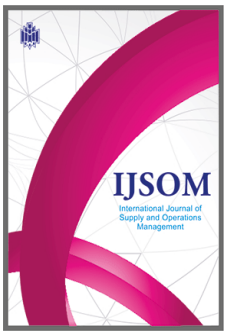
International Journal of Supply and Operations Management

IJSOM

November 2015, Volume 2, Issue 3, pp. 905-924
ISSN-Print: 2383-1359
ISSN-Online: 2383-2525
www.ijsom.com



An efficient genetic algorithm for solving the multi-mode resource-constrained project scheduling problem based on random key representation

Mohammad Hassan Sebt[*a], Mohammad Reza Afshar [a] and Yagub Alipouri [a]

[a] Department of Civil Engineering, Amirkabir University of Technology, Tehran, Iran

Abstract

In this paper, a new genetic algorithm (GA) is presented for solving the multi-mode resource-constrained project scheduling problem (MRCPSP) with minimization of project makespan as the objective subject to resource and precedence constraints. A random key and the related mode list (ML) representation scheme are used as encoding schemes and the multi-mode serial schedule generation scheme (MSSGS) is considered as the decoding procedure. In this paper, a simple, efficient fitness function is proposed which has a better performance compared to the other fitness functions in the literature. Defining a new mutation operator for ML is the other contribution of the current study. Comparing the results of the proposed GA with other approaches using the well-known benchmark sets in PSPLIB validates the effectiveness of the proposed algorithm to solve the MRCPSP.

Keywords: Combinatorial optimization; Multi-mode project scheduling; Resource constraints; Genetic algorithm; Random key representation.

1. Introduction

Project scheduling, as one of the main branches of the project planning, has been considerably employed by researchers in recent years (Chaleshtari and Shadokh 2014). Project scheduling is a troublesome duty due to resource and precedence constraints (Qi et al. 2014). The resource-constrained project scheduling problem (RCPSP) and its extended version, the Multi-mode RCPSP (MRCPSP), are important problems in this context, and the latter problem is more related to the real world (Hartmann and Briskorn 2009).

* Corresponding author email address: sebt@aut.ac.ir

The aims of MRCPSP are finding an execution mode and a feasible start time for each activity, such that makespan of the project is minimized under the resource and precedence constraints.

There are three methods for solving the MRCPSP: exact procedures, heuristic, and meta-heuristic approaches. In the exact methods, the first optimal procedure was a linear programming approach proposed by Slowinski (1980). In 1982, Talbot suggested an enumeration scheme which was improved by Patterson et al. (1989). Then, Sprecher (1994) reduced the computational effort of Talbot's enumeration scheme using some dominance criteria and a border bound. Speranza and Vercellis (1993) suggested a depth-first branch-and-bound algorithm; however, Hartmann and Sprecher (1996) demonstrated that if there are two or more renewable resources, their algorithm may be unable to find the optimal solution. Afterwards, different methods of branch and bound are proposed by Hartmann and Drexl (1998), Sprecher and Drexl (1998). And finally, Zhu et al. (2006) presented a branch-and-cut algorithm. Exact methods solve the problems optimally; however, they are unable to solve problems with more than 20 activities in an acceptable computational time (Sprecher and Drexl 1998). Thus, researchers make use of heuristic and meta-heuristic approaches to obtain reasonable project schedules with short computational time.

In the heuristic methods, Talbot (1982), and Sprecher and Drexl (1998) suggested applying a time limit on their exact branch-and-bound algorithms for solving large-sized problems. A stochastic scheduling procedure was proposed by Drexl and Gruenewald (1993). The single-pass and a multi-pass approach were suggested by Slowinski et al. (1994). Lova et al. (2006) proposed several multi-pass heuristics based on the priority rules.

Meta-heuristic approaches are the new generations of heuristic methods which have been used successfully for solving the MRCPSP over the last fifteen years. In 2004, Kolisch and Hartmann showed that the meta-heuristic approaches outperform the heuristic methods (Glover and Greenberg, 1989). In these methods, authors such as Mori and Tseng (1997), Özdamar (1999), Hartmann (2001), Alcaraz et al. (2003) and Lova et al. (2009) used Genetic Algorithms (GAs) to solve the MRCPSP. Jo´zefowska et al. (2001) presented a Simulated Annealing (SA) algorithm with and without the penalty function. Peteghem and Vanhoucke (2009) applied the Artificial Immune System (AIS) to solve the MRCPSP. Using the extended serial schedule generation scheme, Peteghem and Vanhoucke (2010) also applied a genetic algorithm to solve the preemptive MRCPSP. Ant colony optimization was presented by Zhang (2012). Damak et al. (2009) utilized the Differential Evolution (DE) algorithm for solving the MRCPSP and analyzed the effect of population size on improvement of the solutions. A multi-agent learning approach was provided by Wauters et al. (2009). Elloumi and Fortemps (2010) employed a hybrid rank-based evolutionary algorithm, which transformed the MRCPSP to a bi-objective problem. Wang and Fang (2011) proposed an effective Shuffled Frog-Leaping Algorithm (SFLA). They presented a new representation scheme as the encoding scheme and used the Taguchi method of Design Of Experiment (DOE) to determine a set of suitable parameters for the SFLA. Wang and Fang (2012) also presented an effective Estimation of Distribution Algorithm (EDA). In their algorithm, the activity-mode list (AML) was adopted as the encoding scheme, and parameters of their algorithm were settled based on DOE tests. Peteghem and Vanhoucke (2014) evaluated different meta-heuristic algorithms for solving the MRCPSP and proposed new benchmark instances for

this problem. Hao et al (2014) presented an effective Estimation Distribution Algorithm (EDA) for solving the MRCPSP in an uncertain environment with considering uncertain durations for activities. Cheng et al (2015) solved MRCPSP in three different conditions: with and without activity splitting, as well as preemption using an exact method. However, initial solutions were developed using the heuristic methods. Their results presented that when we permit the activities to have splitting or preemptions, better results are obtained. Beşikci et al (2015) proposed a genetic algorithm for solving the MRCPSP in multi-project environment.

In the present study, a new GA is proposed for solving the MRCPSP. A random key (RK) and the related mode assignment representation scheme are used as encoding scheme and multi-mode serial schedule generation scheme (MSSGS) is considered as the decoding procedure.

A simple, efficient fitness function is also introduced, thereby, average deviation from the optimal solution is reduced. Defining a new mutation operator for ML is the other contribution of the current study.

The remainder of the current study is organized as follows: In section 2, the MRCPSP is described; section 3 deals with the definition of the new proposed GA for solving the MRCPCP; in section 4, the results of our GA in solving the MRCPSP are reported, and finally, in section 5, concluding remarks are drawn out.

