Journal Of Al Azhar University Engineering Sector

Vol. 12, No. 42, January, 2017, 187198

SOLVING RESOURCE-CONSTRAINED PROJECT SCHEDULING PROBLEM USING GENETIC ALGORITHM

Raafat Elshaer, Mona shawky, Hesham Elawady, Gamal Nawara

Industrial Engineering Department, Faculty of Engineering, Zagazig University, Sharkia, Egypt

ABSTRACT

Due to the combinatorial nature of the resource-constrained project scheduling problem (RCPSP), there is a lot of artificial intelligence methods proposed to solve it. The Genetic Algorithm (GA), one of these methods, is considered to be a valuable search algorithm capable of finding a reasonable solution in a short computational time. The primary objective of this paper is to build a genetic algorithm for solving RCPSP problem aiming at minimizing project's makespan. Based on a comprehensive review of different GAs and a full factorial experiment, a proposed GA has been presented. The proposed algorithm has been tested on a well-known benchmark (PSPLIB). The computation results show that the proposed GA outperforms many published algorithms and on average performs as well as other algorithms. Also, the performance of the algorithm improves in solving large scale problems.

Keywords: Resource Constrained Project Scheduling problems; Genetic Algorithm; Project makespan

1. INTRODUCTION

Resource-constrained project scheduling problem (RCPSP) is a well-known problem widely studied in the literature. There are several papers that review research for the problem. The general work of Brucker et al. (1998) as well as the work of Hartmann & Kolisch (2000) focuses on heuristic algorithms to solve the problem. It has been shown by Blazewicz et al. (1983) that the RCPSP belongs to the class of NP-hard optimization problems. The RCPSP can be stated as follows: A single project consists of $n + 1$ activities where each activity has to be processed in order to complete the project. The dummy activities 0 and $n + 1$ correspond to the project start activity and to the project end activity, respectively. The activities are interrelated with two kinds of constraints. First, precedence constraints force activity j not to be started before all its immediate predecessor activities are finished. Second, the activities require resources with limited capacities. There is a set of K resource types. While being processed, activity j requires $r_{j,k}$ units of resource type $k \in K$ during every period of its non-preemptable duration, p_j. Resource type k has a limited capacity of R_k at any point in time. The parameters p_j, $r_{j;k}$, and R_k are assumed to be non-negative and deterministic; for the project start and end activities, we have $p_j = 0$, $r_{j;k} = 0$ for all $k \in K$. The objective of the RCPSP is to find both precedence and resource feasible completion times for all activities such that the makespan of the project is minimized. The conceptual decision model of the RCPSP, Kolisch & Hartmann (1999) is given as follows:

$$Min\ FT_{n+1} \tag{1}$$

Subject to

$$FT_i \le FT_j - p_j \quad j = 1, \dots, n+1; i \in Pred_j \tag{2}$$

$$\sum_{j \in A(t)} r_{j,k} \le R_k \quad k \in K; t \ge 0 \tag{3}$$

$$FT_0 = 0 \tag{4}$$

The variable FT_j denotes the finish times of activity j, ($j = 0, 1, \dots, n+1$); and $A(t)$, the set of activities being in progress in period t, is defined as $A(t) = \{j | j = 1, \dots, n, FT_j - p_j + 1 \le t \le FT_j\}$. The objective function (1) minimizes the completion time of the makespan of the project. Constraints (2) take into consideration the precedence relations between each pair of activities (i, j), where $Pred_j$, the set of activities immediately precedes j. Finally, constraint set (3) limits the total resource usage within each period to the available amount. Constraint (4) is to enforce the project to start at time 0.

In order to find a schedule, the decoding procedures, so-called Schedule Generation Schemes (SGSs), are used. SGS generates the schedule, based on the activity list or the priority list, taking into account the availability of the resources and the precedence relationships. SGS starts from an empty set of sequenced activities and constructs a schedule by stepwise extension of a partial schedule. There are two SGS procedures that are considered the core of most RCPSP heuristics: serial SGS (SSGS) and parallel SGS (PSGS). Whereas SSGS performs activity incrementation, PSGS performs time-incrementation. For details, refer to Kolisch (1996b).

The main objective of this paper is to build a genetic algorithm for solving RCPSP. This paper is organized as follows: Section 2 presents a literature review of solving RCPSP using genetic algorithms. Section 3 presents the proposed algorithm, experimental design and default settings. Computational results for validating the algorithm are discussed in section 5. Section 6 is left for conclusions and future work.

2. REVIEW OF GENETIC ALGORITHM LITERATURE FOR RCPSP

Genetic algorithm (GA) was developed by Goldberg (1989) as a computational approach to solve hard problems. It mimics the principles of biological evolution to solve hard optimization problems. Many researchers have developed different GA algorithms for solving RCPSP problem.

Hartmnn (1998) proposed a GA in which he generated the initial population with two ways, randomly and random sampling using latest finish time (LFT) rule and used both serial (SSGS) and parallel (PSGS) generation scheme. He used three representations: activity list (AL), priority rules and priority value. He used three crossover methods: one-point (1PX), two-point (2PX) and uniform crossover (UX); as well as one mutation method, Invert Mutation (INVM) with probability Pm = 0.01, 0.05 and 0.1; and four selection methods: proportional, tournament size 2 (TS-2), tournament size 3 (TS-3) and ranking (RNKS). Three population sizes were used, Ps= 20, 40 and 50. He tested the three representations and found that AL representation gave the best results. He extended his work in Hartmnn (2002). His proposed GA employed the AL representation and the two decoding SSGS and PSGS procedures; two priority rules (LFT, LST (latest start time)) and a random activity selection method were used for generating the initial population. He used two point crossover (2PX), Insert Mutation (INSM), Pm=0.05 and Ranking Selection (RNKS), the Proportional Selection as well as the Tournament Selection (TS), and two population sizes, 40 and 90.

