An Improved Genetic Algorithm for Resource Constrained Project Scheduling Problem

S Diana
Research Scholar
National Institute of Industrial Engineering, Mumbai

L Ganapathy
Professor,
National Institute of Industrial Engineering, Mumbai

Ashok K Pundir
Professor & Dean (Student Affairs & Placement)
National Institute of Industrial Engineering, Mumbai

ABSTRACT
Project scheduling with limited resources is a challenging management problem that is of immense importance to both practitioners and researchers. This problem is known to be NP-hard even under the simplifying assumptions of single renewable resource constraint, its constant availability over time and minimization of makespan as objective. This paper presents an improved Genetic Algorithm (GA) based approach for the single mode resource constrained project scheduling problem (RCPSP) with makespan minimization as objective. The proposed approach uses binary string based representations and operators for chromosomes. The approach was tested on some difficult instances with high optimality gap in the J120 data set of PSPLIB. It was found that the proposed approaches gave better results as compared to activity list based representations that are commonly used.

General Terms
Genetic Algorithm, Project Scheduling, Resources Allocation

Keywords
Resource Constrained Project Scheduling Problem, RCPSP, Genetic Algorithm, Project Makespan, PSPLIB

1. INTRODUCTION
Due to increasing complexity of managing modern businesses that depend on short product life cycles, practitioners have recognized the need to efficiently allocate scarce resources in industrial projects to reduce their completion time. This problem is widely known as the Resource Constrained Project Scheduling Problem (RCPSP). Briefly stated, the RCPSP is concerned with scheduling of project activities considering both the precedence and resource availability constraints. A typical objective in solving this problem is one of minimizing the project makespan, or the time required to complete all the activities in the project. Over the time, various variations of RCPSP have been proposed by researchers with different objective functions, resource requirement and resource availability patterns. The RCPSP is known to be NP-hard even under the simplifying assumptions of single renewable resource constraint, its constant availability over time and minimization of makespan as objective. Due to NP hard status of the problem, one cannot solve for optimality when the number of activities increases. Various approximation methods are available to solve RCPSP. In this paper, an improved version of Genetic Algorithm (GA) is proposed and tested on some of the worse instances of J120 Data set.

The paper is organized as follows. Section 2 presents a problem formulation. Section 3 presents some literature related to Genetic Algorithms for RCPSP. Section 4 gives details of the proposed Genetic Algorithm Section 5 presents the computational results. Summary and Conclusion is presented in Section 6.

2. PROBLEM FORMULATION
The RCPSP consists of n+2 activities J = (0, 1, 2, 3…n+1) where j = 0 and j = n+1 are dummy activities which represent the project start and project end respectively. Renewable resources are of several types k where k = 1, 2… K and each are available in limited quantity R_k at any point in time. Each activity j (j = 1, 2… n) requires a processing time or duration denoted by dj and a certain amount, $r_{j,k}$ of resources type k for completion.

The activities in the project are interrelated by two kinds of constraints namely Precedence constraint and Resource availability constraint. Precedence constraints dictates that an activity j cannot be started until all its immediate predecessors Pj are completed; and the resource availability constraints require that sum of resources needed on any day by all activities scheduled on that day should not exceed availability of any resource type.

As an illustration, R_2 = 5, it means that the availability of resource type 2 is limited to 5 units every day, and if $r_{3,2}$ = 4, it means that job 3 requires 4 units of resource type 2 every day for its completion.

The parameters dj, Rk, and rj,k are assumed to be integer and non negative as is typically assumed in the literature on RCPSP. The decision variables to be determined are the finish times, Fj for each activity j. Thus, the finish time of the last job, n+1, is given by F_{n+1}. The finish time of the last job is also the makespan (total duration) of the project. Thus, the objective of RCPSP is to find precedence and resource feasible completion times of all activities such that the makespan (total duration) of the project, given by the finish time of the last job, F_{n+1}, is minimized.