2. Problem description

In the MRCPSP, a project has J+2 activities with multiple execution modes with precedence relations among some of the activities. In this paper, these precedence relations are finish-start relations without time lags. The initial mode m, $m\in\{1,..,Mj\}$, of activity j (j=0,…,J+1) cannot be changed and the preemption is not allowed for the activities (Damak et al., 2009). Each mode requires one or several types of renewable or non-renewable resources. The availability of renewable resources k (k=1,…,R) and of non-renewable resources l (l=1,…,N) are R_k and N_l, respectively. If activity j is conducted with mode m, its duration, its required amount of renewable resources k (k=1,…,R), and its required amount of non-renewable resources l (l=1,…,N) are presented with d_{jm}, $r_{jmk,}$ and n_{jml}, respectively. The Start and end activities of the project are considered to be dummy activities without any durations and resource requirements. Under the above-mentioned conditions, the aims of the MRCPSP are to assign a feasible start time and a mode to each activity such that makespan of the project is minimized.

3. Presentation of new genetic algorithm in order to solve the MRCPSP

GA was suggested by Holland (Lee and El-Sharkawi 2008) for solving optimization problems. The initial population of this algorithm is developed randomly or by using heuristic methods, and the population evolves based on the following three sequential and iterative operators.

a) Selection operator
b) Crossover operator
c) Mutation operator (Lee and El-Sharkawi 2008)

In this research, after the preprocessing procedure, the initial population of GA is developed

randomly to diverse the search space and to find the promising regions (Lee and El-Sharkawi 2008). In the remainder of this section, the proposed GA for solving the MRCPSP is described in detail.

3.1 Preprocessing procedure

Before starting with GA, the preprocessing approach is employed for reducing the search space and the computational effort. For the first time, Sprecher et al. (1997) used it in branch and bound algorithm. Based on this preprocessing procedure, on the one hand, non-efficient and non-excusable modes are eliminated and, on the other hand, redundant non-renewable resources are omitted (for more information, see Sprecher et al. (1997).

3.2 Representation scheme

The issue of selecting an appropriate representation is crucial for the search (Lee and El-Sharkawi 2008). Based on RCPSP literature, the RK representation and activity list (AL) representation are the most popular ones. By using RK representation, many of the precedence feasibility issues are moved into the fitness function evaluation. In addition, in this representation scheme, the generation and evolution of the population are independent of the schedule generation schemes.

After considering the above-mentioned advantages, we decided to use the RK representation and the related mode list (ML) as encoding schemes. Therefore, each solution I is presented by two vectors λ and μ ($I=(\lambda, \mu)$), where λ and μ present the RK and the ML, respectively.

3.3 Fitness function

Evaluation of each chromosome is necessary for the evolution of the population, which is obtained through a fitness function. Determination of an appropriate fitness function is important for the correct operation of the GA. Crossover and mutation are general operators of GA which do not consider the feasibility of solutions. Thus, infeasible offsprings are frequently developed, which must be penalized. For this purpose, owing to the lack of information about an optimal solution, fitness function methods consider the distance of the feasible area. These methods are based on the number of violations. In the MRCPSP literature, several fitness functions have been proposed, in which the fitness function of Lova et al. (2009) (Eq.1) is the most recent one. In this last fitness function (Eq.1), infeasible solutions are penalized by $ERR(\mu)$, which presents the non-renewable infeasible degree of ML (Eq. 2). Obviously, when $ERR(\mu)$ is 0, the individual I is a feasible solution.

$$f(I)=\begin{cases} 1-\dfrac{max\_mak(P)-mak(I)}{max\_mak(P)}, & if\ ERR(\mu)=0 \\ 1+\dfrac{mak(I)-min\_CP}{mak(I)}+ERR(\mu), & otherwise \end{cases} \quad (1)$$

$$ERR(\mu)=\sum_{l=1}^{L} max\{0,\frac{\sum_{j=1}^{J} n_{jml}-N_l}{N_l}\} \quad (2)$$

Where $mak(I)$ is the makespan of the individual I and $max\_mak(P)$ gives the maximal makespan of feasible solutions related to individuals of the current generation. min_CP gives the minimal critical path of the project using the minimal duration of activities.

Lova et al. (2009) demonstrated that their fitness function gives better results than others and

removes their faults. However, in this paper, another fitness function is presented which is simple and efficient (Eq. 3).

$$f(I) = \begin{cases} 1 - \dfrac{T - mak(I)}{T}, & \text{if } ERR(\mu)=0 \\ 1 + \dfrac{mak(I)}{T} + ERR(\mu), & \text{otherwise} \end{cases} \quad (3)$$

Where T is the upper bound on the project's makespan given by the sum of the maximal durations of activities and *mak(I)* gives the makespan of the individual *I*.

The benchmark sets of J10, J12, J14, J16, J18, and J20 are selected from the Project Scheduling Problem Library (PSPLIB) as standard instances for testing the fitness functions. The halting criterion considered here is the generation of 5000 schedules.

Considering Table 1, the acquired results show that the CPU times resulted from using the proposed fitness function are approximately equal to those of using Lova et al's, but, the obtained average deviations indicate that, for all of the benchmark sets except for J12, the fitness function of this paper outperforms Lova et al's one.

Table 1. Comparison of performance of the fitness functions

	Fitness function	*Set of instances*					
		J10	***J12***	***J14***	***J16***	***J18***	***J20***
Average deviation (%)	Current study	0.03	0.08	0.15	0.30	0.30	0.45
	Lova et al. [23]	0.04	0.07	0.16	0.30	0.32	0.46
Percentage of solving optimally	Current study	98	94	90	80	81	75
	Lova et al. [23]	96	94	90	81	79	73
CPU time (seconds)	Current study	0.11	0.14	0.17	0.19	0.20	0.22
	Lova et al. [23]	0.11	0.14	0.18	0. 19	0. 20	0.23

After the evaluation of an individual by the fitness function, the corresponding fitness value is attributed to that individual.

3.4 Selection operator

In the current study, based on the fitness of individuals, some of the best individuals (TOP individuals in Figure1) are copied from the current generation to the next generation. This strategy is called elicit strategy. The drawback of the mentioned strategy is its convergence to a local minimum, which is controlled in this paper by two operators. These operators are crossover and new efficient mutation operators which are introduced in the following subsections. In parent selection, the father and mother are randomly selected from the best individuals' pool (TOP individuals in Fig. 1) and total individuals of the current population, respectively (see Fig. 1). This type of selection operator was applied in a GA on the RCPSP by Mendes et al. (2009) and the promising results were obtained.