Alcaraz & Maroto (2001) developed a GA which used SSGS and an activity list representation with scheduling mode (forward/backward) (AL-F/B). A schedule was generated using an additional gene which decided whether a forward or backward scheduling needed to be employed (F/B gene). To generate the initial population, they have employed a sampling procedure with

LFT as the selection rule. They have implemented three different selection mechanisms: remainder stochastic sampling without replacement, TS-2 and RNKS; four crossover methods: the precedence set crossover (PPX), one point forward–backward crossover (1PX-F/B), two point forward–backward crossover (2PX-F/B) and 2PX developed by (Hartmann 1998) with probabilities $Pc = 0.5$ and $Pc = 0.8$; two different mutation operators: INSM and INVM with probabilities $Pm = 0.05$ and $Pm = 0.01$; and two population sizes, $Ps = 50$ and $Ps = 100$.

Debels & Vanhoucke (2005) proposed a genetic algorithm which considered two populations and hence was named as Bi-population Genetic Algorithm (BPGA). Both left-justified (forward) which sorted activities in the increasing order of the start time and right justified (backward) which sorted activities in the decreasing order of the finish times population were considered. The default settings of their proposed GA are as follows: randomly generated initial population, SSGS, TS-2 and 2PX, with no mutation.

Franco et al. (2007) have used the following GA: AL representation, randomly generated initial population, SSGS, two crossover methods: 1PX and 2PX with probabilities$Pc = 0.7$, $Pc = 0.3$, INVM mutation and population size 100.

Toni et al. (2008) developed a GA with the following settings: AL representation, randomly generated initial population, two selections: Steady state, tournament size 3 (TS-3), maximum number of generations = 300 or maximum number of consecutive generations without best solution improvement = 50, Uniform crossover (UX) with probability $Pc = 0.5$ and swap mutation (SWM) with probability Pm=0.05.

Cervantes et al. (2008) developed a steady-state genetic algorithm that used a dynamic population, i.e. the algorithm started with a determined number of individuals and as the search is progressing, the size of the population grows. They increased the population size by 25% of the previous population size each time that 1000 new schedules have been evaluated. They used the activity list (AL-F/B) representation, both serial and parallel SGS in two directions forward (F) and backward (B), initial population generated using priority rules, TS-2, 2PX crossover with probability$Pc = 0.8$, and Insert Mutation (INSM) with probability$Pm = 0.05$.

Valls et al. (2008) suggested a hybrid genetic algorithm (HGA) with activity list (AL-F/B) representation, the initial population obtained using the LFT priority rule, serial SGS in two directions: forward and backward, peak crossover (PeX) with probabilities $Pc = 0.75$ and 0.9, INVM mutation with probability $Pm = 0.05$, and RNKS selection. The values of POP size selected were Ps = 24, 50, 100,200, and 400.

Klimek (2010) proposed a GA which used the following settings: AL representation, randomly generated initial population, SSGS, population size $Ps = 50$, roulette wheel selection (RWS) and tournament selection (TS), three crossover operators: 1PX, 2PX and PPX with probability (Pc = 0.7), four mutation operators: INVM, Swap adjacent (SADM), SWM, and INSM with probability $Pm = 0.2$; elite size was equal to 0 (no elitist) or 2 (two elite chromosomes) , and maximal number of generations = 100.

Diana et al. (2013) proposed a GA which used binary-string-based representations, randomly generated initial population with size $Ps = 200$, SSGS decoding procedure, roulette wheel selection (RWS), one point binary crossover (1PX-B) with probability $Pc = 0.95$ and binary mutation (BM) with probability $Pm = 0.95$.

Table 1 shows a summary of different GAs used in solving RCPSP problem. The table summaries each study as follows: authors, publication year, problem representation, how initial population is generated, the SGS employed, the stop criterion, operators and parameters used.

3. Proposed Algorithm

The good performance of a genetic algorithm depends on the selection of a good combination of GA operators and parameters. Based on the literature review and Table 1, four selection methods, five crossover operators and four mutation operators are used in designing different genetic algorithms for solving RCPSP as shown in Table 2.

