource constrained project scheduling problem", Technical Report, Universite Catholique de l'Ouest, Angers, (1994).

[23] Gagnon, M., Boctor, F.F. and d'Avignon, G. "A tabu search algorithm for the resource-constrained project scheduling problem", In: Proceedings of administrative sciences association of Canada annual conference, (2004).

[24] Baar, T., Brucker, P. and Knust, S. "Tabu-search algorithms and lower bounds for the resource-constrained project scheduling problem", In: Voss S, Martello S, Osman I, Roucairol C, EDs., Meta-heurisitics: advances and trends in local search paradigms for optimization. Dordrecht: Kluwer, pp 1–8 (1998).

[25] Zhang, H., Li, X., Li, H. and Huang, F. "Particle swarm optimization-based schemes for resource-constrained project scheduling", Automation in Construction, 14, pp 393– 404 (2005).

[26] Chen, R.M., Wu, C.L., Wang, C.M. and Lo, S.T. "Using novel particle swarm optimization scheme to solve resource-constrained scheduling problem in PSPLIB", Expert Systems with Applications, 37, pp 1899–1910 (2010).

[27] Fang, C. and Wang, L. "An effective shuffled frog-leaping algorithm for resource-constrained project scheduling problem", Computers & Operations Research, 39, pp 890–901 (2012).

[28] Sebt, M.H., Alipouri, Y. and Alipouri, Y. "Solving resource-constrained project scheduling problem with evolutionary programming", Journal of the Operational Research Society advance online publication, doi:10.1057/jors.2012.69, (2012).

[29] Long, L.D. and Ohsato, A. "Fuzzy critical chain method for project scheduling under resource constraints and uncertainty", International Journal of Project Management, 26, pp 688–698 (2008).

[30] Demeulemeester, E.L. and Herroelen, W.S. "Project scheduling: a research handbook", Kluwer Academic Publishers, 685p (2002).

[31] Khatib, W. and Fleming, P.J. "The Stud GA: A Mini Revolution?", Springer, pp 683-691 (1998).

[32] Fedoroff , N. and Botstein, D., Eds., "The Dynamic Genome: Barbara Mc-Clintock's ideas in the century of genetics", Cold Spring Harbor, New York: Cold Spring Harbor Laboratory Press, (1992).

[33] Tang, W.K.S., Kwong, S.T.W. and Man, K.F. "A jumping genes paradigm: theory, verification, and applications", IEEE Circuits and Systems mag., 8, pp 18-38 (2008).

[34] Haupt, R.L. and Haupt, S.E. "Practical Genetic Algorithms", second ed., John Wiley & Sons, Inc., Publication, (2004).

[35] Brucker, P., Drexl, A., Mohring, R., Neumann, K. and Pesch, E. "Resource-constrained project scheduling: notation, classification, model, and methods", Eur. J. Operat. Res., 107, pp 272–288 (1998).

[36] Kolisch, R. "Serial and parallel resource-constrained project scheduling methods revisited: theory and computation", Eur. J. Operat. Res., 90, pp 320–333 (1996).