The conceptual model of RCPSP described by Christofides et al. (1987) is as follows:

$$Min \quad F_{n+1} \tag{1}$$

Subject to:

$$F_i \le F_j - d_j \qquad i \in P_j \tag{2}$$

$$\sum_{j \in A(t)} r_{j,k} \leq R_k \qquad k \in K; t \geq 0 \qquad (3)$$

$$F_j \geq 0 \qquad j = 1,2,3,.....n+1 \qquad (4)$$

The objective function (1) minimizes the total duration (makespan) of the project by minimizing the finish time of n+1th activity. Constraint (2) shows the precedence relationship which doesn't allow activity J to start until all its predecessors, i, are finished. Constraints (3) limit the resource demand imposed by the activities being processed at time t to the available capacity. Constraints (4) shows the finish time to be non – negative.

The difficulty with this formulation is that the conceptual constraint (3) cannot be explicit because it depends on the unknown jobs scheduled at time t. Various time indexed formulations are available to work around this difficulty, but they exponentially increase the size of the formulation.

3. 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. It provides an environment where the solutions in a population continuously crossbreed, mutate and compete with each other in a survival of the fittest strategy, until they evolve into the best solutions. The search for better solutions in the GA based approach is largely context independent and hence they can be readily applied across a variety of situations.

GA considers a population of solutions instead of one solution. After creating the initial population, new solutions are generated by mating two existing solutions (Crossover) or by altering an existing one (Mutation). The fittest of these survive and move on to the next generation by the means of a selection process while the rest are discarded. Fitness value measures the quality of solution, depending on the objective function of the problem to be solved.

Pseudo code of basic Genetic Algorithm

```
Begin Genetic Algorithm
    Generate initial population with random parents
    Evaluate each parent in the population
    For (j =1 to NGEN)
        Apply crossover with probability Pc on parents
        Apply mutation with probability Pm on child
        Add children to the current population
        Retain best parents
        Evaluate the new generation
    End For
End
```

Researchers have developed different representation schemes, genetic operators (crossover and mutation) and algorithms to solve the RCPSP. Hartmann (1998) developed an algorithm based on permutation of activities in an Activity List representation. A Serial Schedule Generation Scheme (SGS) is employed to generate a precedence and resource feasible active schedule using one and two point crossover techniques.

GA proposed by Kohlmorgen et al. (1999) uses the concept of Island Model and studied the variants of parallel Genetic algorithm. A random key representation was employed which uses a real values between 0 and 1 for each task. A two point standard crossover was applied to the representation.

Lee and Kim (1996) tested three different algorithms namely Simulated Annealing (SA), Tabu Search (TS) procedure and a GA based on the random key representation. A parallel SGS was used to develop an active schedule. A different variant of pair wise interchange for SA and TS was applied and for GA one point crossover was employed.

Alcaraz and Maroto (2001) developed a GA which used a serial SGS and activity list representation. A schedule is generated using an additional gene which decided whether a forward or backward scheduling needs to be employed. During the crossover stage child's activity can be generated by using either forward or backward scheduling. Alcaraz et al. (2004) (c.f. Kolisch and Hartmann, 2006) extended the genetic algorithm of Alcaraz and Maroto (2001). SGS is generated based on an additional gene (Hartmann, 2002) and the forward backward improvement of Tormos and Lova (2001).

Coelho and Tavares (2003) introduced a crossover operator called Late join Function Crossover and they employed Serial SGS and activity list representation. The late join crossover creates an offspring by copying the father chromosomes and then swapping the adjacent pair from the mother in reverse order.

Hindi et al. (2002) proposed a crossover technique similar to Hartmann (1998) which preserved the order thus not violating the precedence constraint. The main difference between the two was in the way the initial population is generated. A serial SGS and Activity list representation were used.

GA developed by Toklu (2002) is applied on the schedules directly and a penalty function is used to evaluate the violation of constraints as infeasible schedules may be generated.

Debels and Vanhoucke (2005) proposed a genetic algorithm which considers two populations and hence named as bi-population Genetic algorithm (BPGA). Both left-justified, (forward) which sort activities in the increasing order of the start time and right justified (backward), which sort activities in the decreasing order of the finish times population are considered. The above method thus took into account features of both Forward/Backward scheduling local search technique.