Table 1: Summary of the genetic algorithms used for solving RCPSP

Author	year	Solution representation (Encoding)	Initial population generation	Evaluation (SGS)	Stop criterion	Operators			Parameters		
						Mutation	Crossover	Selection method	Mutation probability (Pm)	Crossover probability (Pc)	Population Size (PS)
J.Alcaraz and C.Maroto	**2001**	**1. The standard activity list. 2. Activity list with scheduling mode.**	**Priority rule (LFT).**	**Serial**		**1.Insert 2.Invert**	**1- Precedence Crossover. 2- One point F/B Crossover 3- Two point F/B crossover. 4- Two point crossover.**	**1- Remainder stochastic sampling without replacement. 2- Tournament.(2-tour) 3- Ranking.**	**0.05 0.01**	**0.5 0.8**	**50 100**
Mariamar and Antonio etl	**2008**	**Activity list**	**Priority rules**	**Serial, parallel**	**When no improvement is achieved in two consecutive BF iterations**	**Insert**	**Two-point crossover**	**2- Tournament**	**0.05**	**0.8**	**Increasing with 25%**
S Diana,L Ganapathy.etl	**2013**	**Activity list**	**Randomly**	**Serial**	**When number of Generation as 500**	**Binary Mutation**	**One-point Binary crossover**	**Roulette wheel**	**.95**	**.95**	**200**
Marcin Klimek	**2010**	**Activity list**	**Randomly**	**Serial, parallel**	**Maximal number of generations = 5000 schedules**	**1- Invert 2- Swap 1. 3-Swap Adjacent 2. 4-Insert**	**-One-point ,Two point ,Precedence Crossover**	**1-Roulette wheel 2- Tournament**	**0.2**	**0.7**	**50**
Toni Frankola.Marin Golub,etl	**2008**	**Priority value**	**Randomly**		**Maximum number of generations (300) or maximum number of consecutive generations without best solution improvement50**	**swap**	**Uniform vector crossover**	**Steady state, tournament**	**0.05**	**0.5**	**500**
Franco,EG,etl	**2007**	**Activity list**	**Randomly**	**serial**		**Invert**	**One-point, Two point**	**Elitist**		**0.7 0.3**	**100**
Sonke hartmann	**1997**	**Activity list, priority value, priority rule**	**Randomly, priority rules, priority value**	**Serial, parallel**		**Adjacent**	**One point ,Two point, Uniform crossover**	**1-Ranking 2-Proportional**	**0.05,0.01,0.10**		**40**
Sonke hartmann	**2001**	**Activity list**	**Randomly, priority rule (LFT&LST)**	**Serial, parallel**	**Not more than 5000**	**Swap adjacent**	**Two-point**	**Ranking, proportional, tournament**	**0.05**		**40 ,90**
Vicente,francisco,etl	**2007**		**Random sampling (LFT)**	**Serial, parallel**	**Maximum no. of schedules equal5000**	**invert**	**Peak crossover**	**Ranking**	**0.05**	**.9**	**24,50, 100,200 ,400**
Edgar,Fernando,etc		**Activity list**	**Randomly**	**serial**		**Adjacent**	**One-point Two-point**		**0.7**	**0.3**	**100**

Table 2: selection, crossover and mutation methods used in RCPSP

	Method	Code
Selection	Random Selection	RNDS
	Roulette Wheel Selection	RWS
	Ranking Selection	RNKS
	Tournament Selection (tour size = 2)	TS-2
	Tournament Selection (tour size = 3)	TS-3
	Tournament Selection (tour size = 4)	TS-4
	Tournament Selection (tour size = 5)	TS-5
Crossover	One point crossover	1PX
	Two point crossover	2PX
	Uniform crossover	UX
	Peak crossover	PeX
Mutation	Invert Mutation	INVM
	Insert Mutation	INSM
	Swap Mutation	SWM
	Swap Adjacent Mutation	SADM

4.1 Experimental design

For selecting best candidate GA, a full factorial experiment was designed with three operators, selection methods (Sm), crossover methods (Cm) and mutation methods (Mm), as shown in Table 2. Where there are seven selection methods (random, roulette wheel, ranking and tournament with four sizes), four crossover methods and four mutation methods. In addition, two evaluation methods (Ev), SSGS and PSGS, were used. The GA depends also on the following parameters: population size (Ps), crossover probability (Pc) and mutation probability (Pm). The proposed values for these parameters are listed in Table 3. Therefore, the total number of combinations of operators and parameters $= 7Sm \times 4Cm \times 4Mm \times 10Pm \times 11Pc \times 2Ev \times 6Ps = 147840$

Table (3) GA proposed parameter

Parameter/operator	Value/code
Population size	10, 20, 30, 40, 50, 80
No. of generation	5000
Mutation probability	0, 0.02, 0.03, 0.05, 0.07, 0.1, 0.15, 0.2, 0.25, 0.3
Crossover probability	0.0, 0.1,0.2,0.3,0.4,0.5, 0.6,0.7 0.8,0.9,1.0
Evaluation method	SGS, PGS

