International Journal of Industrial Engineering Computations 2 (2011) 737–748

Contents lists available at GrowingScience

International Journal of Industrial Engineering Computations

homepage: www.GrowingScience.com/ijiec

A new improved genetic algorithm approach and a competitive heuristic method for large-scale multiple resource-constrained project-scheduling problems

Mostafa Khanzadi[a*], Rambod Soufipour[a] and Mohammad Rostami[b]

[a]*Department of Civil Engineering, Iran University of Science & Technology, Tehran, Iran*
[b]*Department of Industrial Engineering, Iran University of Science & Technology, Tehran, Iran*

A R T I C L E I N F O

Article history:
Received 6 May 2010
Received in revised form
June, 28, 2011
Accepted 29 June 2011
Available online
30 June 2011

Keywords:
RCPSP problem
Resource-constrained
Metaheuristics
Genetic Algorithm

A B S T R A C T

The aim of this paper is to present a new genetic algorithm approach for large scale multiple resource-constrained project-scheduling problems (RCPSP). It also presents a heuristic approach to achieve proper solutions for large scale problems. This research area is very common in industry especially when a set of activities needs to be finished as soon as possible subject to two sets of constraints, precedence constraints and resource constraints. The emphasis in this research is on investigating the complexity of scheduling problems and developing a new GA approach to solve this problem in such a way that the advantages of GA are appropriately utilized by applying a novel method to reduce the complexity of the problem. Computational results are also reported for the most famous classical problems taken from the operational research literature.

1. Introduction

The RCPSP is a general scheduling problem that includes precedence and resource constraints and it is considered as NP-hard problem (Philippe et al., 2008). The RCPSP problem presents a very large search space for exhaustive enumeration to attain the optimum solution. Moreover, they would require more computational expense as the problem size grows or additional constraints are added. Practically it is impossible to search the entire search space, which means minimization of makespan becomes tedious and time-consuming. Despite all difficulties to solve RPSP, it has been the focus of a massive research efforts, because even incremental improvements in project scheduling can lead to huge benefits in terms of resource and production time saving (Wall, 1996).

The structure of this paper is organized as follows, in section 2, the related works are reviewed, which consist of exact and heuristic methods. Section 3 defines the problem and its formulation. In section 4, the structure of the proposed genetic algorithm (GA) and its operators are described, in the next section, a novel heuristic capable of achieving near-optimal solutions is proposed. Section 6 includes computational results associated with conventional benchmarks. Finally, section 7 concludes our approach and expresses some more probable application of proposed GA method.

* Corresponding author Tel.: +9821- 77896623 Ex: 3164. Fax: +98-21-77454053
E-mail: m.rostami.iust@gmail.com (M. Rostami)

doi: 10.5267/j.ijiec.2011.06.009

2. Literature review

Solution methods applied to solve RCPSP form two different classes: exact and heuristic methods. Exact methods are involved to find the optimal solutions but they are impractical for large-scale problems having significant number of constraints. Exact methods can be classified into two groups: (1) mathematical programming and (2) enumerative techniques. Methods of solving such problems through mathematical programming are reviewed in Garfinkel and Nemhauser (1976) and Daellenbach and George (1979). They reported that generally mathematical programming (MP) methods require much computation time to solve real-world problems. Additionally, they claimed that MP methods do not consider particular properties to solve all kinds of problems. Consequently, MP problems tend to take longer computations to find a solution than implicit enumeration algorithms designed specifically for a particular class of problems.

Enumerative techniques simply list, or enumerate all possible schedules and then eliminate the non-optimal schedules from the list, leaving those, which are optimal. Branch and Bound search is a typical implicit enumerative method. Conway et al. (1967) introduced the uses of branch and bound in scheduling. Patterson and Huber (1974) suggested two exact bounding algorithms, a minimum bounding algorithm and a maximum bounding algorithm.

In order to solve real world problems appropriately, many researchers have switched to metaheuristics. Basically, finding near optimal results and using limited computation time are the major characteristics of any heuristic algorithm. There are different heuristic approaches which deal with the RCPSP.