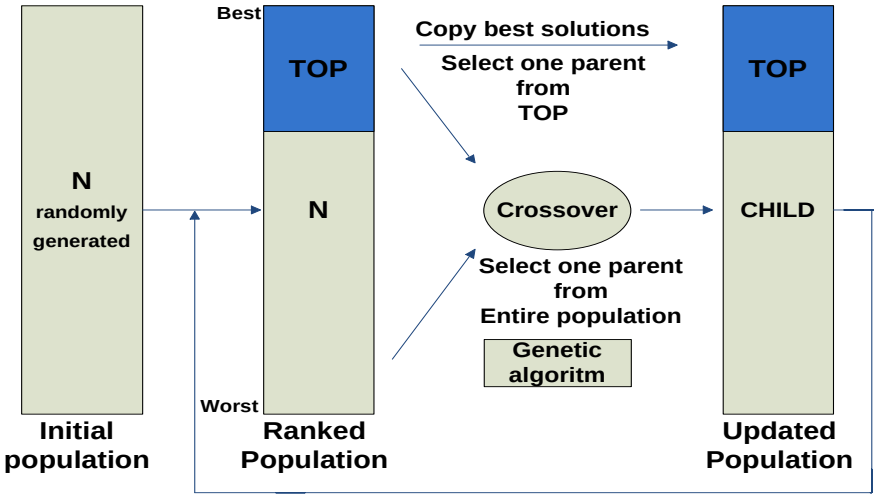


Figure 1. Selection operator in the proposed GA of the current study

3.5 Crossover operator

Spears and De Jong (1991) investigated different crossovers and indicated that uniform crossover gives the best results. Based on their study, performance of the uniform crossover is independent of the chromosomes length. Unlike the one and two point crossovers, the uniform crossover can develop offspring in each point of space. Furthermore, it has been proven that the uniform crossover has more exploratory power than the n-point crossover (Eschelman et al., 1989). The uniform crossover was also used for solving the RCPSP by Mendes et al (2009) and the promising results were acquired. Therefore, the selection of crossover is advisable and used in this paper.

In the uniform crossover, a number from interval [0,1] is considered as the Crossover Probability (CProb). Then, a random vector containing numbers between [0,1] is generated. After comparing the crossover probability and the numbers of random vector, the genes are determined to be transferred from father and mother to offspring. When the random number is smaller than the crossover probability, the related gene in the father is transferred to the son offspring and the related gene in the mother is transferred to the daughter offspring; otherwise, the exchange of genes is from mother to son and father to daughter (see Figure2). Production of the daughter offspring is important to avoid the convergence to a local minimum. In the example of Figure 2, 0.7 is considered to be the crossover probability.

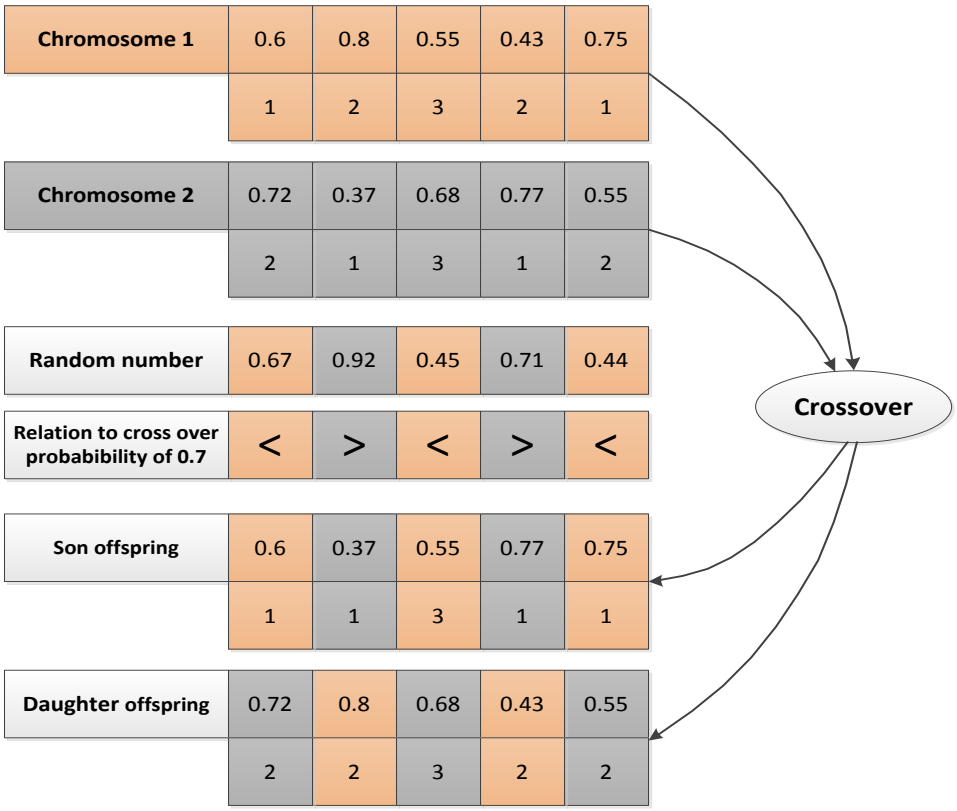


Figure 2. An example for Crossover operator in the proposed GA (crossover probability= 0.7)

3.6 Mutation operator

The mutation operator is an important tool to maintain diversity within the population in order to prevent the premature convergence (Lee and El-Sharkawi 2008). In this study, in order to apply mutation on activities priority, several activities in different or similar solutions are selected randomly and their priorities are also changed randomly. In order to change the mode for the chosen activity, a new mutation operator is defined. This new mutation operator acts differently on feasible and infeasible solutions. For the feasible solution and a randomly selected activity, if the new mode list stays feasible and $\eta\ (\mu)$ in Eq.4 increases, the previous mode of that activity is replaced with the new one; otherwise, no change is made. Obviously, the value of $\eta\ (\mu)$ for the feasible solution is negative or zero and with the help of Eq. 4, it exploits non-renewable resources.

$$\eta(\boldsymbol{\mu}) = \sum_{l=1}^{N} \frac{\sum_{j=1}^{j} n_{jml} - N_l}{N_l} \tag{4}$$

For the infeasible solution and a randomly selected activity, if $ERR\ (\mu)$ of the new mode list reduces, the new mode of that activity is replaced with the previous one; otherwise no change is made. The proposed mutation operator is repeated until the pre-specified mutation rate is reached.