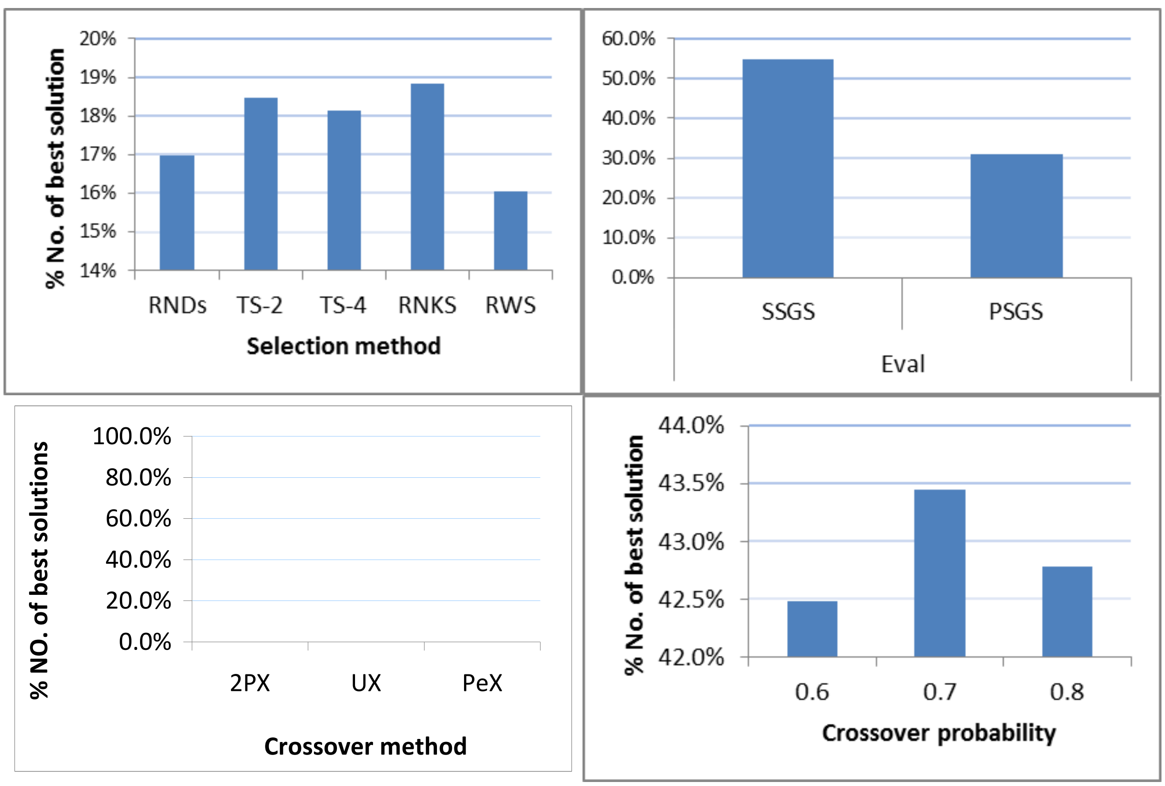
The implementation model of the proposed experiment is coded using C# language (Microsoft Visual Studio 2010). The experiment is applied on the first problem (J30_1.rcp) of J30 set available in the Project Scheduling Problem Library PSPLIB[1]. For the purpose of brevity, the computational results and the statistical analysis show that the following settings give the best minimum deviations from the optimal makespan, five selection methods {RNDS, RWS, RNKS, TS-2, and TS-4}, three crossover methods {2PX, UX, and PeX}, three mutation methods {INV, IINS, and SADM}, three crossover probabilities {0.6, 0.7, and 0.8}, two Evaluations {SSGS and PSGS}, one population size Ps = 50 and three mutation

[1] Download datasets: http://www.bwl.uni-kiel.de/Prod/psplib/

probabilities {0.15, 0.2, and 0.25}. From the results, the second experiment shown in Table 4 is tested on the first ten problems of J60 PSPLIB (60 activities each). 5 runs and 5000 generations per run are used for solving each problem. The performance measure used is the percentage of getting the best known solution as shown in Figure 1. Based on Table 4, we have the following combinations= $5Sm \times 3Cm \times 3Mm \times 3Pm \times 3Pc \times 2Ev \times 1Ps = 810$.

Table (4): Operators and parameters settings for second experiment

Parameter/ operator	Value/Code
Selection method	RNDS, RWS, TS-2, TS-4, RNKS
Crossover method	2PX, UX, PeX
Mutation method	INVM, INSM, SADJM
Crossover probability	0.6, 0.7, 0.8
Mutation probability	0.25, 0.2, 0.15
Population size	50
No. of generations	5000
Evaluation method	SSGS, PSGS



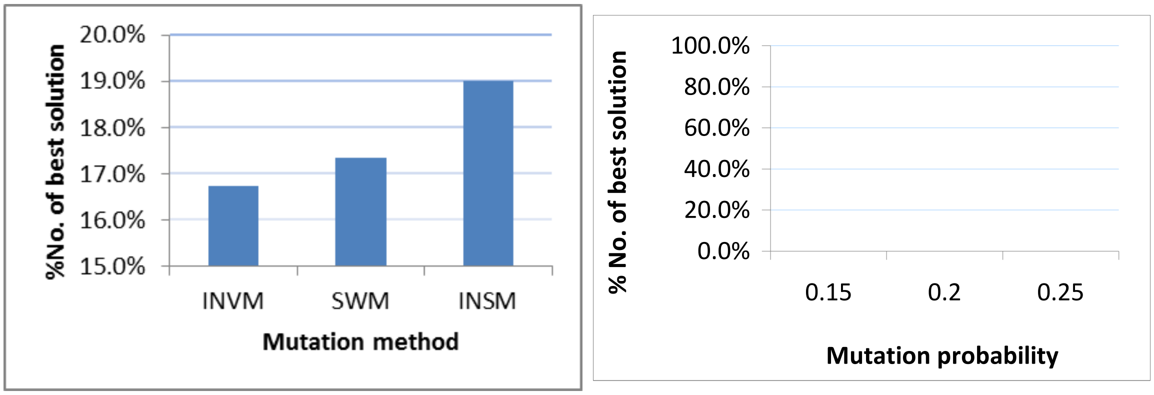


Figure 1: Impact of different GA's operators and parameters

4.2 Impact of GA's operators/parameters

The experiment results are shown in Figure 1 for each operator and parameter mentioned in Table 4. The charts in the figure show the following: for the selection methods, the linear ranking method (RNKS) outperforms the others. However, TS-2 and TS-4 are on average doing well, while roulette wheel method gives the worst performance. The two point crossover (2PX) outperforms the UX and PeX methods and the crossover probability Pc=0.7 gives better performance than the other two values, 0.6 and 0.8. Insert mutation (INSM) is better than INVM and SWM, while the mutation probability Pm = 0.25 is the best. Regarding the evaluation method, it is observed that the SSGS is better than PSGS.