Fayer (1990) reported fairly good performance by a simulated annealing approach on scheduling problems. The more important thing is that Fayer's implementation maintained precedence feasibility by restricting the neighborhood operator to only precedence-feasible task swaps. That means any step during the searching process must obey the priority rule. Some researchers reported that they had achieved high-quality project scheduling results by Tabu search (Dell' Amico and Trobian, 1993; Nowicki and Smutnicki, 1996).

Merkle and Schmeck (2002) reported that they could achieve excellent results in project scheduling by an ant algorithm. They presented a new ant colony algorithm procedure (AS-RCPSP) for the project scheduling, which combines the direct (local) and summation (global) pheromone evaluation methods. Actually the only difference between these two pheromone evaluation methods is that the second pheromone evaluation methods gives weighted value to each pheromone value in order to get rid of the local minimum. Furthermore, they discussed the changing strength of heuristic influence, the changing rate pheromone evaporation over the ant generations. In recent years, with developing hybrid meta-heuristic algorithms, there have been employed such methods for RCPSP; specifically one can refer to Valls et al. (2005) and Kim et al. (2003).

In addition, there are several genetic algorithms applied to either MRCPSP or RCPSP. Next part briefly describes their approaches and related methods.

2.1 Previous genetic algorithms

One of first attempts to apply GA in scheduling problems was made by Davis (1985). The main idea of his approach was to encode the representation of a schedule in a meaningful and legal way. In general, most GA operators often produce illegal schedules when they are applied to a scheduling problem. David and Lingle (1985) introduced a new crossover operator (Partially mapped crossover) for the sequence problem. In the same year, Davis (1985) introduced order crossover (OC) for the sequence problem. Two years later, Oliver (1987) developed another new crossover operator called cycle crossover (CC). These special developed crossover operators for the sequence problem will be discussed in the design of our proposed GA. Genetic algorithms are natural candidates for parallel

processing. One approach is splitting the population into subpopulations and assigning a processor to each subpopulation. A standard genetic algorithm is run on each processor and the subpopulations evolve. Periodic migration is permitted when some chromosomes from one subpopulation are transferred or copied to another.

Grefenstette (1986) did some surveys on parallel processing of GA. Some researchers presented their comments about the control parameter setting for GA. However these comments only fit for some particular problems and not for general problems. Grefenstette (1986) believed the parameter setting of a GA should be in a range rather than some particular numbers. He made his comments about the best parameter settings for GA: population size from 30 to 80, crossover rate from 0.45 to 0.95, and mutation rate to 0.01. Based on the computational experience, Kolisch and Padman (1996) carried out a survey to investigate the performances of different algorithms. Their research revealed that the GA approach of Hartmann (2002) has been the best so far. This is the approach that its operations are imitated for our proposed genetic algorithm.

Mendes et al. (2009) blended genetic algorithm with a novel method capable to produce active schedules. They also exploited a new fitness function to enhance their procedure qualities. Agarwal et al. (2011) proposed a Neurogenetic approach, which was a hybrid GA and neural-network (NN) approaches. In their hybrid approach, the search process relied on GA iterations for global search and on NN iterations for local search. Akbari et al. (2011) presented an artificial bee colony as an alternative and efficient optimization strategy for solving RCPSP and investigated its performance on the RCPSP compared with other metaheuristics for solving case studies in the PSPLIB library.

3. Problem definition

Let n be the number of activities with identical durations imposed for execution under precedence constraints. Interruption is not allowed while an activity is being processed. Although there is no constraint to the number of activities being processed, there are constant quantities of resources which are renewable, which means any feasible solution is not allowed to exceed available resources but every activity can be finally executed by being floated to a time period which has sufficient available resources. Logically, r_{ij} indicates the amount of the resource j needed to progress activity i, which is assumed constant during the activity processing time.The LP model can be written as follow (1987).

Parameters:

R_k *: available* k^{th} *resource level*

P_j *: processing time for activity j*

r_{jk} *:the amount of resource*

Decision variables:

F_j *:finishing time of activity j*

LP model:

$$\min z = F_{n+1} \tag{1}$$

$$F_h \le F_j - p_j, j = 1,\cdots,n+1; h \in \text{Pred}(j) \tag{2}$$

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

$$F_j \ge 0, j = 1,\cdots,n+1. \tag{4}$$

The first constraint does not allow activities to start before the finish time of their precedent activities. The second constraint assures that there must be adequate resources needed by activities in each period. The last constraint imposes completion times to be positive.