3.7 Schedule generation scheme (decoding procedure)

Schedule generation scheme (*SGS*), as an efficient method, transforms a representation solution of the RCPSP to a schedule (Lova et al. 2009). There are two types of *SGS*: serial *SGS* and parallel *SGS*. Since the parallel *SGS* cannot always find the optimal solution (Lova et al. 2009), we make use of the serial *SGS*. In the MRCPSP, the serial *SGS* must be modified in order to deal with

nonrenewable resources. In fact, two important issues must be considered: first, the transformation of infeasible solutions to feasible ones as far as possible, and second, full exploitation of non-renewable resources in the feasible solutions. For these two purposes, we use both infeasible and feasible tackling procedures. These procedures are the extended versions of the serial *SGS*, namely multi-mode serial schedule generation scheme (MSSGS).

3.7.1 Infeasible tackling procedure

In case of infeasible solutions, the best technique is the application of repairing procedure for some of them (Lee and El-Sharkawi 2008). Thus, in this research, after generating the initial population and employing the preprocessing procedure, the infeasible tackling approach is employed to transform some of the infeasible solutions into the feasible ones. In this procedure, an activity is selected randomly and its mode is changed to a new mode. If the resulted *ERR (μ)* is less than the previous *ERR(μ)*, the previous mode is replaced with the new one. This procedure continues until all the infeasible solutions are transformed into feasible ones or the pre-specified maximum number of iterations for SGS (*SGS_adj*) is equal to the number of non-dummy activities, J. After the mentioned procedure, only feasible solutions are allowed to enter the multi-mode forward-backward iteration (MM-FBI) method, which is described in the following subsection. The feasible tackling procedure is applied in the MM-FBI.

3.7.2 Multi- mode forward-backward iteration (MM-FBI) method

In the RCPSP, Forward–Backward iteration (FBI) is an efficient technique to improve the solutions' quality (Lova et al. 2009). This idea was extended by Lova et al. (2009) for the MRCPSP, namely multi- mode forward-backward iteration (MM-FBI) method. If this approach applies to one activity, using only mode change of that activity, its finish time will reduce.

As a drawback for the MM-FBI, the activities scheduled earlier have a greater chance for more occupation of non-renewable resources. For overcoming this weakness, the mode improvement procedure is randomly applied on some of the activities and the *SGS_Pper* specifies the percentage of total non-dummy activities selected for the mode improvement procedure. Consequently, it is guaranteed that all of the activities have equal opportunity for the occupation of non-renewable resources (Lova et al. 2009).

The MM-FBI is repeated until no further improvement in the makespan is achievable. Here, an illustrative simple instance is provided to present the effectiveness of the Lova et al's mode improvement procedure as follows:

Consider a project with 9 non-dummy activities and 2 dummy activities, one renewable resource type and one non-renewable resource type. In this project, each activity has two modes. Start and end activities are dummy activities with only one mode, zero durations and no consumption of resources. The availability of renewable and non-renewable resources are 2 and 28, respectively. The complete information and precedence constraints of this project are presented in Table 2 and Figure 3, respectively.

Table 2. Data for activities in the example project

Activity No.	*Mode 1*			*Mode 2*		
	r_{j1k}	n_{j1l}	d_{1j}	r_{j2k}	n_{j2l}	d_{2j}
0	0	0	0	0	0	0
1	2	4	2	1	2	3
2	1	5	2	1	2	4
3	1	3	2	1	1	3
4	1	2	1	1	1	2
5	2	5	2	2	3	4
6	1	4	2	1	3	3
7	2	2	1	2	1	2
8	2	3	1	1	2	3
9	1	3	4	1	1	6
10	0	0	0	0	0	0

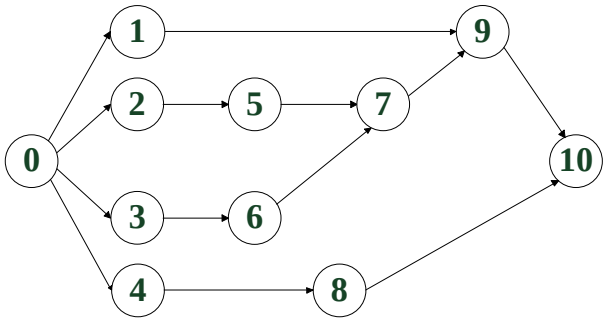


Figure 3. Activity network of an example of the MRCPSP

In the above example, consider solution A which is given in Figure 4.

Solution A	λ=	0.00	0.65	0.90	0.55	0.80	0.75	0.50	0.40	0.45	0.35	0.00
	μ=	1	1	1	2	2	1	1	2	2	1	1

Figure 4. An example of a particle representing a solution for the MRCPSP.

A resource graph of solution A is developed by SSGS procedure, which is presented in Figure5-a (3.2 means activity 3 with mode 2).

The MM-FBI starts with backward scheduling on the existing schedule which is presented in Fig. 5-a. For this purpose, the finish times of activities in the existing schedule (Figure5-a) are considered as the activities priority, and the mode improvement procedure is applied over the generation of this schedule on some of the activities specified randomly. Suppose activities 5 and 7 are selected randomly for this purpose.

As it can be seen in Figure5-b, the mode improvement procedure has reduced the duration of activity 5 by 2 periods and the duration of activity 7 by 1 periods. From these changes, the required non-renewable resource has changed from 25 to 28 (i.e., the availability of non-renewable resource). A single iteration of the MM-FBI is completed with the forward

scheduling. For the forward scheduling, start times of activities in the existing backward schedule are used for the activities priority. If the availability of non-renewable resources is not used completely, the mode improvement procedure is also applied in this direction.

In solution A, as the availability of non-renewable resources is used completely for the above backward schedule, the mode improvement procedure is not applied in the forward scheduling. After finishing this single iteration of MM-FBI, makespan of the project is reduced from 18 to 12 (see Figure5-c).