4.3 Default settings

It is clear from the above section that the GA settings shown in Table 5 give the best performance. Therefore, the default settings of these values provide the proposed GA.

Table 5: Proposed GA settings (Default Values)

Operator/Parameter	Type / Value
Selection Method	Linear Rank method(RNKS)
Crossover Operator Method	Two point Crossover (2PX)
Mutation Operator Method	Insert mutation (INSM)
Crossover Probability	0.7
Mutation Probability	0.25
Population size	50

5 COMPUTATIONAL RESULTS

5.1 Test Design

For validating the proposed algorithm, we have taken three standard sets of RCPSP instances from the literature constructed by the project generator ProGen of Kolisch, et al. (1995). These instance sets are open source as mentioned earlier. The first two sets, J30 and J60, contain 480 instances with 30 and 60 activities per project, respectively. The third one, J120, consists of 600 instances with 120 activities. For the purpose of comparison, we have selected 1000, 5000 and 50000 schedules as stopping criteria and the two SGS are applied.

5.2 Computational results

The experiments have been performed on Dell XPS L502X (i7 – 2630 QM CPU 2GHz, 8 GB RAM). The computational results of the three sets are as shown in Table 6. For the J30 set, the results are given in terms of average deviation from the optimal solution. For the other two sets, some of the optimal solutions are unknown. Thus, the average deviation from the well-known critical path-based lower bound is reported. From table 6, it can be seen that the deviation values increase as the size of the problem increase. This means that the RCPSP with larger scale is more difficult to solve since the problem is NP-hard problem. In addition, for each set of problems, the deviation values decrease as the maximum number of schedules increase. It shows that the proposed algorithm can keep finding better results as more schedules are explored. Also, it is observed that the serial SGS outperforms the parallel SGS on small problems and it is doing on average as well as it in large ones.

Table 6: Average deviation (%) from optimal or lower bound makespan for J30, J60 and J120

Problem set	SGS	Max # of Schedule		
		1000	5000	50000
J30 Ave. OPT. Dev	Serial	0.50%	0.18%	0.12%
	Parallel	1.36	1.25	1.13
J60 Ave .LB. Dev	Serial	13.43	12.61	11.77
	Parallel	13.43	13.08	12.6
J120 Ave. LB. Dev	Serial	37.25	34.54	32.03
	Parallel	37.18	34.49	32.93

The results for the three instance sets are also compared with the results of 28 existing algorithms from the literature as shown in Tables 7, 8 and 9 respectively. Each algorithm is briefly described by a few keywords, the SGS employed, and the reference. Also as mentioned above, the results are given in terms of average deviation from the optimal solution for the J30 set shown in Table 7, and in terms of the average deviation from the well-known critical path-based lower bound for the other two sets shown in Tables 8 and 9. The methods are sorted with respect to the results for 50,000 schedules and then for 5,000.

Table 7: Average deviation (%) from optimal makespan — ProGen set J=30

Algorithm	SGS	Reference	Max. #schedules		
			1000	5000	50,000
GA, TS — path relinking	Both	(Kochetov & Stolyar (2003))	0.1%	0.04%	0%
Scatter Search—FBI	Serial	(Debels et al. (2003)).	0.27	0.11	0.01
ACOSS		(Chen et al. (2010))	0.14	0.06	0.01
GAPS		(Mendes et al. (2009))	0.06	0.02	0.01
GA — hybrid, FBI	Serial	(Valls et al. (2008))	0.27	0.06	0.02
GA — FBI	Serial	(Valls et al. (2005))	0.34	0.2	0.02
GA — forw.–backw., FBI	Both	(Alcaraz et al. (2003))	0.25	0.06	0.03
GA — forw.–backw.	Serial	(Alcaraz & Maroto (2001))	0.33	0.12	–
ABC-RK		(Shi et al. (2010))	0.35	0.12	0.04
Sampling — LFT, FBI	Both	(Tormos & Lova (2003b))	0.25	0.13	0.05
TS — activity list	Serial	(Nonobe & Ibaraki (1998))	0.46	0.16	0.05
Sampling — LFT, FBI	Both	(Tormos & Lova (2001))	0.3	0.16	0.07
GA — self-adapting	Both	(Hartman, (2002))	0.38	0.22	0.08
GA — activity list	Serial	(Hartmann (1998))	0.54	0.25	0.08
Sampling — LFT, FBI	Both	(Tormos & Lova (2003a))	0.3	0.17	0.09
Sampling — random, FBI	Serial	(Valls et al. (2005))	0.46	0.28	0.11
THIS STUDY	**Serial**		**0.5**	**0.18**	**0.12**
SA — activity list	Serial	(Bouleimen & Lecocq (2003))	0.38	0.23	–
GA — late join	Serial	(Coelho & Tavares (2003))	0.74	0.33	0.16
Sampling—adaptive	Both	(Kolisch & Drexl (1996))	0.74	0.52	–
GA—random key	Serial	(Kiel & Hartmann (1997))	1.03	0.56	0.23
Sampling—LFT	Serial	(Kolisch (1996b))	0.83	0.53	0.27
Sampling—global	Serial	(Coelho & Tavares (2003))	0.81	0.54	0.28
Sampling—random	Serial	(Kolisch (1995))	1.44	1	0.51
GA—priority rule	Serial	(Hartmann (1998))	1.38	1.12	0.88
Sampling—WCS	Parallel	(Kolisch, (1996a, b))	1.4	1.28 –	–
Sampling—LFT	Parallel	(Kolisch, (1996b))	1.4	1.29	1.13
THIS STUDY	**Parallel**		**1.36**	**1.25**	**1.13**
Sampling—random	Parallel	(Kolisch (1995))	1.77	1.48	1.22
GA—problem space	Mod. par.	(Leon & Ramamoorthy (1995))	2.08	1.59	–