4. Design of the GA

A new GA approach for RCPSP is developed in this section. In general practices of GA, we should consider: (1) GA's exploration and exploitation ability, (2) the convergence and diversity of the population, and (3) the nature of multiple resources constrained project scheduling problem, especially in the feasibility of each solution (constraints of resource and precedence). In designing of our proposed GA, we will consider these issues and discuss some measures associated with them.

4.1 Encoding and representation of chromosome

For many optimization problems, GA does not operate directly on the solutions for the problems. Instead, they make use of problem-specific representations of the solutions. The genetic operators modify the representation, which is then transformed into a solution by means of a so-called decoding procedure (Hartmann, 2002). First we discuss how to represent the sequence of a set of numbers in the chromosome. Then we discuss how to allocate the resources and start time to each activity. To facilitate the discussion, we use the previous sequence studies as a starting point.

During the last few decades, there have been two main chromosome encoding methods for representing the sequence of a set of numbers: (i) Adjacency, (ii) Path representation (Jean, 1996). However adjacency method is proper for some problems such as traveling salesman problem (TSP), but when it is used for RCPSP, adjacency representation has some disadvantages (Jean, 1996). Therefore, we choose the path representation as the presentation of the chromosome for our proposed GA. We create an activity list that indicates the sequence of activities. After discussing how to represent the sequence of a set of numbers in the chromosome, we need to discuss how to allocate the resources and start time for each activity.

There are two possible schedule generation schemes (SGC): a serial schedule generation scheme and a parallel schedule generation scheme. The serial SGA is constructed from activity lists as follow: First, the activity 1 is started at time 0. Then the activities are scheduled in the order prescribed by the activity list. Thereby, each activity is assigned the earliest precedence and resource feasible start time. In parallel SGC the activity starts at time 0. The difference is that it computes a so-called decision point, which is the time in which an activity to be scheduled is started. This decision point is determined by the earliest finish time of the activities currently in process. For each decision point, the set of eligible activities is computed as the set of those activities, which could be feasibly started at the decision point. The eligible activities are selected successively and started until none of them is left. Then, the next decision point and a related set of eligible activities are computed. This is repeated until all activities are feasibly schedule (Hartmann, 2002). Hartmann also pointed out that the activity list representation together with the serial SGC as decoding procedure leads to better results than other representations for the RCPSP. Based on this research we choose the serial SGC in our proposed GA.

4.2 Crossover

For the RCPSP, a simple crossover reproduction scheme does not work since it makes the chromosomes inconsistent. In other words, some activities may be repeated while others are missed out and hence solutions cannot meet the precedence constraints. The traditional crossover operators like one point crossover is regarded as inappropriate in the study of scheduling problems. The drawback of the simple crossover mechanism is illustrated in Fig. 1.

$$P_1 = [1,2,3,4|5,6,7] \qquad C_1 = [1,2,3,4|5,2,4]$$
$$P_2 = [3,7,6,1|5,2,4] \qquad C_2 = [3,7,6,1|5,6,7]$$

Fig. 1. Illegal solutions created by inappropriate crossover

A simple crossover operator also cannot guarantee the precedence constraints; even no activity is missed out or visited twice. For example, in Fig. 1, we assume, as the precedence constraint, activity 5 must begin after activity 4 is finished. In this case, after the simple crossover, activity 5 maybe begins before activity 4. Except the traditional one or two point crossover, recently two crossover operators were developed for the sequence problem: partially-mapped (PMX) and order (OX) crossovers. However, they cannot guarantee that no activities are missing or visited twice. Sometimes they still break the precedence constraints. The crossover operator used in this paper is derived from them and inherits their merits. According to PMX and OX crossover operator, the crossover operator in this paper is described as follow. There are three main differences among this operator and PMX and OX: (i) Both PMX and OX focus on missed or replicated activities. However they do not consider the precedence constraints. The crossover operator used in this paper can guarantee the precedence constraints. (ii) The crossover operator used in this paper only produces one offspring rather than two. (iii) This crossover focuses on the change between the two cut points rather than outside the two cut points. The pseudo-code of the crossover operator is as follows.