Debels and Vanhoucke (2007) suggested a Decomposition-Based Genetic Algorithm (DBGA) for RCPSP. This method divides the RCPSP problem into smaller problems and obtained the solution for the problem by combining the solution of sub problems. It was shown that the decomposition based approach finds satisfactory near-optimal solutions and gives a better solution.

Franco et al. (2007) developed algorithms in which initial population was generated based on different features like staring date, ending date and makespan. The algorithm used two point crossover and Serial SGS.

Kim and Ellis (2008) presented a permutation-based elitist genetic algorithm and observed the performance of the algorithm on large-sized projects. Initial population is generated randomly and then Elitist selection mechanism is used to discard the worse solutions thus keeping just the best solutions in the population.

Valls et al. (2008) suggested a hybrid genetic algorithm with activity list representation and which makes use of forward–backward improvement of Valls et al. (2005). They developed a cross over scheme (Peak Crossover where instead of random generation of the crossover point, properties of the schedule were used when combining the parents. During first phase parts of the parent schedule corresponding to peaks in the resource usage is inherited. During the second phase evolution is carried out using the neighbors, corresponding to best individual generated from first phase. The neighbors are constructed with the approach used in Valls et al. (2003).

Goncalves et al. (2009) proposed a random key based genetic algorithm which used the concept of parameterized active schedule (Goncalves et al., 2005). In the first phase priorities are given to the activities and delay time and in the second phase these priorities are used to construct a parameterized active schedule. Instead of one point or two point crossover parameterized uniform crossover (Spears and Dejong, 1991) was employed.

Mehdi and Fariborz (2009) developed a Genetic algorithm with new crossover strategy which uses combination of order crossover and partially mapped crossover. The crossover is similar to Hartmann (2002) but it creates only one child.

Goncalves et al. (2011) developed a biased random key genetic algorithm with forward-backward improvement. Active schedule was created using serial scheduling scheme based on the priorities of activities and then schedule was improved using forward backward improvement procedure. After schedule improvement gene adjustment was carried out to reflect the changes in priorities. In this study instead of using gene by gene mutation, some new individuals are introduced in the next generation.

Agarwal et al. (2011) proposed an approach in which GA and NN iterations are interleaved, feeding their best solutions to each other alternately. The study shows that hybrid approach gives better result than using GA and NN independently.

Liu and Yang (2011) developed a new serial insertion SGS The algorithm builds an active schedule inserting an unscheduled activity inside the partial schedule.

Kim (2012) developed a hybrid genetic algorithm to solve the construction resource constrained project to address the effect of GA parameters like crossover probability, mutation probability, population size and number of generation

Kanchan and Karuna (2012) modified the Hartmann's (1998) crossover strategy and incorporated precedence with temporal relationship amongst activity to produce feasible offspring i.e for the child 1 the best schedule forms the father and randomly selected schedule forms the mother and vice versa for Child2. Then a standard two point crossover is carried out on the selected parents. The initial population was generated using the priority rules Minimum Late Finish Time (LFT), Minimum Slack (MINSLK) and Greatest Rank Position Weight (GRPW).

4. PROPOSED GENETIC ALGORITHM

One of the biggest advantages of Hartmann's approach is that the permutations generated from the crossover operation are guaranteed to be precedence feasible. However, this also restricts the exploratory power of the GA because of limited opportunities for mutation and crossover. Hence, the above RCPSP GA algorithms may not really capture the potential of the schema building blocks proposed by Holland (1975). Holland's GA performs mutation and crossover at bit level. Bit string representations may help prevent premature convergence and give better solutions. For permutation problems like RCPSP, obtaining feasible solutions using bit strings is not simple using the bit string representation.

In case of binary strings, during the crossover stage, parent chromosomes are swapped as shown in Fig. 1. Child 1 is produced by taking initial part of parent 1 and tail part of parent 2. Similarly for child 2, initial part is taken from parent 2 and tail end from parent1. This allows greater freedom in generating children. In case of Hartmann's one point crossover, child 1 is produced by taking initial part from the parent 1 and then the parent 2 list is scanned from the beginning and those activities which are not part of the initial part and considered as shown in Fig. 2. This ensures precedence feasibility of child.

