A Modified Genetic Algorithm for Resource Constrained Project Scheduling

Kanchan Joshi
Research Scholar
Indian Institute of Technology, Bombay

Karuna Jain
Professor
Indian Institute of Technology, Bombay

ABSTRACT
An Evolutionary algorithm for resource constrained project scheduling (RCPS) problem is proposed with the objective to minimize project duration (makespan). The algorithm is tested with standard data sets given by Kolisch library for J30 and J60 and found to be performing well.

General Terms
Project Scheduling, Resources.

Keywords
Makespan, Evolutionary Algorithms, GA, RCPSP.

1. INTRODUCTION
Complexity and the challenging nature makes resource constrained project scheduling problem very popular in project scheduling area. There is a tradeoff between resource requirement and project duration (makespan). This tradeoff reveals the challenging nature of the problem. Moreover uncertainty in availability of resources adds to complexity of the problem.

Since decades, various methods are being evolved to solve the complex RCPSP. Early attempts were focused on developing exact methods (LP, IP, MIP) which gained popularity and are the most reliable approach to solve the problem [1]. However their application to the real life problems is difficult due to the size of problems. This led to development of heuristics methods for this class of NP hard problem. Priority rule based heuristic methods are simple to use and hence became popular. However, performance of priority rule depends mainly on characteristics of the project. Hartmann found that metaheuristics perform better than heuristics [2]. Once trapped in local optima, heuristic tends to converge earlier thus showing their inability to explore larger search space. This led to development of better search algorithms like Simulated Annealing (SA), Genetic Algorithm (GA), Tabu Search (TS), Ant Colony Optimization (ACO), Particle Swarm Optimization (PSO). In this paper, an evolutionary approach based Genetic Algorithm (GA) is developed to solve RCPSP with the objective to minimize project duration.

The paper is organized in 5 sections. In section 2, problem definition is discussed. The proposed genetic algorithm is described in section 3. In section 4 computational experiences are presented. Finally, conclusions are presented in section 5.

2. PROBLEM DEFINITION
In this section the mathematical model for makespan minimization objective subject to resource and precedence constraints is discussed. Herroelen model for RCPSP is considered as basic model for proposed GA [3]. The project is represented as activity on node network; G (N, A) where N denotes set of activities and A denotes set of pairs of activities between which there is finish-start precedence relationship. In the project network, first and last activities are dummy activities with zero time duration and resource requirement. There are minimal time lags (0). Each activity requires certain amounts of resources to perform a job. The set of resources is referred to as K. The processing time (duration) of an activity i is referred as d_i, and r_{ik} denotes resources required by activity i of type k. Once started, an activity cannot be interrupted. The RCPSP is formulated as follows.

The notations used for the mathematical model are

n = number of activities in a project
N = Set of activities {1, 2, …, n}
R = set of resources and defined as R = 1, 2 …k
k = number of resource types
d_i= duration of activity i
S_i = start time of activity i
S_j= start time of activity j, where j > i. $\forall$ (i, j) $\in$ A
r_{ik} = amount of resource type k that is required by activity i
a_k = the total availability of resource type k
t = time instance, t $\in$ 1,…, S_{n+1}
P = Set of activities in process in the time period t

The RCPSP model is expressed as

$$\text{Min. } S_{n+1} \qquad (1)$$

such that

$$S_i + d_i \leq S_j \qquad \forall (i,j) \in A \qquad (2)$$

$$\sum_{i \in P} r_{ik} \leq a_k \qquad \forall k \in R \qquad (3)$$

$$S_0 = 0 \qquad (4)$$

Equation (1) gives the objective function of project duration (makespan) minimization, i.e. minimizing the start time of n+ 1 th (dummy) activity. Equation (2) gives the precedence constraint whereas Equation (3) gives resource constraint i.e total number of resources required at time instant t; for all activities in process at t must be less than or equal to the maximum resources available. Equation (4) states that project start at instance zero.

3. PROPOSED GENETIC ALGORITHM
GA is most popular metaheuristics among the researchers due to its efficiency. One of the first attempts to apply GA in the scheduling problem was made by Davis [4]. Drexl and Gruenewald compared GA approach with stochastic scheduling methods and established the supremacy of GA approach [5]. Hartman introduced permutation based encoding strategy i.e the activity list representation in their genetic algorithm. Here the individual schedule is represented by precedence feasible permutation of the set of activities [6].