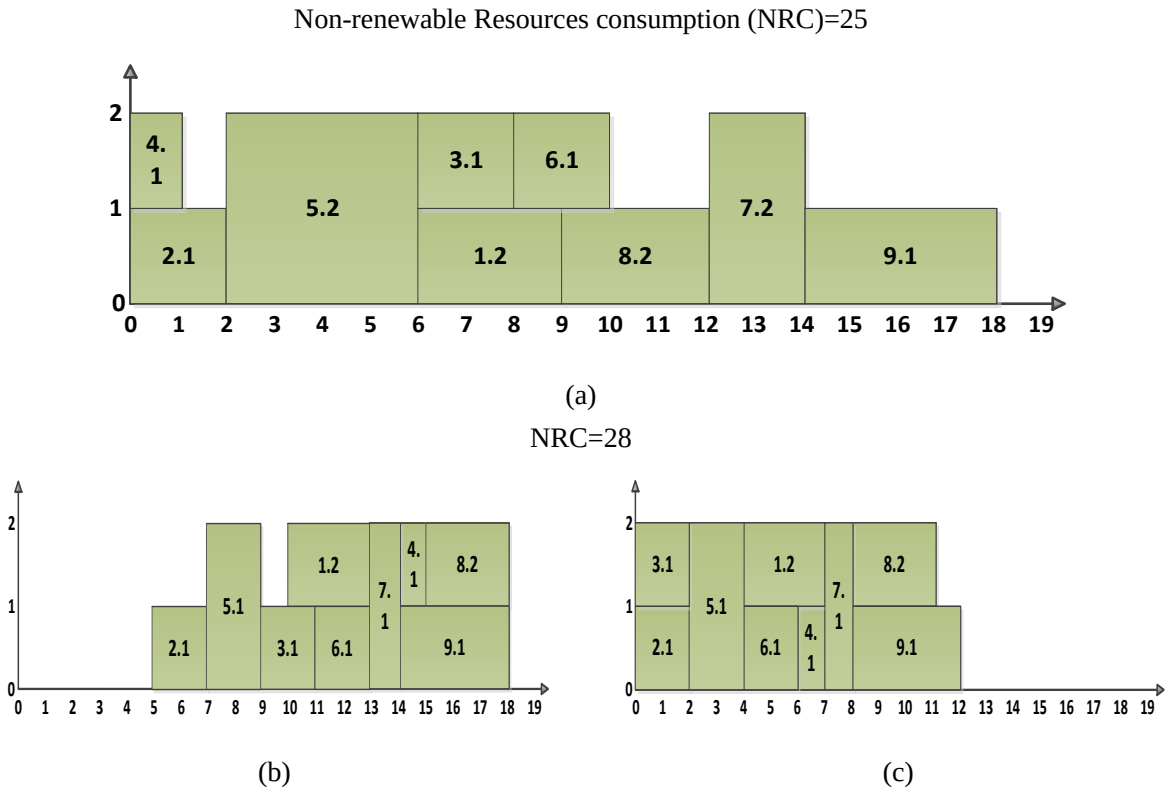


Figure 5. A single iteration step of the MM-FBI and non-renewable resource consumed (NRC); a) existing schedule b) backward scheduling c) forward scheduling.

4. Computational experiments

In this section, the performance of the proposed GA to solve the MRCPSP is investigated. The proposed GA was programmed with Matlab R2012a, and the tests were accomplished on a laptop with an Intel® core 2 T9300 2.5 GHz processor.

In the literature, the CPU time and the number of generated schedules are used as the stopping criteria for the comparison of different proposed algorithms. Since "5000 generated schedules" halting criterion has been used more than the other halting criteria by the researchers for the comparison, it is also used in the current study.

To illustrate the effectiveness of the proposed GA, the well-known benchmark sets of J10, J12, J14, J16, J18, J20, and J30 generated by the project generator ProGen for the MRCPSP and

available at http://129.187.106.231/psplib/ have been used . These sets contain instances with 10, 12, 14, 16, 18, 20, and 30 non-dummy activities, respectively. Each of these sets has 640 instances, some of which are infeasible instances.

Therefore, we exclude these infeasible instances from the experiments. In each instance, an activity can be performed in one out of three modes with specific duration between 1 and 10 periods of time. Each instance contains two renewable resource types and two non-renewable resource types.

For all of the mentioned benchmark sets, with the exception of J30, the optimal makespans of all feasible instances are known. In fact, the optimal values for J30 have not been found so far.

4.1. Parameter setting

Taguchi method of design of experiment (DOE) (Montgomery, 2005) has been used for parameter setting of the proposed GA. For this purpose, data set of J20 is selected as the most difficult standard MRCPSP data set. The aim of this subsection is to determine the appropriate values for four key parameters of the proposed GA: Population Size (Pop_Size), the best individuals (TOP), Crossover Probability (CProb), and Mutation rate. Different test values of these parameters are presented in Table 3. In this table, Pop_size is the initial population size and other parameters have been described in different sections of the current study. Based on the number of parameters and the number of factor levels, the orthogonal array $L_{16}(4^4)$ is developed in Table 4.

Table 3. Combinations of parameter values

Parameters	*Factor level*			
	1	*2*	*3*	*4*
Pop_size	2`J	3J	4J	5J
Top	0.05	0.10	0.15	0.20
CProb	0.65	0.70	0.75	0.80
Mutation rate	0.05	0.10	0.15	0.20

In order to find the best combination, different combinations of parameter values in Table 4 were tested using the set J20 and then the average response variable (*ARV*) value (Eq. 7) of each combination was calculated and presented in the sixth column of Table 4. In this section, 5000 generated schedules have been also considered as the halting criterion.

$$ARV = \frac{1}{N}\sum_{i=1}^{N}\frac{Makespan_i - OPT_i}{OPT_i} \quad (7)$$

where $Makespan_i$ is the makespan of each instance of J20 which was obtained by the proposed GA, OPT_i is the optimum makespan of each instance of J20 which exists in the PSBLIB, and N is the number of feasible instances of set J20 which contains 554 instances.

According to the obtained *ARV* values from different combination of parameter values in Table 4, the trend of each factor level is demonstrated in Fig. 6. The relevant results are listed in Table 5.

As it can be observed in Fig. 6, the population size has the most significant impact on the performance of the proposed GA such that with increasing the population size, solution quality improves. Based on these trends, the best combination of values is listed in Table 6.

Table 4. Orthogonal table for the proposed GA.

Test number	*factors*				*ARV*
	Pop_size	*Top*	*CProb*	*Mutation rate*	*5000 scheduling*
1	1	1	1	1	0.036557
2	1	2	2	2	0.014197
3	1	3	3	3	0.050063
4	1	4	4	4	0.014557
5	2	1	2	3	0.021634
6	2	2	3	4	0.028676
7	2	3	4	1	0.033184
8	2	4	1	2	0.014206
9	3	1	3	1	0.027619
10	3	2	4	2	0.015499
11	3	3	1	3	0.031665
12	3	4	2	4	0.014135
13	4	1	4	3	0.014266
14	4	2	1	4	0.014127
15	4	3	2	1	0.017260
16	4	4	3	2	0.013747