From Tables 7 to 9, it is observed that the proposed algorithm with serial SGS is capable of getting good results compared by the other algorithms. With 50,000 schedules among all the 28 algorithms for J30, J60 and J120, the algorithm with serial SGS ranks 17th, 17th and 7th respectively. In case of J30, the gap between our algorithm and the best one is 0.44% with 1000 schedules, 0.16% with 5000 schedules and 0.12% with 50,000 schedules. For data set J120, our algorithm is the 7th best with 50,000 schedules, where the gap between it and the best one is 3.18% with 1000 schedules, 2.06% with 5000 schedules and 1.47% with 50,000 schedules. As observed, the gap decreases as the number of schedules increases and the order of the algorithm improves as increasing the problem size. So, the proposed algorithm considers an effective and competitive in solving the RCPSP with medium and large scales. In addition, the performance of parallel SGS is not competitive comparing with the rest of algorithms.

Table 8: Average deviations (%) from critical path lower bound — ProGen set J = 60

Algorithm	SGS	Reference	Max. #schedules		
			1000	5000	50,000
ACOSS		(Chen et al. (2010))	11.75%	10.98%	10.66%
Scatter search — FBI	Serial	(Debels et al. (2003)).	11.73	11.1	10.71
GA — hybrid, FBI	Serial	(Valls et al. (2008))	11.56	11.1	10.73
GA, TS — path relinking	Both	(Kochetov & Stolyar (2003))	11.71	11.17	10.74
GA — FBI	Serial	(Valls et al.(2005))	12.21	11.27	10.74
GAPS		(Mendes et al. (2009))	11.72	11.04	10.67
GA — forward–backward, FBI	Both	(Alcaraz et al. (2003))	11.89	11.19	10.84
ABC-RK		(Shi et al. (2010))	12.75	11.48	11.18
GA — self-adapting	Both	(Hartmann (2002))	12.21	11.7	11.21
GA — activity list	Serial	(Hartmann (1998))	12.68	11.89	11.23
Sampling — LFT, FBI	Both	(Tormos & Lova (2003b))	11.88	11.62	11.36
Sampling — LFT, FBI	Both	(Tormos & Lova (2003a))	12.14	11.82	11.47
GA — forward–backward	Serial	(Alcaraz & Maroto (2001))	12.57	11.86	–
Sampling — LFT, FBI	Both	(Tormos & Lova (2001)	12.18	11.87	11.54
SA — activity list	Serial	(Bouleimen & Lecocq (2003))	12.75	11.9	–
TS — activity list	Serial	(Nonobe & Ibaraki (1998))	12.97	12.18	11.58
THIS STUDY	**Serial**		**13.43**	**12.61**	**11.77**
Sampling — random, FBI	Serial	(Valls et al. (2005))	12.73	12.35	11.94
GA — late join	Serial	(Coelho & Tavares (2003))	13.28	12.63	11.94
GA — random key	Serial	(Hartmann (1998))	14.68	13.32	12.25
GA — priority rule	Serial	(Hartmann (1998))	13.3	12.74	12.26
THIS STUDY	**Parallel**		**13.43**	**13.08**	**12.5**
Sampling — adaptive	Both	(Kolisch & Drexl (1996))	13.51	13.06	–
Sampling — WCS	Parallel	(Kolisch (1996a, b))	13.66	13.21	–
Sampling — global	Serial	(Coelho & Tavares (2003))	13.8	13.31	12.83
Sampling — LFT	Parallel	(Kolisch (1996b))	13.59	13.23	12.85
GA — problem space	Mod. par.	(Leon & Ramamoorthy (1995))	14.33	13.49	–
Sampling — LFT	Serial	(Kolisch (1996b))	13.96	13.53	12.97
Sampling — random	Parallel	(Kolisch (1995))	14.89	14.3	13.66
Sampling — random	Serial	(Kolisch (1995))	15.94	15.17	14.22

Table 9: Average deviations (%) from critical path lower bound — ProGen set J = 120