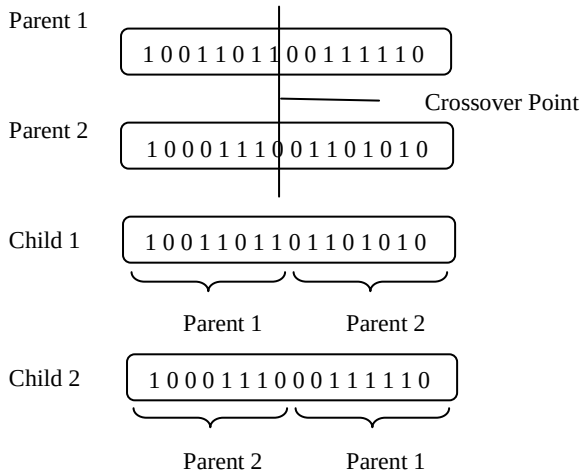

Fig 1: Bit level encoding and Genetic Operations

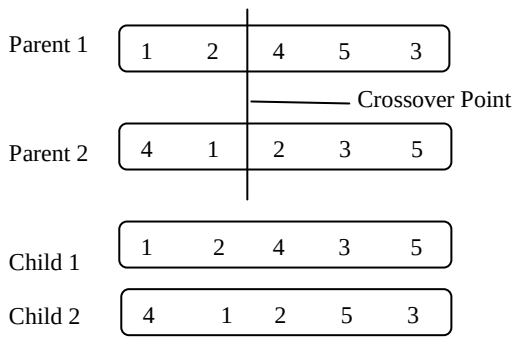

Fig 2: Permutation Encoding and Genetic Operations

The existing Genetic Algorithms based methods proposed by Hartmann (1998) are very powerful for large test problems, and even for some of the worst instances of J120 dataset in PSPLIB, the Genetic Algorithm of Hartmann (HGA) with single point crossover gives solutions within 20 percent of the

upper bound value. This paper presents the variation of the simple Genetic Algorithm of Hartmann (HGA) with different representation scheme and crossover logic to see if better results can be obtained for the worse case instances. Proposed variant of genetic algorithm described below attempts to combine the power of bitwise representation while ensuring precedence feasibility of schedules generated.

4.1 Details of Proposed Genetic Algorithms

The proposed variant of HGA called as DGA; uses the classical binary representation to denote activity priority. In this variant (DGA), the chromosome for each activity has 8 bits and thus can represent up to 255 values of priority. The activity priorities are used to generate the precedence feasible activity list through a roulette wheel selection process. The activity list is then used to generate an active schedule using a serial SGS. The makespan for the schedule is computed and stored as a fitness function for that chromosome. Once the initial population of parents is generated, they are sorted in fitness order. Using an elitist scheme, two best parents are used to obtain new offspring by one point binary crossover of the priority chromosomes. Occasionally a mutation operator is applied to change the existing priority sequence. The offspring are added to the parent pool, and the best of these are retained as parents for the next generation.

Pseudo Code for Generate Parents for DGA

```
For ( i=1 to POPSIZE)
        Generate Binary Random sequence
        Generate Activity Priority using binary random Priority
        Generate Precedence Feasible activity list through Roulette Wheel Selection
        Evaluate each parent in the population
End For
```

4.1.1 Initialization

Priority for each activity is encoded as 8 bit binary string. The string is generated by random assignment of 0's and 1. This can represent upto 255 values of priority. An activity list is generated from precedence feasible candidates based on the priority value using a roulette wheel selection. Using the activity list, an active schedule is generated using SGS and makespan value is calculated.

4.1.2 Crossover and Mutation

Using an elitist strategy, pairs of adjacent parents in the population are selected for crossover. A one point binary crossover is carried out on the priority parents as shown in Fig. 1. The generated children are then converted to precedence feasible activity list and makespan computed as before. Then they are added to the parent population. Occasionally, a binary mutation is used on the child priority sequence.

4.1.3 Selection

The combined pool is then sorted based on minimum makespan value (fitness value) and only the best ones are retained for the next generation.

4.1.4 Termination

The process is repeated for several generations so that convergence is obtained.