Kolisch and Padman examined the performance of different algorithms based on their computational experiences [7]. Various other metaheuristics techniques like SA, TS, ACO, PSO techniques have been developed. However the speed and effectiveness of the algorithm while exploring the search space is what distinguishes GA as an efficient algorithm.

GA is a search heuristics that belong to the class of evolutionary algorithm. The evolutionary process starts with population generated by convex combination based heuristics. In GAs, solution to a problem (schedule) is represented as a chromosomes or genotypes. The initial population is composed of such chromosomes. Makespan becomes the fitness function in our case. The fitter chromosomes have higher probability of survival and participate in reproduction process. In each generation, multiple individuals are selected from current initial population. These multiple individuals (chromosomes) are modified to form a new population (off springs) with different traits. Characteristics of these fitter chromosomes are combined together using crossover operator and mutation operator to produce evolved chromosomes called offspring's. Thus chromosomes evolve through successive generations. This process repeats until the termination criterion is met. The termination criteria can be total number of generations or the total solutions generated. The success of the algorithm depends on population size, crossover probability, mutation probability and fitness evaluation function and solution representation scheme. The algorithm for GA is given as:

Step 1: Initialize t = 0, Generate Initial Population (POP). Select the solution representation technique.

Step 2: If t > 1 update population, else go to step 3.

Step 3: Crossover.
Step 3.1: Parent Selection for Crossover.
Step 3.2: Generate Child population (POP') such that
Number of parent schedules = number of child schedules and Number of child schedules consists of equal number of son and daughters.

Step 4: Mutation.

Step 4.1: Set Mutation probability.
Step 4.2: Select Chromosome (Schedule).
Step 4.3: Perform swap mutation such that precedence is not violated.
Step 4.4: Generate mutated schedules.

Step 5: Selection of Next Generation Population.

Step 6: Increment t.

Step 7: Check if termination criteria is achieved. If yes end loop else go to step 2.

3.1 Initial Population

Ideally, the initial population should have a gene pool as diverse as possible in order to be able to explore the larger search space. Generally to achieve this, the initial population is randomly generated and sometimes population is implanted with solutions generated by some kind of heuristic. The advantage of using heuristically developed population element is that the fitness of the population can be improved thereby enabling the algorithm to find good solutions faster. Hartmann suggests that using two good priority rules and a randomized activity selection method leads to a diversified initial population of good activity lists in single mode RCPSP [6]. The larger the population is, the easier it is to explore the search space. But it has been found that the time required by a GA to converge is directly proportional to the population size. The population converges when all the individuals are very much alike (clones). However, when the population size becomes too large, only few generations can be computed within the time limit and thus GA will not give improved results.

While making the choice, for priority rules Kolisch and Hartmann findings were considered and LFT (Latest Finish time), Min SLK (Minimum Slack) and GRPW (Greatest Rank Position Weight) are selected [8]. Using combinatorial considering of these priority rules only 30 unique schedules can be obtained. However larger the number of schedules in initial population pool, better is the solution, hence weighted average approach is used. Weights in the range of [0.1 to 0.9] are assigned and 150 schedules are generated. Thus the convex combination model is a distinguishing method to obtain initial population of precedence and resource feasible schedules. The schedules are ranked in ascending order i.e with increasing project duration (makespan). The first schedule amongst ranked 150 schedules is the one with minimum makespan and is chosen to be the best schedule. Here we get only limited number of unique and feasible schedules. If we increase the number of schedules we get clones and hence there is chance of getting stuck in local optima. Table 1 gives the priority rules used to generate initial population.

Table 1. Priority Rules for generation of Initial Population

Minimum late finish time(LFT)	Priority: min $\{t_i^{LF}\}$, where t_i^{LF} is the latest finish time of activity i from eligible set
Minimum slack (MINSLK)	Priority: min $\{t_i^{LS} - t_i^{ES}\}$, where t_i^{ES} & t_i^{LS} is the early start & late start time respective of the activity i from the eligible set.
Greatest Rank Position Weight (GRPW)	Priority : inverse(sum of duration of all the successors of i + duration of activity i)

3.1.1 Solution Representation

The way in which solutions are represented is crucial for the performance of the algorithm. Kolisch and Hartmann discussed five representation techniques [8].Most popularly being the activity list and random key representation for solutions representation. Goncalves have represented chromosomes using random key generation technique [9]. Activity list representation technique for the chromosomes has been used in the proposed GA.