Algorithm	SGS	Reference	Max. #schedules		
			1000	5000	50,000
ACOSS		(Chen et al. (2010))	35.19%	32.48%	30.56%
GA — hybrid, FBI	Serial	(Valls et al. (2008))	34.07	32.54	31.24
GAPS		(Mendes et al. (2009))	35.87	33.03	31.44
GA — forward–backward, FBI	Both	(Alcaraz et al. (2003))	36.53	33.91	31.49
Scatter Search — FBI	Serial	(Debels et al. (2003))	35.22	33.1	31.57
GA — FBI	Serial	(Valls et al. (2005))	35.39	33.24	31.58
THIS STUDY	**Serial**		**37.25**	**34.54**	**32.03**
GA, TS — path relinking	Both	(Kochetov & Stolyar (2003)).	34.74	33.36	32.06
Population-based — FBI	Serial	(Valls et al. (2005))	35.18	34.02	32.81
THIS STUDY	**Parallel**		**34.7**	**34.49**	**32.93**
GA—self-adapting	Both	(Hartmann (2002))	37.19	35.39	33.21
ABC-RK		(Shi et al. (2010))	36.29	34.18	33.69
Sampling—LFT, FBI	Both	(Tormos & Lova (2003b))	35.01	34.41	33.71
GA — activity list	Serial	(Hartmann (1998))	39.37	36.74	34.03
Sampling — LFT, FBI	Both	(Tormos & Lova (2003a))	36.24	35.56	34.77
Sampling — LFT, FBI	Both	(Lova & Tormos (2001))	36.49	35.81	35.01
GA — forward–backward	Serial	(Alcaraz & Maroto (2001))	39.36	36.57	–
TS — activity list	Serial	(Nonobe & Ibaraki (1998))	40.86	37.88	35.85
GA — late join	Serial	(Coelho & Tavares (2003))	39.97	38.41	36.44
Sampling — random, FBI	Serial	(Valls et al. (2005))	38.21	37.47	36.46
SA — activity list	Serial	(Bouleimen & Lecocq (2003))	42.81	37.68	–
GA — priority rule	Serial	(Hartmann (1998))	39.93	38.49	36.51
Sampling — LFT	Parallel	(Kolisch (1996b))	39.6	38.75	37.74
Sampling — WCS	Parallel	(Kolisch (1996a, b))	39.65	38.77	–
GA — random key	Serial	(Hartmann (1998))	45.82	42.25	38.83
Sampling — adaptive	Both	(Kolisch & Drexl (1996))	41.37	40.45	–
Sampling — global	Serial	(Coelho & Tavares (2003))	41.36	40.46	39.41
GA — problem space	Mod. par.	(Leon & Ramamoorthy (1995))	42.91	40.69	–
Sampling — LFT	Serial	(Kolisch (1996b))	42.84	41.84	40.63
Sampling — random	Parallel	(Kolisch (1995)	44.46	43.05	41.44
Sampling — random	Serial	(Kolisch (1995))	49.25	47.61	45.6

6. CONCLUSIONS AND FUTURE WORK

This paper, presents a genetic algorithm based heuristic for the classical resource-constrained project scheduling problem. The computational experiments on a large set of standard test instances have shown that the proposed algorithm leads to better results than several heuristic approachs from the literature. The algorithm is capable of providing near-optimal solutions for a large scale RCPSP. Impact of the project network topology on the performance of serial and parallel SGS will be considered in the future work.

REFERENCES

1. Alcaraz, J., Maroto, C. "A Robust Genetic Algorithm for Resource Allocation in Project Scheduling" *Annals of Operations Research, 102*(1–4), 2001, 83–109. http://doi.org/10.1023/A:1010949931021.
2. Alcaraz, J., Maroto, C., Ruiz, R. "Solving the Multi-Mode Resource-Constrained Project Scheduling Problem with genetic algorithms" *Journal of the Operational Research Society*, *54*(6), 2003, 614–626. http://doi.org/10.1057/palgrave.jors.2601563.
3. Blazewicz, Lenstra, J., Rinnooy Kan A. "Scheduling Subject to Resource-Constraints: Classi cation and Complexity" Discrete Applied Mathematics 5, 1983, 11-24.
4. Bouleimen, K., 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*(2), 2003, 268–281. http://doi.org/10.1016/S0377-2217(02)0076