Pseudo-code 1: Crossover operator

Input: parent1,parent2.

Output: child.

Begin

Set *R*= □

Determine randomly x,y,crossover points.

For each a_i in parent1 chromosome

If *i<x or i>y*

Then copy ai in the same position in child's chromosome

Else

R= R U { a_i }

Sort R according to the positions of the associated individuals in parent2's chromosome.

Insert R in vacant space in child's chromosome

End.

Based on the procedure of our proposed crossover operator shown above, let us set the following two parents to show the crossover operation in Fig. 2.

$$P_1 = [1,2,4|5,3,6|7,8]$$
$$P_2 = [1,3,2|6,5,4|7,8] \qquad C_2 = [1,2,4|3,6,5|7,8]$$

Fig. 2. Proposed crossover operator

The crossover of our proposed GA is also similar to Hartmann (2002). However, Hartmann's crossover can create two children, the crossover of this paper only creates one child. Another important thing is that our proposed crossover operator only considers the changes in the middle part of the schedules rather than the whole parts of solution.

4.3 Mutation

The classic mutation operator is also inappropriate in scheduling problems in terms of precedence constraints as well as crossover operators. Mutation used in this paper will first randomly choose 2 activities, and if possible, it swaps them. The reason for using the word possible is that the offspring after mutation may be invalid and our proposed algorithm checks the precedence constraints. This means that if the swap breaks the precedence constraints, the procedure abandons this switch. This procedure continues till the first feasible exchange is found or the number of tries to find it exceeds *k,* if *k* is determined as a big number, we need a significant running time to find feasible swaps, on the other hand, a very small *k* practically eliminates the mutation chance in the population.

4.4 Selection

The parent selection operator used in this paper randomly chooses two individuals and compares them. Consequently, better individual is added to the mating pool. It is known as tournament-selection in literature, which not only chooses better individuals but also gives a chance to those with worse fitness value. For each individual *i,*the fitness function is computed as follow.

$$\text{Fitness value(i)} = {}^{1}\!/_{C_{max}^{i}} \tag{5}$$

Appling crossover and mutation operators lead to a similar-size child population. After that, recent population is combined with the parent population. Logically the best solutions will participate for the next generation.

4.5 Population diversity

A good selection of population size plays an important role for the convergence of the proposed GA method. In this paper, in order to keep the population diverse, not only mutation is applied but also some new individuals are added in each generation. There are either new individuals are as well as others or not, which helps population avoid premature convergence. Although the number of new individuals effectively affects the performance of the GA and the computation time, it seems to be the best way when the mutation scheme is not able to keep diversity in population because of the complexity caused by precedence constraints.

5. Heuristic method

Heuristic methods are usually applied to solve problems, because they are simple and fast enough to be used, commercially.

In this paper a two-stage heuristic method is introduced where in the first stage, it simply calculates the longest path to achieve the project finishing time from the current activity and rates them according the following expression.

$$\alpha_j = p_j + \max\{\alpha_i \mid j << i\}. \tag{6}$$

Note that it is assumed that there is a dummy job with zero progressing time, which must be executed at last with $\alpha_j = 0$.

In the second stage, we schedule activities based on the results of the previous stage. As it is summarized below, it starts with an empty scheduled list U, after that, it lists available jobs at the

current time, which is shown A_t they should respect just precedence constraints. Next step considers resource constraints and schedules activities in the order. If there was any excess requirement to do the activity, the algorithm simply checks the next activity. When there is no feasible activity at current time the time is raised to the next event. The event happens whenever an activity ends and causes a change in resource levels. The following procedure is repeated for all planned activities.

Pseudo-code 2: Crossover operator

MR heuristic approach:

- *Let* $\alpha_j = p_j + \max\{\alpha_i \,|j << i\}$
- *Let* $t = 0, U = \{1,2,\dots,n\}$

1) *While* $U \neq \phi$
2) *Let* $A_t = \{available\ activities\ at\ t\ according\ precedence\ constraints\}$,
3) *Sort* $A_t according\ \alpha_j, descendingly$,
4) *For each* $a_j \epsilon A_t$
5) *If starting* a_j *at* t *does not exceed resource limits*,
6) *Schedule* a_j *and modify* U *unscheduled activities and resources capacity*,
7) *Increase* t *to the nearest resource modifications*,
8) *End*.