3.2 Crossover

Crossover combines the feature of two parent chromosomes to form two offspring's which inherit their characteristics. Various crossover techniques stated in literature are single point, two point crossover, partially matched crossover, uniform crossover methods. Two point crossover method has been used by many authors and found to be very effective. The advantage with two point cross over technique is that it will not destroy entire chromosome structure. This will help retain the genetic characteristics of parent population. It is possible to capture the strongest portion of the chromosomes. This will enhance the quality of the solutions in the next generations. Hartmann's two point uniform crossover strategy is modified to incorporate the precedence with temporal

relationship amongst activity to produce feasible offspring's [6].

Next step is the parent selection from the initial population for crossing. For the resource constrained project scheduling the various selection strategy are rank selection, the tournament selection and random selection method. The algorithm uses random selection method for crossover parent selection. The best schedule i.e. the one with minimum time duration (makespan) from initial population is chosen as one of the parent while other parent is selected randomly from the remaining population.

The procedure to perform crossover operation starts with selection of crossover points. Since two point crossover method has been employed, two points, say p and q are generated randomly such that $1 \leq p \leq n$ and $1 \leq q \leq n$ except p+1 and p-1. The position of crossover point p is selected randomly. Then any other point is selected as q except point's p-1, p and p+1. The length of the chromosome string is equal to the number of activities n in the project.

For generation of son (daughter) the best schedule forms the father (mother) while randomly selected schedule forms mother (father). Son (daughter) will be generated by copying father's (mother's) genes from the locations 1 to p. The gene positions from p+1 to q are filled from the mother's (father's) schedule by traversing from left to right such that an activity already existing in the son(daughter) schedule is not selected from mother's (father's) schedule again. The gene positions from q+1 to n are filled from father's (mother's) schedule by traversing from left to right such that an activity already existing is not selected again. This will maintain the precedence relationship among the activities. The crossover technique is used to generate equal number of offspring's as that of parent population. Also, crossover operator in the algorithm is adjusted such that equal number of son and daughter are generated. Thus population pool is generated which consists of parent population (POP) and offspring (POP').

3.3 Mutation

Mutation operator maintains genetic diversity in the population. It modifies the genetic structure thus generating modified diverse population. Swap Mutation technique has been used due to its popularity in the literature. The operator simply selects two activities at random and if the temporal precedence relationship is satisfied it performs a swap, otherwise the operator selects another position randomly [10]. The process continues until swapping is feasible. For RCPSP the mutation probability takes a value in the range 0.02 to 0.05depending upon other genetic operators. The algorithm is tested for various mutation probabilities in the range suggested by literature and it is found that algorithm gives best results for mutation probability of 0.03.

3.4 Next Generation Population Selection

This is the one of the very crucial step in the evolutionary process. It determines the effectiveness of the evolutionary process. The proportions of the population from each pool needs to be carefully chosen so as to maintain diversity while conserving the original characteristics of the parent population. The results were computed considering various combinations of child and parent populations. The extensive computational experiments were performed to determine most effective combinatorial population pool. As the initial population pool consists of limited schedules, child population must form the major proportion of next. This increases the diversity and decreases the chances of getting stuck in local optima. The child population is ranked and for next generation population pools consists of 80 % of the schedules from child population, all the mutated schedules generated and the balance from the best schedules of parent population pool so as to have same number of schedules as initial population for next generation**.**

3.5 Termination Criteria

Number of schedules generated forms the termination criteria. The termination criteria is also based on number of solutions, such that they have either achieved minimum criteria or solutions have reached a plateau where successive iterations no longer produce improved results. The algorithm performs iteration for successive generations until any further improvement is achieved.