5. Brucker, P., Knust, S., Schoo, A., Thiele, O. "A Branch & Bound Algorithm for the Resource-Constrained Project Scheduling Problem" European Journal of Operational Research, 1998.
6. Cervantes, M., Lova, A., Tormos, P., Barber, F. "A Dynamic Population Steady-State Genetic Algorithm for the Resource-Constrained Project Scheduling Problem" In International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems, 2008, (pp. 611-620). Springer Berlin Heidelberg.
7. Chen, W., Shi, Y., Teng, H., Lan, X., Hu, L. "An Efficient Hybrid Algorithm for Resource-Constrained Project Scheduling" *Information Sciences*, *180*(6), 2010 1031–1039. http://doi.org/10.1016/j.ins.2009.11.044
8. Coelho, J., Tavares, L. "Comparative Analysis of Metaheuricstics for the Resource-Constrained Project Scheduling Problem" Technical report, Department of Civil Engineering, Instituto Superior Tecnico, Portugal, 2003.
9. Debels, D., De Reyck, B., Leus, R., Vanhoucke, M. "A Hybrid Scatter Search. Electromagnetism Meta-Heuristic for Project Scheduling" *DTEW Research Report 0340*, 2003, 1–21. http://doi.org/10.1016/j.ejor.2004.08.020
10. Debels, D., Vanhoucke, M. "A Bi-Population Based Genetic Algorithm for the Resource-Constrained Project Scheduling Problem" ICCSA, Vol. 4, 2005, 378-387.
11. Diana, S., Ganapathy, L. Ashok K. "An Improved Genetic Algorithm for Resource-Constrained Project Scheduling Problem" International Journal of Computer Applications (0975 – 8887) Volume 78 – No.9, September 2013.
12. Franco, E.G., Zurita, F.T., Delgadillo, G.M. "A Genetic Algorithm for the Resource-Constrained Project Scheduling Problem (RSPSP)" Bolivia Research and Development, Vol.7, 2007, 41–52.
13. Goldberg, D.E. "Genetic Algorithms in Search, Optimization, and Machine Learning" The University of Alabama, Addison Wesley publishing, 1989.
14. Hartmann, S. "A Competitive Genetic Algorithm for Resource-Constrained Project Scheduling" *Naval Research Logistics (NRL)*, *45*, 1998, 733–750. http://doi.org/10.1002/(SICI)1520-6750(199810)45:7<733::AID-NAV5>3.0.CO;2-C
15. Hartmann, S. "A Self-Adapting Genetic Algorithm for Project Scheduling Under Resource-Constraints" *Naval Research Logistics*, *49*(5), 2002, 433–448. http://doi.org/10.1002/nav.10029
16. Hartmann, S., Kolisch, R. "Experimental Evaluation of State-of-the-Art Heuristics for the Resource-Constrained Project Scheduling Problem" European Journal of Operational Research, 127(2), 2000, 394-407.
17. Kiel, D. U., Hartmann, Manuskripte aus den Instituten fur Betriebswirtschaftslehre "A Competitive Genetic Algorithm for Resource-Constrained Project Scheduling 1 Introduction" *Mathematical Programming*, (451), 1997.
18. Klimek, Marcin. "A genetic Algorithm for the Project Scheduling with the Resource-Constraints" Annales UMCS, Informatica. Vol. 10. No. 1. 2010.
19. Kochetov, Y., Stolyar, "A. Evolutionary Local Search with Variable Neighborhood for the Resource-Constrained Project Scheduling Problem" *Workshop on Computer Science and Information Technologies CSIT'2003*, 1–4.
20. Kolisch, R. "Project Scheduling under Resource-Constraints-Efficient Heuristics for several Problem Classes" Physica 14, 1995.
21. Kolisch, R., Sprecher, A., Drexl, A. "Characterization and Generation of a General Class of Resource–Constrained Project Scheduling Problems" Management Science 41 (10) (1995) 1693–1703.
22. Kolisch, R. "Efficient Priority Rules for the Resource-Constrained Project Scheduling Problem" Journal of Operations Management 14 (1996a) 179–192.

23. Kolisch, R. Serial and Parallel "Resource-Constrained Project Scheduling Methods Revisited: Theory and computation" *European Journal of Operational Research*, *90*(2), 1996b, 320–333. http://doi.org/http://dx.doi.org/10.1016/0377-2217(95)00357-6
24. Kolisch, R., Drexl, A. "Adaptive Search for Solving Hard Project Scheduling Problems" *Naval Research Logistics*, *43*(1), 1996, 23–40. http://doi.org/10.1002/(SICI)1520-6750(199602)43:1<23::AID-NAV2>3.3.CO;2-4
25. Kolisch, R., Hartmann, S. "Heuristic Algorithms for Solving the Resource-Constrained Project Scheduling Problem" *Project Scheduling: Recent Models, Algorithms, and Applications*, *14*(1), 1999, 147. Retrieved from http://www.hsba.de/de/pdf/professoren/Hartmann_Hartmann_klosch_Heuristicalgorithms.pdf.
26. Leon, V.J., Ramamoorthy, B. "Strength and Adaptability of Problem-Space based Neighborhoods for Resource-Constrained scheduling" OR Spektrum 17 (1995) 173–182.
27. Mendes, J. J. M., Gonçalves, J. F., Resende, M. G. C. "A Random Key Based Genetic Algorithm for the Resource-Constrained Project Scheduling problem" *Computers & Operations Research*, *36*(1), 2009, 92–109. http://doi.org/10.1016/j.cor.2007.07.001.
28. Nonobe, K., Ibaraki, T. "A Tabu Search Approach to the Constraint Satisfaction Problem as a General Problem" Solver. *European Journal of Operational Research*, *106*(2–3), 1998 599–623. http://doi.org/10.1016/S0377-2217(97)00294-4.
29. Shi, Y. J., Qu, F. Z., Chen, W., & Li, B. An Artificial Bee Colony with Random Key for Resource-Constrained Project Scheduling. Life System Modeling and Intelligent Computing, 2010, 148–157.
30. Toni F., Marin G., Domagoj J. Evolutionary Algorithms for the Resource-Constrained Scheduling Problem. Faculty of Electrical Engineering and Computing, University of Zagreb, 2008, domagoj.jakobovic@fer.hr
31. Tormos, P., Lova, A. A Competitive Heuristic Solution Technique for Resource-Constrained Project Scheduling, Annals of Operations Research 102, 2001, 65–81.
32. Tormos, P., Lova, A. An Efficient Multi-Pass Heuristic for Project Scheduling with Constrained-Resources, International Journal of Production Research 41 (5) (2003a) 1071–1086.
33. Tormos, P., Lova, A. Integrating Heuristics for Resource-Constrained Project Scheduling: One Step Forward. Technical report, Department of Statistics and Operations Research, Universidad Polite´cnica de Valencia, 2003b
34. Valls, V., Ballestín, F., Quintanilla, S. Justification and RCPSP: A Technique that Pays. *European Journal of Operational Research*, *165*(2), 2005, 375–386. http://doi.org/10.1016/j.ejor.2004.04.008
35. Valls, V., Ballestín, F., Quintanilla, S. A Hybrid Genetic Algorithm for the Resource-Constrained Project Scheduling Problem. *European Journal of Operational Research*, *185*(2), 2008, 495–508. http://doi.org/10.1016/j.ejor.2006.12.033