6. Computational results

Kolisch and Sprecher (1996) presented a set of benchmark instances for the evaluation of scheduling techniques for the RCPSP called PSPLIB and it is accepted in the literature as the set of benchmark problems. The feature of PSPLIB is that the instances are classified according to various indicators. Either the GA approach or the proposed heuristic developed in this research were coded in C# and have been run on a PC with core i5 2.53 GHz CPU and 4 GB of RAM. Table 1 compares proposed GA with MR heuristic. Note that after 1000s the GA finds the best solution. Consequently, reported results, which have greater running time than 1000s could not complete their search. Although the GA has better results, as the number of activities increases the running time rises dramatically. In this sense MR heuristic has acceptable performance.

Table 1
Comparison between proposed algorithms

Problem	Activities	GA				MR heuristic	
		N = 1000		N = 5000			
		Average deviation	Run time(s)	Average deviation	Run time(s)	Average deviation	Run time(s)
j30	30	0.04	23.90	0.02	119.96	0.9	0.01
J60	60	10.10	102.38	9.94	512.66	3.84	0.06
J90	90	11.02	224.40	10.75	>1000	10.76	1.25
J120	120	27.65	511.95	21.73	>1000	31.31	8.16

In order to distinguish between algorithms those are faster than average deviation is usually computed as follow,

$$\text{Ave. Deviation} = \frac{\sum_{j=1}^{n} D_j - n \times D}{n \times D} \times 100. \quad (7)$$

Parameters:

D_j: the best solution found till j_{th}

n: the number of iterations

D : optimal solution or the best found solution

Fig. 3 and Fig. 4 show the running time for both the proposed GA and the proposed heuristic, respectively. Although both GA and MR heuristic algorithms practically solve problem in exponential running time, the MR heuristic is capable of dealing with larger problems properly.

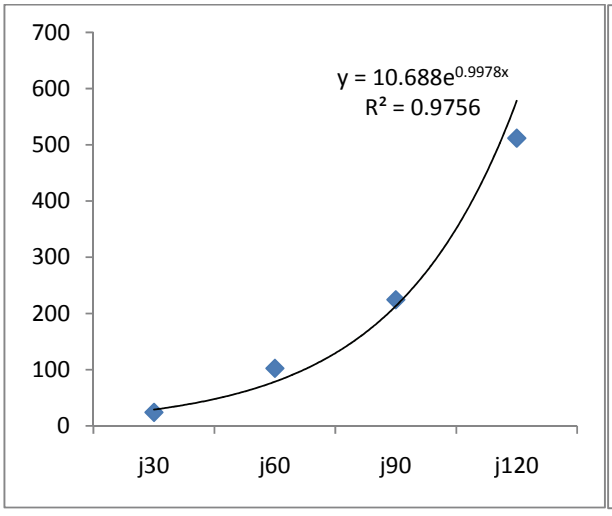

Fig. 3. Proposed GA running time

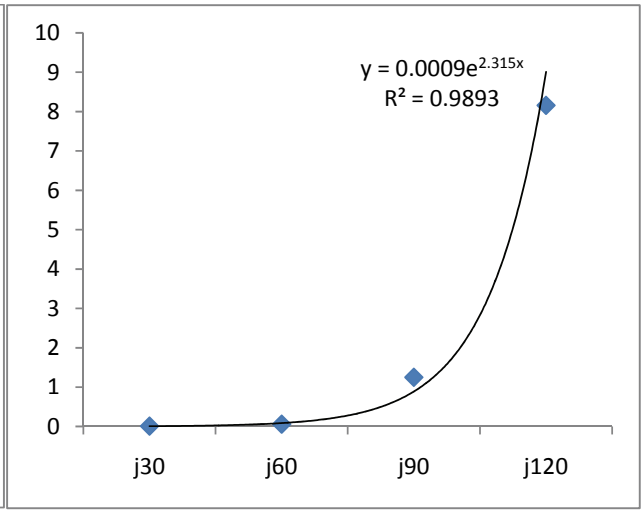

Fig. 4. MR heuristic running time

Table 2 to Table 4 compare our proposed GA with other metaheuristics in the literature. Table 1 reports the results of our proposed method for J30 problems where only Ranjba's (2008) algorithm performs better than our GA when the number of generations is five thousands. Although our GA algorithm used to have better performance but it consumes exhaustive running time to find feasible solutions and add new members to the population. Similarly Table 2 shows the results associated with J60.