4. EXPERIMENTS AND RESULTS

Here, the performance of the proposed GA for RCPSP is compared with other popular solution approaches. For this purpose the results are tested for the standard data sets of RCPSP instances from Project Scheduling Problem Library (Kolisch and Sprecher, 1996) (PSPLIB) [11]. For each instances, the algorithm is tested for problem set that uses four renewable resources. The results are compared with the makespan obtained from PSPLIB for problems dealing with 30 and 60 activities. Percentage deviation is computed as,

$$\%\ deviation = \frac{(makespan_i - optimal\ makespan_i\ )}{(optimal\ makespan_i)}$$

Further Average % deviation is computed for summing up the results. Here the range of parameters Network Complexity (NC)={1.5,1.8,2.1} ; Resource Strength (RS)={0.2,0.5,0.7} and Resource Factor (RF) = {0.25, 0.5, 0.75, 1} are considered as complexity indicators for the problem instances. The resource strength measures the proportion of resource demand and availability. RS is normalized to the interval [0, 1]. Hence, the problem is highly resource constrained for RS = 0 while for RS = 1, the problem is no longer resource-constrained. The third parameter is RF is normalized to the interval [0, 1]. If we have RF = 1, then each activity requests all resources. RF=0 indicates that no activity requests any resources.

The algorithm is tested for J30 as well as J60 project sets for various complexity criteria. In order to select the project instances from data sets, selective random sampling for J30 and J60 is performed. The algorithm is tested for almost 250 project instances for each of them for various complexity cases.

Table 2. Average % Deviation from Kolisch library results

Project	Maximum number of schedules			
	500	1000	5000	10000
J30	1.0	0.9	0.5	0.4
J60	1.8	1.6	1.5	1.5

The table 2 gives the comparative analysis for J30 and J60 projects. It gives the % average deviations of the results from Kolisch data set makespan. For both J30 (1.5 %) as well J60 (0.4%) the deviation is very minimal. With the increase in the number of schedules the deviations reduces depicting the efficiency of the evolutionary mechanism and proximity to the

best available makespan. The results are very sensitive to the resource tightness given by RS and RF factor. The results obtained match with the Kolisch library best available results for loose and moderate resource constraint for most of the instances. However deviation is observed for tight resource constraint. It shows a decreasing trend with increase in number of schedules. Thus with increase in the number of generations the deviation can be reduced to acceptable level. Table 3 illustrates the performance for different resource constraints for J30.

Table 3. Average % deviation for different resource constraints

Resource Constraints	Number of Schedules		
	1000	5000	10000
Loose	0	0	0
Moderate	0.2	0	0
Tight	1.8	1.5	1.3

Further, Table 4 and 5 summarize Average % deviation for various parameters that impact the efficiency of the proposed algorithm for J30. Similar results follow for J60 data sets as well. The algorithm was tested for various mutation probabilities within the range described by the literature to be effective. Various mutation probabilities tested were 0.2 0, 0.33 and 0.6 i.e 3, 5 and 10 mutated schedules respectively. It was observed that 0.33 is the best mutation probability for proposed GA. Table 4 give the performance for various mutation probabilities,

Table 4. Impact of mutation probability

Number of Schedules	Mutation Probabilities		
	0.2	0.33	0.6
500	1.9	1	2.3
1000	1.6	0.9	1.9
5000	1.5	0.5	1.8

Proportion of Child population is another parameter that impacts the proposed significantly. The numbers of initial schedules are limited to 150 hence to increase the diversity it was observed that the proportion of child population should be increased. It was tested for next generation population pool with 80% of child population, equal parent and child population and more number of parent scheduled i.e 80 % of parent population. It is observed that best results are obtained with the greater child population pool for next generation.

Table 5. Impact of Proportion of Child Population for next generation

Number of Schedules	Child Population Proportion		
	0.2	0.5	0.8
500	1.7	1.7	1
1000	1.67	1.65	0.9
5000	1.08	0.9	0.5

When comparing the child population proportion it was observed that the convergence is faster when proportion is 0.5 than 0.2 and best results are achieved due to still earlier convergence when proportion is 0.8.

All the results were compared with best results given by Kolisch library. This gives a comparative analysis of the proposed algorithm with established algorithms considering various techniques. The convex combination heuristics for generation of initial population, parent selection for crossover and the selection technique of the population pool for next generation distinguishes the proposed algorithm.