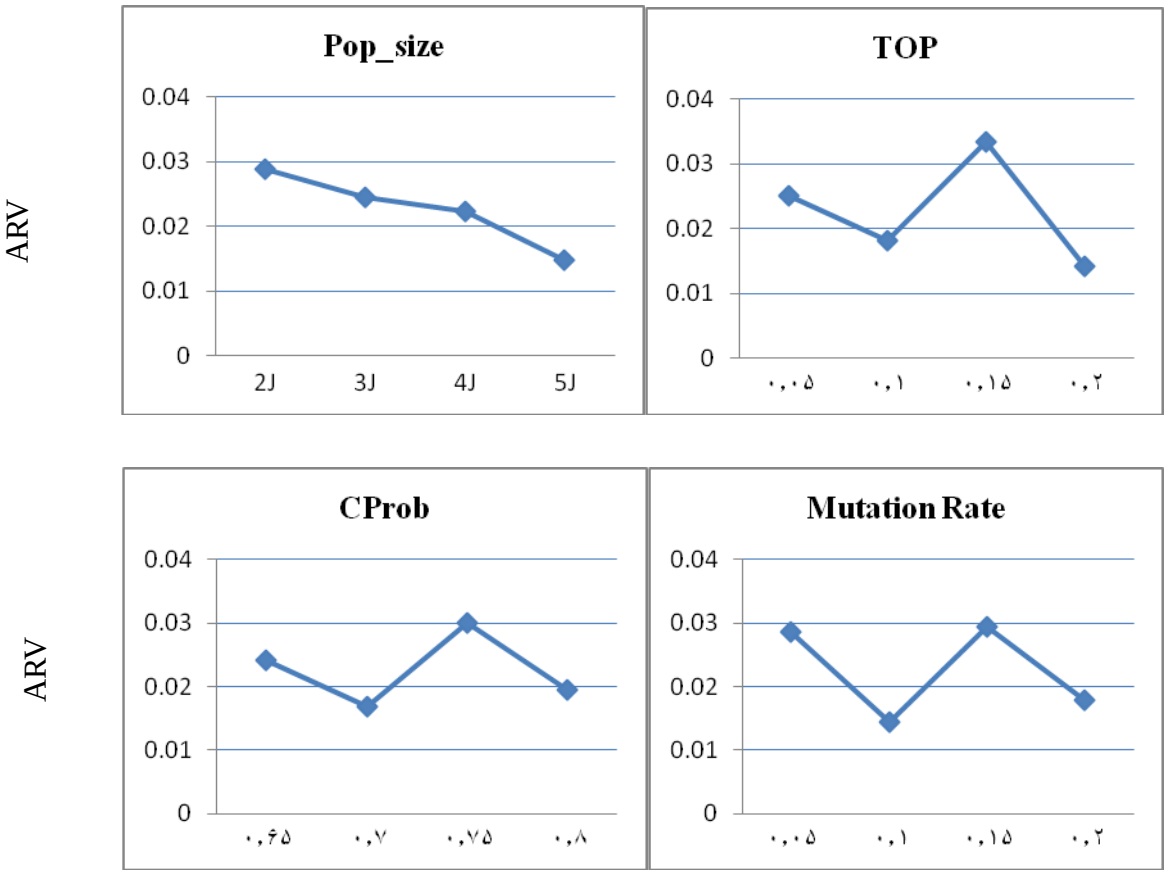


Figure 6. Factors' level trend with 5000 schedules.

Table 5. Response table for ARV with 5000 schedules.

Level	*POP_size*	*TOP*	*CProb*	*Mutation rate*
1	0.028844	0.025019	0.024139	0.028655
2	0.024425	0.018125	0.016807	0.014412
3	0.02223	0.0333043	0.030026	0.029407
4	0.01485	0.014161	0.019377	0.017874

Table 6. The best combination of parameters for the proposed GA.

Level	*POP_size*	*TOP*	*CProb*	*Mutation rate*
5000 schedules	5J	0.20	0.70	0.10

4.2. Comparison with existing algorithms

Our statistical results are presented in Table 7. Average deviation from the optimal makespans (in the case of set J30, from the lower bounds), percentage of optimally solved instances, and the average CPU time in seconds are used for the comparison. Optimal values for sets of J10-20 and the best-known solutions (BKS) till now (Results on the PSPLIB-server on 18 September 2015) for set J30 are considered as the base for calculating the average deviations. Accordingly, our

results are compared with the existing state-of-the-art methods for solving the MRCPSP such as the GA and the AIS presented by Peteghem and Vanhoucke (2010, 2009) denoted respectively as VPVGA and VPVAIS, EDA and SFLA presented by Wang and Fang (2012, 2011) denoted respectively as LCEDA and LCSFLA, the hybrid genetic algorithm developed by Lova et al. (2009) denoted as LHGA, the hybrid rank-based evolutionary algorithm proposed by Elloumi and Fortemps (2010) denoted as EFEA, the hybrid scatter search developed by Ranjbar et al. (2009) denoted as RSS, the genetic algorithm developed by Alcaraz et al. (2003) denoted as AGA, the two-phase genetic local search algorithm applied by Tseng and Chen (2009) denoted as TCGLS and the simulated annealing proposed by Jo´zefowska et al. (2001) denoted as JSA.

Table 7. Performance of the proposed GA on solving benchmark sets.

Benchmark set	*Feasible solutions*	*Number of schedules*	*Average deviation (%)*	*Optimal rate (%)*	*CPU-time(s)*
J10	536	1000	0.13	94.44	0.03
		3000	0.04	96.97	0.06
		5000	0.03	98	0.10
J12	547	1000	0.37	85	0.03
		3000	0.15	92	0.07
		5000	0.07	95	0.13
J14	551	1000	0.52	74	0.04
		3000	0.23	86	0.10
		5000	0.21	85	0.15
J16	550	1000	0.96	55	0.04
		3000	0.49	72	0.12
		5000	0.32	74	0.17
J18	552	1000	0.89	59	0.05
		3000	0.54	73	0.12
		5000	0.40	76	0.17
J20	554	1000	1.50	47	0.06
		3000	0.72	64	0.13
		5000	0.56	67	0.19

A comparison of average deviations can be made using Table 8. As it can be seen, in the set of J10, only VPVGA and VAVAIS perform better than our GA. While, increasing the number of activities, in the sets of J14, J16, J18, and J20, our proposed GA performs better than the other proposed algorithms and the trend of solution quality is improved. This implies that the proposed GA can outperform other algorithms by increasing the complexity. A comparison of our GA with VPVGA in Table 9, with regard to set J30, may also confirm this statement. The third and fourth columns of Table 9, present the percentage of instances that their results are equal or inferior to that of the best-known solutions.

It is worth mentioning that no improvement is made in the best-known solutions for set J30, which is analogous to VPVGA.

Table 8. Average deviations (%) from optimal makespans (5000 generated schedules as the stopping condition).