Table 2
Average deviation from the optimal solution for J30

Algorithm	Reference	Number of generations	
		N = 1000	N = 5000
		Average deviation	Average deviation
Optimized GA	The proposed GA	**0.04**	0.02
Scatter search, path relinking	Mobini et al., 2009	0.05	0.02
Filter and fan	Ranjbar, 2008	0.09	**0**
Hybr scatter search	Ranjbar et al., 2009	0.1	0.03
GA, TS, path relinking	Kochetov & Stolyar, 2003	0.1	0.04
Decomposition Based GA	Debels & Vanhoucke, 2007	0.12	0.04
Neurogenetic (FBI)	Agarwal et al., 2010	0.13	0.1
Sampling—LFT–FBI	Tormos & Lova, 2003	0.23	0.14
GA—forw.–backw.	Alcaraz et al., 2004	0.25	0.06
HNA—FBI	Colak et al., 2006	0.25	0.11
Sampling—LFT–FBI	Tormos & Lova, 2001	0.25	0.15
GA—hybrid, FBI	Valls et al., 2008	0.27	0.06
Scatter search—FBI	Debels et al., 2006	0.27	0.11
GA—forw.–backw.	Alcaraz & Maroto, 2001	0.33	0.12
GA-FBI	Wall, 1996	0.34	0.2

Because there is not optimal solution for problems when problem size is greater than 30, the results are a lower bound for average deviation from the optimal solution. The PSO algorithm presented by

Tchomte (2007) outperformed other algorithms followed by the proposed GA. For J90 problems, filter and fan search method proposed by Ranjbar (2008) demonstrates the best performance.

Table 3
Lower bound for average deviation from the optimal solution for J60

Algorithm	Reference	Number of generations	
		N = 1000	N = 5000
		Average deviation	Average deviation
Optimized GA	The proposed GA	10.10	9.94
Scatter search, path relinking	Mobini et al., 2009	**9.52**	**9.01**
Filter and fan	Ranjbar, 2008	11.12	10.74
Hybr scatter search	Ranjbar et al., 2009	10.66	10.56
GA, TS, path relinking	Kochetov & Stolyar, 2003	11.59	11.07
Decomposition Based GA	Debels & Vanhoucke, 2007	11.31	10.95
Neurogenetic (FBI)	Agarwal et al., 2010	11.51	11.29
Sampling—LFT–FBI	Tormos & Lova, 2003	12.04	11.72
GA—forw.–backw.	Alcaraz et al., 2004	11.72	11.39
HNA—FBI	Colak et al., 2006	11.56	11.10
Sampling—LFT–FBI	Tormos & Lova, 2001	11.73	11.10
GA—hybrid, FBI	Valls et al., 2008	11.89	11.19
Scatter search—FBI	Debels et al., 2006	12.21	11.27

Table 4
Lower bound for average deviation from the optimal solution for J90

Algorithm	Reference	Number of generations	
		N = 1000	N = 5000
		Average deviation	Average deviation
Optimized GA	The Proposed Method	11.02	10.75
Filter and fan	Ranjbar, 2008	**10.52**	**10.11**
Decomposition Based GA	Debels & Vanhoucke, 2007	10.80	10.35
Neurogenetic (FBI)	Agarwal et al., 2010	11.51	11.29
GA—hybrid, FBI	Valls et al., 2008	NA	10.46

Table 5 presents the lower bound for average deviation from the optimal solution in large instances, i.e. J120. We have generated the performance of all our methods for different algorithms for N=1000 and N=5000.

Finally, Table 6 indicates that our proposed GA has the best performance for large-scale problems and it is observed that it produces massive amount of individuals as well as keeping population diverse could improve the performance of the algorithm for larger problems.