5. CONCLUSION

The paper discusses various cases of the computational experimentations for the proposed algorithm. The results obtained elaborate the efficiency while comparing with the best available results from Kolisch library. The proposed algorithm is sensitive to resource constrainedness or tightness. While considering loose and moderate resource constraints the algorithm has achieved the best available results of Kolisch library for most of the instances. For tight resource constrains the average % deviation does not exceed 2% deviation which is reasonably acceptable. The algorithm was further tested for parameters like proportion of child population and mutation probability. The algorithm considered multiple resources constraints simultaneously, scenario closer to real life problems. Although the proposed algorithm is tested for 30 and 60 activities it can be tested for 120 activities as well as a future extension. In future we shall enhance the research for cost parameters as well as trade off problems.

6. REFERENCES

[1] Davis, E.W and Patterson, J.H. "A comparison of heuristic and optimal solutions in resource-constrained project scheduling," *Management Science*, 21(8), 1975, 944-955.

[2] Kolisch and Hartmann, "Experimental Investigation of heuristics for resource constrained project scheduling: An update", *European Journal of Operational Research*, 174(1), 2006, 23-37.

[3] Herroelene, Demeulemeester , "Resource constrained project scheduling: A survey of Recent Developments", *Computer Operations Research*, 24, 1997,pp.279-302

[4] Davis, "Job shop scheduling with genetic algorithms," Proceeding of the first International conference on genetic algorithms, 1985, 136-140.

[5] Drexl and Gruenwald, "Non-preemptive multi-mode resource-constrained project scheduling", *IIE Transactions*, 25(5), 1993, 74-81.

[6] Hartmann, "A competitive genetic algorithm for resource-constrained project scheduling", Naval Research Logistics, 45(7), 1998, 733-750.

[7] Kolisch and Padman, "An integrated survey of deterministic project scheduling", *Omega*, 29(3), 2001, 249-272.

[8] Hartmann and Kolisch. , "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.

[9] J.JM. Mendes, J.F. Goncalves and M.G.C. Resende, "A random key based genetic algorithm for the resource

constrained project scheduling problem", *Computers & Operations Research*, vol.36, 2009, pp.92-109.

[10] Masato Watanabe, Kenichi Ida and Mitsuo Gen, "A genetic algorithm with modified crossover operator and search area adaptation for the job-shop scheduling problem", *Computer and Industrial Engineering*, vol.48, 2005, pp.743-752.

[11] Kolisch and Sprecher, "PSPLIB-A project scheduling problem library," *European Journal of Operational Research*, 96(1), 1996, 205-216.

[12] P. Brucker, A Drexl, R. Mohring, K. Neumann, E. Pesch, "Experimental investigation of Resource constrained project scheduling: an update", *European Journal of Operational Research*, Vol. 169, 2009, pp. 638-653.

[13] Hartmann and Briskorn, "A survey of variants and extensions of the resource-constrained project scheduling problem," *European Journal of Operational Research*, 207, 2010, 1-14.

[14] T.Hegazy, "Optimization of resource allocation and leveling using Genetic algorithms," *Journal of Construction Engineering and Management*, 125(3), 1999, 167-175.

[15] Icmeli, Erenguc and Zappe, "Project scheduling problems: A survey*," International Journal of Operations and Production Management*, 13(11), 1993, 80-91.

[16] Lova, Maroto and Tormos, "A multi criteria heuristic method to improve resource allocation in multi project scheduling," *European Journal of Operational Research*, 127, 2000, 408-424.

[17] Kim, Yun, Yoon, Gen and Yamazaki, "Hybrid genetic algorithm with adaptive abilities for resource-constrained multiple project scheduling," *Computers in Industry*, 56, 2005, 143-160.

[18] Valls, V., Ballestin, F and Quintanilla, S. "A hybrid genetic algorithm for the resource-constrained project scheduling problem," *European Journal of Operational Research*, 185, 2008, 495-508.

[19] D. Sundar, B. Umadevi, K. Alagarasamy, "Multi Objective Genetic Algorithm for optimized Resources usage and the Prioritization of the Constraints in the Software Project Planning", *International Journal of Computer Applications*, vol. 3, 2010, 0975-8887.