Algorithm	*J10*	*J12*	*J14*	*J16*	*J18*	*J20*
Proposed GA	0.03	0.07	**0.21**	**0.32**	**0.40**	**0.56**
VPVGA	**0.01**	0.09	0.22	0.32	0.42	0.57
VPVAIS	0.02	**0.07**	0.20	0.39	0.52	0.70
LCEDA	0.12	0.14	0.43	0.59	0.90	1.28
LHGA	0.06	0.17	0.32	0.44	0.63	0.87
LCSFLA	0.10	0.21	0.46	0.58	0.94	1.40
EFEA	0.14	0.24	0.77	0.91	1.30	1.62
RSS	0.18	0.65	0.89	0.95	1.21	1.64
AGA	0.24	0.73	1.00	1.12	1.43	1.91
TCGLS	0.33	0.52	0.93	1.081	1.32	1.69
JSA	1.16	1.73	2.60	4.07	5.52	6.74

Table 9. Average deviation (%) from the lower bounds of instances in set J30 (5000 generated schedules as the stopping condition).

Algorithm	*Dev. BKS (%)*	*CPU time (s)*	*equal*	*Worth*
Proposed GA	1.05	0.27[a]	55.6	44.4
VPVGA	1.08	0.24[b]	71.0	29

[a] T9300 2.5 GHz.
[b] Pentium 2.80 GHz

A comparison of CPU time must be drawn based on computing power. In fact, computers with powerful processors process a program in a short time. Comparing CPU time in Table 10 and considering the lower computational power of this study compared to that of others, the CPU time of proposed algorithm can be considered satisfactory.

Table 10. Comparison with other proposed algorithms considering the average CPU time (5000 generated schedules as the stopping condition).

Algorithm	*J10*	*J12*	*J14*	*J16*	*J18*	*J20*
Proposed GA[a]	0.10	0.13	0.15	0.17	0.17	0.19
VPVGA[b]	0.12	0.13	0.14	0.15	0.16	0.17
LHGA[c]	**0.08**	0.10	**0.11**	**0.12**	**0.13**	**0.15**
LCSFLA[d]	0.07	**0.09**	0.13	0.15	0.19	0.27

[a] T9300 2.5 GHz.
[b] Pentium 2.80 GHz.
[c] Pentium 3GHz.
[d] T7500 2.2 GHz.

Table 11 presents the comparison of the optimal rates. As it can be seen, the optimal rates of the proposed algorithm are better than JSA, but not as good as the other proposed algorithms. However, the results of the proposed GA present that most of the solutions found are close to the optimum ones. For example, in set J10, our algorithm solved 525 out of 536 problems to optimality, and in all other instances, the solutions are close to the optimum. Therefore, this algorithm worked well on all of the problems of set J10. Similarly, it worked well on 99% of the problems of J12, 97% of the problems of J14, 95% of the problems of J16 and J18, and 93% of the problems of J20. It is worth mentioning that in this paper, the performance of a solution with the maximum deviation of 2 periods from optimality is considered to be well. All comparisons indicate the effectiveness of the proposed GA.

Table 11. Comparison with other proposed algorithms considering optimal rates

Algorithm	*J10*	*J12*	*J14*	*J16*	*J18*	*J20*
Proposed GA	98	95	85	74	76	67
VPVGA	**99.63**	**98.17**	**94.56**	**92.00**	**88.95**	**85.74**
LHGA	98.51	96.53	92.92	90.00	84.96	80.32
LCSFLA	97.93	95.98	90.86	86.49	79.44	72.84
JSA	85.60	80.30	66.40	54.7	43.50	35.70

5. Conclusion

In the current study, a new GA was proposed to solve the MRCPSP. A random key and the related mode list representation are used as encoding and MSSGS was considered as the decoding procedure. In the feasible tackling procedure, we employed Lova et al's mode improvement procedure. One of the contributions of this paper was to define a new mutation operator for feasible and infeasible solutions. We also presented a new fitness function, which was simple and effective. The well-known benchmark sets J10, J12, J14, J16, J18, J20, and J30 in PSPLIB were used for testing the proposed GA. The performance comparisons indicated that the proposed GA is among the most competitive algorithms, particularly when the complexity increases.

References

Alcaraz J. Maroto C. and Ruiz, R. (2003). Solving the multi-mode resource-constrained project scheduling problem with genetic algorithms. *Journal of the Operational Research Societ*, Vol. 54, pp. 614–626.

Anderson E. and Ferris M. (1994). Genetic algorithm for combinatorial optimization: assembly line balancing problem. *Operations Research Society of America journal on computing, Vol.* 6, pp. 161–173.

Beşikci U. Bilge Ü. and Ulusoy, G. (2015). Multi-mode resource constrained multi-project scheduling and resource portfolio problem. *European Journal of Operational Research*, Vol. 240 (1), pp. 22-31.

Chaleshtari A.S. and Shadokh S. (2014). A branch and cut algorithm for resource-constrained project scheduling problem subject to nonrenewable resource with pre-scheduled procurement *Arabian Journal of Science Engineering*, Vol. 39, pp. 8359- 8369.Cheng, J. Fowler, J. Kempf, K. and Mason, S. (2015). Multi-mode resource-constrained project scheduling problems with non-preemptive activity splitting. *Computers & Operations Research*, Vol. 53, pp. 275-287.

Damak N. Jarboui, B. Siarry, P. and Loukil, T. (2009). Differential evolution for solving multi-mode resource constrained project scheduling problems. *Computers and Operations Research*, Vol. 36, pp. 2653-2659.

Drexl A. and Grünewald J. (1993). Nonpreemptive multi-mode resource-constrained project scheduling. *IIE Transactions*, Vol. 25, pp. 74–81.

Elloumi S. and Fortemps P. (2010). A hybrid rank-based evolutionary algorithm applied to multi-mode resource-constrained project scheduling problem. *European Journal of Operational Research*, Vol. 205, pp. 31–41.

Eschelman L., Caruana R. and Schaffer D. (1989). *Biases in the Crossover Landscape. Proc. third international conference on genetic algorithms*, Morgan Kaufman Publishing, February, pp. 21-29.

Glover F. and Greenberg H.J. (1989). New Approaches for Heuristic Search: A Bilateral Linkage with Artificial Intelligence. *European Journal of Operational Research*, Vol. 39, pp. 119-130.

Guangqiang Li, G. Zhao F. Guo C. and Teng H. (2006). *Parallel Hybrid PSO-GA Algorithm and Its Application to Layout Design*. ICNC 2006, Part I, LNCS 4221, pp. 749 – 758.

Haoa X. Linb L. Gen M. (2014). An Effective Multi-objective EDA for Robust Resource Constrained Project Scheduling with Uncertain Durations. *Procedia Computer Science*, Vol. 36, pp. 571-578.

Hartmann S. (2001). Project scheduling with multiple modes: A genetic algorithm. *Annals of Operations Research*, Vol. 102, pp. 111–135.

Hartmann S. and Briskorn D. (2009). A survey of variants and extensions of the resource-constrained project scheduling problem. *European Journal of Operational Research*, Vol. 207, pp. 1-14.