Table 5
Lower bound for average deviation from the optimal solution for J120

Algorithm	Reference	Number of generations	
		N = 1000	N = 5000
		Average deviation	Average deviation
Optimized GA	The proposed GA	**27.65**	**21.73**
Scatter search, path relinking	Mobini et al., 2009	34.49	32.61
Filter and fan	Ranjbar, 2008	32.96	31.42
Hybr scatter search	Ranjbar et al., 2009	33.55	32.18
GA, TS, path relinking	Kochetov & Stolyar, 2003	34.65	34.15
Decomposition Based GA	Debels & Vanhoucke, 2007	36.32	35.62
Neurogenetic (FBI)	Agarwal et al., 2010	36.53	33.91
Sampling—LFT–FBI	Tormos & Lova, 2003	34.94	34.57
GA—forw.–backw.	Alcaraz et al., 2004	35.98	35.30
HNA—FBI	Colak et al., 2006	34.07	32.54
Sampling—LFT–FBI	Tormos & Lova, 2001	35.22	33.10
GA—hybrid, FBI	Valls et al., 2008	35.39	33.24

7. Conclusions

In this paper, we have proposed a new GA method to solve large-scale multiple resource-constrained project-scheduling problems. The proposed model of this paper takes advantages of GA and chooses special designs for specific requirements of the problem such as the representation of the chromosome, the precedence and the resources constraints. We have compared the performance of the proposed model of this paper with other available methods in the literature using some well-known benchmark problems. The proposed GA presented of this paper represented o the chromosome, the crossover and mutation operator in a way to obey the precedence and resource constraints. This means that after crossover, the child solutions could still satisfy precedence constraints. This is a crucial issue for the RCPSP, but many other alternative methods could not handle such a problem. The preliminary results indicated that the proposed model of this paper performed relatively well compared with other existing methods.

References

Akbari, R., Zeighami, V., & Ziarati, K. (2011). Artificial Bee colony for resource constrained project scheduling problem. *International Journal of Industrial Engineering Computations,* 2(1), 45-60.

Alcaraz, J., Maroto, C., Ruiz, R. (2004). Improving the performance of genetic algorithms for the RCPS problem. In: Proceedings of the ninth international workshop on project management and scheduling, 40–43.

Alcaraz, J., Maroto, C. (2001). A robust genetic algorithm for resource allocation in project scheduling. *Annals of Operations Research*, 102, 83–109.

Agarwal, A., Colak, S., & Erenguc, S. (2011). A Neurogenetic approach for the resource-constrained project scheduling problem. *Computers & Operations Research*, 38(1), 44–50.

Christofides, N., Alvarez-Valdes, R., & Tamarit, J.M. (1987). Project scheduling with resource constraints: a branch and bound approach. *European Journal of Operational Research*, 29, 262-273.

Colak, S., Agarwal, A., & Erenguc, S. S. (2006). Resource-constrained project scheduling problem: a hybrid neural approach. In: Weglarz J, Jozefowska J, editors. Perspectives in modern project scheduling, 297–318.

Conway, R. W., Maxwell, W. L. & Miller, L. W. (1967). *Theory of scheduling.* Addison-Wesley, Mass.

Daellenbach, H. & George, J. (1979). *Operation Research Techniques*. Allyn and Bason.

David, E. G. & R. Lingle (1985). Alleles, Loci and the Traveling Salesman Problem, Proceeding of the First International Conference on Genetic Algorithms. Carne-Mellon University, Pittsburg, PA.

Davis, L. (1985). Job shop scheduling with genetic algorithms. Proceeding of the First International Conference on Genetic Algorithms, Carne-Mellon University, Pittsburg, PA.

Debels, D., & Vanhoucke M. (2007). A decomposition-based genetic algorithm for the resource-constrained project-scheduling problem. *Operations Research*, 55(3), 457–469.

Debels, D., De Reyck, B., Leus, R., & Vanhoucke, M. (2006). A hybrid scatter search/ electromagnetism meta-heuristic for project scheduling. *European Journal of Operational Research*, 169(2):638–653.

Dell'Amico, M., & Trobian, M. (1993). Applying tabu search to the job shop scheduling problem. *Annals of Operations Research*, 41, 231-252.