5. COMPUTATIONAL EXPERIENCE WITH PROPOSED GA

For carrying out the computational analysis of Genetic Algorithms, a select list of difficult instances from the standard benchmark instances of PSPLIB were identified. The identification is done as follows. Only the problems with large number of activities, that is, those in J120 set are considered for reporting because many of the smaller problem instances in J30 and J60 were readily solved. It is seen from the PSP Library that the 600 instances of J120 data set are grouped into 60 sets of 10 instances each. The possible optimality gap, UB-LB, is calculated for each of the 600 instances. Then the average gap over the 10 instances in each of the 60 instance sets is calculated. The instance sets are then sorted in descending order on average gap as shown in Table 1. Only instances with larger average gaps are considered for the analysis. the Table shows only the top 10 instance sets with higher average gap.

For detailed computational analysis, the worst three instance sets in Table 1, that is, J120_51, J120_56 and J120_31 are taken up. For comparison purpose, results were also obtained by GA coded in C language based on Hartmann's (1998) single point crossover rule which we call as HGA.

Table 1: Average Gap between Upper and Lower Bounds for the 10 instances in each instance set sorted in descending order

Sl.	Instance Set	Average Gap (UB-LB)
1	**J120_51**	**23.5**
2	**J120_56**	**20.1**
3	**J120_31**	**18.8**
4	J120_36	16.1
5	J120_11	16.0
6	J120_16	13.8
7	J120_52	13.3
8	J120_46	12.8
9	J120_26	12.7
10	J120_57	12.0

The proposed GA discussed in the earlier section 4.1 was tested on the instance set j120_51, j120_56 and j120_31 from instance J120 as this set consists of some instances with high gap between the upper and lower bound values for makespan. The instances were solved using the following parameter values: Population size as 200, number of Generations as 500, crossover probability as 0.95 and mutation probability as 0.95. The proposed GA is compared with the HGA and the average makespan obtained from five replications (using different random seeds) and a summary of results is reported in Table 2

Table 2: Comparison of Average Makespan obtained from five different replications

Instance Set J120	Avg. Makespan (HGA)	Avg. Makespan (DGA)
J120_51	253.86	244.26
J120_56	287.48	279.02
J120_31	232.64	225.34

As can be readily seen in Table 2, the modified algorithm, DGA outperform HGA. The solution obtained from the above mentioned GA was better than the one obtained from HGA

In order to look at the best values obtained using different methods, Table 3 shows the smallest value of makespan obtained among the 5 replications. The corresponding percentage deviation of these values from the upper bound is shown in Table 4.

Table 3: Best average makespan value out of the five replications

Instance Set J120	Average UB	Best Avg. Makespan (HGA)	Best Avg. Makespan (DGA)
J120_51	216.0	249.5	242.5
J120_56	251.7	283.8	277.2
J120_31	199.0	229.9	222.9

Table 4: Percent. Deviation of J120 instance set from the upper bound

Instance Set J120	Upper Bound, UB	% Deviation HGA	% Deviation DGA1
J120_51	216.0	15.51	12.27
J120_56	251.7	12.75	10.13
J120_31	199.0	15.53	12.01

6. SUMMARY AND CONCLUSION

The objective of the research was to study the single mode RCPSP with minimization of project makespan as performance measure. As a NP hard problem, it is difficult to solve for optimality when the number of activities in the project increases. This report presents modified GA for resource constrained project scheduling problem (RCPCP) with an objective of minimizing the makespan or total duration of the project.

It can be seen that by using binary encoding of the activity priority, and creating a precedence feasible activity list, one can obtain better results than HGA for some difficult instances in J120 data set. There is thus scope for further tuning these algorithms so as to get better results closer to the upper bound using genetic algorithms.

7. REFERENCES

[1] Agarwal, A., Colak, S., Erenguc, S. “A neurogenetic approach for the resource-constrained project scheduling problem”, Computers and Operations Research, 38, 2011, 44-50.

[2] Alcaraz, J., Maroto, C. “A robust genetic algorithm for resource allocation in project scheduling”, Annals of Operations Research, 102, 2001, 83–109.

[3] Alcaraz, J., Maroto, C., Ruiz, R. “Improving the performance of genetic algorithms for the RCPS problem”, Proceedings of the Ninth International Workshop on Project Management and Scheduling, 2004, 40– 43.