Hartmann S. and Drexl A. (1998). Project scheduling with multiple modes: A comparison of exact algorithms. Networks, Vol. 32, pp. 283–297.

Hartmann S. and Sprecher A. (1996). A note on `Hierarchical models for multi-project planning and scheduling. *European Journal of Operational Research*, Vol. 94, pp. 377-383.

Jo´zefowska J. Mika M. Rozycki, R. Waligora G. and Weglarz J. (2001). Simulated annealing for multi-mode resource-constrained project scheduling. *Annals of Operations Research*, Vol. 102, pp. 137–155.

Juang C.F. (2004). A Hybrid of Genetic Algorithm and Particle Swarm Optimization for Recurrent Network Design. *IEEE TRANSACTIONS ON SYSTEMS*, Vol. 34 (2), pp. 997-1006.

Kao Y.T. and Zahara E. 2008. "A hybrid genetic algorithm and particle swarm optimization for multimodal functions. *Applied Soft Computing*, Vol. 8, pp. 849–857.

Kolisch R. and Drexl A. (1997). Local search for non-preemptive multi-mode resource constrained project scheduling. *IIE Transactions*, Vol. 29, pp. 987–999.

Lee K. and El-Sharkawi M. (2008). *Modern heuristic optimization techniques*.

Lova, A., Tormos P. and Barber F. (2006). Multi-mode resource constrained project scheduling: Scheduling schemes, priority rules and mode selection rules. *Inteligencia Artificial*, Vol. 30, pp. 69–86.

Lova A., Tormos P. Cervantes M. and Barber F. (2009). An efficient hybrid genetic algorithm for scheduling projects with resource constraints and multiple execution modes. *International Journal of Production Economics*, Vol. 117 (2), pp. 302– 316.

Marco A. and Stützle T. (2008). *Convergence behavior of the fully informed particle swarm optimization algorithm. Proceedings of the 10th annual conference on Genetic and evolutionary computation*, January, pp.71-78.

Mendes J., Gonçalves J. and Resende M. (2009). A random key based genetic algorithm for the resource constrained project scheduling problem. *Computers and Operations Research*, Vol. 36, pp. 92 – 109.

Mendes R., Kennedy J. and Neves J. (2004). The Fully Informed Particle Swarm: Simpler, Maybe Better. *IEEE Transactions on Evolutionary Computation*:, pp. 204-210.

Montgomery D.C. (2005). Design and analysis of experiments. Arizona, John Wiley & Sons Press.

Mori M. and Tseng C. (1997). A genetic algorithm for multi-mode resource constrained project scheduling problem. *European Journal of Operational Research*, Vol. 100, pp. 134–141.

Özdamar L. 1999. A genetic algorithm approach to a general category project scheduling problem. IEEE *Transactions on Systems*, Vol. 29, pp. 44–59.

Patterson J.H., Słowinski R. Talbot F. and Weglarz J. (1989). An algorithm for a general class of precedence and resource constrained scheduling problems. *Advances in Project Scheduling*, pp. 3–28.

Peteghem V.V. and Vanhoucke M. (2009). *An artificial immune system for the multi-mode resource-constrained project scheduling problem*. In: EvoCOP, Springer.

Peteghem V.V. and Vanhoucke M. (2010). A genetic algorithm for the preemptive and

non-preemptive multi-mode resource-constrained project scheduling problem. *European Journal of Operational Research*, Vol. 201, pp. 409–418.

Peteghem V.V. and Vanhoucke M. (2014). An experimental investigation of metaheuristics for the multi-mode resource-constrained project scheduling problem on new dataset instances. *European Journal of Operational Research*, Vol. 235 (1), pp. 62 – 72.

Qi J.J. Liu, Y.J. Lei H.T. and Guo B. (2014). Solving the multi-mode resource availability cost problem in project scheduling based on modified particle swarm optimization. *Arabian Journal of Science Engineering*, Vol. 39, pp. 5279- 5288.

Ranjbar M. Reyck B. and Kianfar F. (2009). "A hybrid scatter search for the discrete time/resource trade-off problem in project scheduling. *European Journal of Operational Research*, Vol. 193 (1), pp. 35–48.

Slowinski R. (1980). Two approaches to problems of resource allocation among project activities – A comparative study. *Journal of Operational Research Society*, Vol. 8, pp. 711–723.

Slowinski R. Soniewicki B. and Weglarz J. (1994). DSS for multi-objective project scheduling. European Journal of Operational Research, Vol. 79, pp. 220–229.

Spears W. and De Jong K. (1991). On the virtues of parameterized uniform crossover. In: Proceedings of the fourth international conference on genetic algorithms, February, pp. 230–236.

Speranza M.G. and Vercellis C. (1993). Hierarchical models for multi-project planning and scheduling. *European Journal of Operational Research*, Vol. 64, pp. 312-325.

Sprecher A. (1994). *Resource-constrained project scheduling: exact methods for the multi-mode case*. Lecture Notes in Economics and Mathematical Systems.

Sprecher A. and Drexl A. (1998). Solving multi-mode resource-constrained project scheduling problems by a simple, general and powerful sequencing algorithm. *European Journal of Operational Research*, Vol. 107, pp. 431–450.

Sprecher A. Hartmann S. and Drexl A. (1997). An exact algorithm for project scheduling with multiple modes. OR Spektrum. *Organ der Deutschen Gesellschaft fur Operations Research*, Vol. 19 (3), pp. 195-203.

Talbot F.B . 198 *Management Science*, Vol. 28, pp. 1197–1210.

Tseng L.Y. and Chen S.C. (2009). Two-phase genetic local search algorithm for the multi-mode resource-constrained project scheduling problem. *IEEE Transactions on Evolutionary Computation*, Vol. 13 (4), pp. 848–57.

Wang L. and Fang C. (2011). An effective shuffled frog-leaping algorithm for multi-mode resource-constrained project scheduling problem. *Information Sciences*, Vol. 181, pp. 4804–4822.

Wang L. and Fang C. (2012). An effective estimation of distribution algorithm for the multi-mode resource-constrained project scheduling problem. *Computers and Operations Research*, Vol. 39: 449–460.

Wauters T. Verbeeck K. Berghe G. and De Causmaecker P.(2009). A multi-agent learning approach for the multi-mode resource-constrained project scheduling problem. In: Proceedings of the 8th International Conference on Autonomous Agents and Multi agent Systems, June, pp. 1-8.

Zhang H. (2012). Ant Colony Optimization for Multimode Resource-Constrained Project Scheduling." *American Society of Civil Engineers*, Vol. 28, pp. 150-159.

Zhu G., Bard J. and Tu G. (2006). A branch-and-cut procedure for the multimode resource-constrained project-scheduling problem. *Journal on Computing*, Vol. 18 (3), pp. 377–39.