Fayer, F. B. (1990). Some efficient multi-heuristic procedures for resource-constrained project scheduling. *European Journal of Operational Research*, 49 (1), 3-13.

Garfinkel, R. S. & Nemhauser, G. L. (1972). *Integer Programming*, John Wiley, New York.

Gonçalves, J.F. Mendes, J.J., & Resende, M.G.C. (2006) A genetic algorithm for the resource constrained multi-project scheduling problem. *European Journal of Operational Research*, 189,1171–1190.

Grefenstette, J. J. (1986). Optimization of Control Parameters for Genetic Algorithms. *IEEE Transactions on Systems, Man, and Cybernetics*, 16(1),122-128.

Hartmann, S. (2002). A self-adapting genetic algorithm for project scheduling under resource constraints. *Naval Research Logistics*, 49, 433-448.

Jean, Y. P. (1996). Genetic Algorithms for the Traveling Salesman Problem. Annual of Operations Research. 63, 339-370.

Kim, K. W. & Gen, M., & Yamazaki, G. (2003). Hybrid genetic algorithm with fuzzy logic for resource-constrained project scheduling. *Applied Soft Computing* 2 (3F), 174-188.

Kochetov, Y., Stolyar, A. (2003). Evolutionary local search with variable neighborhood for the resource constrained project scheduling problem. In: Proceedings of the third international workshop of computer science and information technologies, Russia.

Kolisch, R., & Sprecher, A. (1996). PSPLIB – A project scheduling library. *European Journal of Operational Research*. 96, 205-216.

Kolisch, R., & Padman, R. (2001). An integrated survey of deterministic project scheduling . *The International Journal of Management Science (omega)*, 29, 249-272.

Mahdi Mobini, M.D, Rabbani, M., Amalnik ,M.S., Razmi, J., Rahimi-Vahed, A.R. (2009). Using an enhanced scatter search algorithm for a resource-constrained project scheduling problem. *Soft Computing*, 13, 597–610.

Merkle, D. M. M.&Schmeck, H. (2002). Ant colony optimization for resource-constrained project scheduling. Evolutionary Computation. *IEEE Transactions*, 6 (4), 333-346.

Nowicki, E. & Smutnicki, C. (1996). A fast tabu search algorithm for the job shop problem. Management Science, 42(6),797-813.

Oliver, I. M., Smith, D.J., & Holland , J. R. C. (1987). A study of Permutation Crossover Operators on the Travelling Salesman Problem, paper presented to Proceeding of the Second International Conference on Genetic Algorithms. Lawrence Erlbaum Association, Hillsadale, NJ.

Patterson, J. H., & Huber, W. D. (1974). A horizon-varying, zero-one approach to project scheduling. *Management Science*, 20 (6), 990-998.

Philippe, B., Claude, L. P., & Wim, N. (2001). *Constraint-based scheduling; applying constraint programming to problems.* Kluwer Academic Publishers, Massachusetts, USA.

Ranjbar, M. (2008). Solving the resource constrained project scheduling problem using filter-and-fan approach. *Applied Mathematics and Computation*s, 201, 313–318.

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

Tchomte, S.K., Gourgand, M., & Quilliot, A. (2007). Solving resource-constrained project scheduling problem with particle swarm optimization. In: Proceedings of fourth multidisciplinary international scheduling conference, 251–258.

Tormos, P., & Lova, A. (2003). An efficient multi-pass heuristic for project scheduling with constrained resources. *International Journal of Production Research*, 41(5), 1071–1086.

Tormos, P., & Lova, A. (2001). A competitive heuristic solution technique for resource constrained project scheduling. *Annals of Operations Research*, 102, 65–81.

Valls, V., Ballestin, F., & Quintanilla, S. (2008). *A hybrid genetic algorithm for the resource-constrained project scheduling problem*. *European Journal of Operational Research*, 185, 495-508.

Valls, V., Ballestin, F., Quintanilla, M.S. (2005). Justification and RCPSP: a technique that pays. *European Journal of Operational Research*, 165(2), 375–386.

Wall, M. B. (1996). *A Genetic Algorithm for Resource-Constrained Scheduling,* Ph.D. thesis, Massachusetts Institute of Technology.