[4] Coelho, J., Tavares, L. “Comparative analysis of meta heuristics for the resource constrained project scheduling problem”, Technical report, Instituto Superior Tecnico at Portugal, 2003.

[5] Debels, D., Vanhoucke, M. “A Bi-population Based Genetic Algorithm for the Resource-Constrained Project Scheduling Problem”, ICCSA, Vol. 4, 2005, 378-387.

[6] Debels D, Vanhoucke M: A Decomposition-Based Genetic Algorithm for the Resource-Constrained Project-Scheduling Problem. Operations Research 2007; 55: 457-469

[7] Franco, E.G., Zurita, F.T., and Delgadillo, G.M. “A Genetic Algorithm for the Resource Constrained Project Scheduling Problem (RSPSP)”, Bolivia Research and Development, Vol.7, 2007, 41–52.

[8] Goldberg, D.E. “Genetic algorithms in search, optimization, and machine learning”, Addison-Wesley, 1989.

[9] Goncalves, J., Mendes, J., Resende, M.G.C. “A hybrid genetic algorithm for the job shop scheduling problem”, European Journal of Operational Research, 167, 2005, 77-95.

[10] Goncalves, J., Mendes, J., Resende, M.G.C. “A random key based genetic algorithm for the resource-constrained project scheduling problem”, Computers and Operations research, 36, 2009, 92-109.

[11] Goncalves, J., Resende, M.G.C, Mendes, J. “A biased random key genetic algorithm with forward-backward improvement for resource-constrained project scheduling problem”, Journal of Heuristics, 17, 2011, 467-486.

[12] Hartmann, S. “A competitive genetic algorithm for resource-constrained project scheduling”, Naval Research Logistics, 45, 1998, 733–750.

[13] Hartmann, S. “Self-adapting genetic algorithm for project scheduling under resource constraint”, Naval Research Logistics, 49, 2002, 433-448.

[14] Hindi, K.S., Yang, H., Fleszar, K. “An evolutionary algorithm for resource-constrained project scheduling”, IEEE Transactions on Evolutionary Computation, Vol.6, 2002, 512–518.

[15] Holland, J. H. “Adaptation in Natural and Artificial Systems”, University of Michigan, Press, Ann Arbor, MI 1975.

[16] Jin lee Kim. "Hybrid genetic algorithm parameter effects for optimization of construction resource allocation problem", Construction research Congress ASCE, 2012.

[17] Kanchan, J., Karuna, J. "A modified genetic algorithm for resource constrained project scheduling problem", International journal of Computer Applications, 57, 2012, 41-45

[18] Kohlmorgen, U., Schmeck, H., Haase, K. "Experiences with fine-grained parallel genetic algorithms", Annals of Operations Research, 90, 1999, 203–219.

[19] Kolisch, R., Hartmann, S. "Experimental investigation of heuristics for resource-constrained project scheduling: An update", European Journal of Operational Research, 174, 2006, 23-37

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

[21] Mehdi, D. and Fariborz, J. "A new efficient Genetic algorithm for project scheduling under resource constraints", World Applied Sciences Journal, 7, 2009, 987-997.

[22] Spears, W.M., and Dejong, K. A. "On the virtues of parameterized uniform crossover", Proceedings of the Fourth International Conference on Genetic Algorithms, 1991, 230-236.

[23] Toklu, Y.C. "Application of genetic algorithms to construction scheduling with or without resource constraints", Canadian Journal of Civil Engineering, 29, 2002, 421–429.

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

[25] Valls, V., Ballestin, F., Quintanilla, M.S. "A Resource Constrained Project Scheduling: A critical activity reordering heuristic", European Journal of Operational Research, 149, 2003, 282–301.

[26] Valls, V., Ballestin, F., Quintanilla, M.S. "Justification and RCPSP: A technique that pays", European Journal of Operational Research, 165, 2005, 375–86.

[27] Valls, V., Ballestin, F., Quintanilla, M.S. "A hybrid genetic algorithm for the RCPSP", European Journal of Oerational Research, 185, 2008, 496-508.