GASolver-A Solution to Resource Constrained Project Scheduling by Genetic Algorithm

Dr Mamta Madan
Professor(Comp science)
Vivekananda Institute of Professional Studies,
Affiliated to GGSIPU AU-Block Pitam PuraDelhi, India

Mr Rajneesh Madan
Architect,
NIIT Technologies Ltd., Gurgaon-11

***Abstract*-The Resource Constrained Scheduling Problem (RCSP) represents an important research area. Not only exact solution but also many heuristic methods have been proposed to solve RCPSP (Resource Constrained Project Scheduling Problem). It is an NP hard problem. Heuristic methods are designed to solve large and highly Resource Constrained software projects. We have solved the problem of resource constrained scheduling problem and named as GASolver. It is implemented in C# using .net platform. We have used Dependency Injection to make the problem loosely coupled, so that other arena of scheduling like Time Cost Tradeoff (CT), Payment Scheduling (PS) etc can be merged with same solution in the future. We have implemented GASolver using Genetic Algorithm (GA).**

Keywords-Genetic Algorithm; Dependency Injection; GASolver.Core; Resource Constrained Scheduling.

I. INTRODUCTION

The Resource Constrained Project Scheduling Problem represents an important research problem. Not only exact solution but also many heuristic methods have been proposed to solve RCPSP. It being an NP hard problem, Alcaraz and Maroto [5] mentioned that the optimal solution can only be achieved by exact solution procedures in small software projects, usually with less than 60 activities, which are not highly resource constrained.

Therefore heuristic methods are designed to solve large and highly Resource Constrained software projects. Mohring [6] mentioned that RCPSP is one of the most intractable problems in operations research and many latest optimization techniques and local search were applied to solve it. We have solved the problem of resource constrained scheduling problem and named as GASolver. It is implemented in C# using .net platform. We have used Dependency Injection (DI) to make the problem loosely coupled, so that other arena of scheduling like Time Cost Tradeoff, Payment Scheduling etc can be merged with same solution in the future. We have implemented GASolver using Genetic Algorithm. The problem statement is explained in the following section. *Problem Statement for RCPSP (Resource Constrained Project Scheduling Problem)*

What is the best way to assign the resources to the activities at specific times such that all of the constraints are satisfied and the best objective measures are produced?

II. GENETIC ALGORITHM

Genetic algorithms (GAs) are search algorithms that are conceptually based on the methods that living organisms adapt to their environment. These methods, known as natural selection or evolution, combine the concept of survival of the fittest among string structures with a structured yet randomized information exchange to form a search algorithm with some of the innovative flair of human search. In each generation, a new set of string structures is created from (bits and pieces of) the fittest strings from the previous generation and occasionally a randomly altered new part. This process of exploiting historical data allows the GA to speculate on new search points that will improve performance thus producing better solutions. Genetic algorithms were initially developed by JohnH.Holland, a professor of psychology and computer science at the University of Michigan. As an optimization tool, the Genetic Algorithm attempts to improve performance leading to an optimal solution.

In this process, there are two distinct steps, (1) the process of improvement and (2) reaching the optimum itself. Of these two steps, the most important is the process of improvement. In complex systems, due to the potential high costs involved, reaching the optimum solution may not be justified as long as continuous improvement is being made and an optimal (desirable) solution can be found.

Genetic algorithm (GA) [1][2][3] is a pioneering method of metaheuristic optimization which originated from the studies of cellular automata of Holland in the 1970s. It is also known as an evolutionary algorithm and a search technique that copies from biological evolution. In Genetic Algorithm, a population of candidate solutions called individuals evolves toward better solutions from generation to generation.

ADVANTAGES OF GENETIC ALGORITHM

- GA can quickly scan a vast solution set. Bad proposals do not affect the endSolution negatively as they are simply discarded.
- The inductive nature of the GA means that it doesn't have to know any rules of the problem - it works by its own internal rules. This is very useful for complex or loosely defined problems.
- They efficiently search the model space, so they are more likely (than local optimization techniques) to converge toward a global minima.
- There is no need of linearization of the problem.
- There is no need to compute partial derivatives.
- More probable models are sampled more frequently than less probable ones

III. Related Work

In 1978, Stinson et al.[9] formulated the multiple resource-constrained scheduling problem as an integer programming problem and advanced a branch-and-bound algorithm for solving it. The algorithm they developed was similar to branch-and-bound algorithm with differences in the node selection heuristics employed and the number of resources handled (Johnson's algorithm allows for a single resource). In their algorithm, branching corresponds to creating new partial feasible schedules from given partial feasible schedules.

An experimental investigation was completed in 1988 by Dumond and Mabert[4]. They studied RCPSP in an environment where new software projects arrive continuously or randomly to a system in which software projects share common resources and receive completion deadlines. Dumond and Mabert tested the performance of four due date procedures and five scheduling heuristics with full control on the due date assignment. A second test was conducted to examine the performance of the due date procedures when deadlines were set externally. Their experimental results failed to indicate a rule that uniformly outperformed the others.

In 2006 an improved Particle Swarm Optimization (PSO) algorithm[10] for resource-constrained software project scheduling problem was proposed. Improvements based on the basic PSO include: the particle swarm is initialized by heuristic rule to improve the quality of particles; inertia weight was self-adapted with iteration of the algorithm to decelerate the speed of particles; crossover mechanism of genetic algorithm were applied to particle swarm to enable the exchange of good characteristics between two particles.

Computational results for software project instances of PSPLIB demonstrate that this improved PSO was effective as compared with other mataheuristic approaches

In 2007 YanLiu presented a fuzzy genetic algorithm for software project scheduling problem with resource constraints and uncertain activity duration [11]. The objective of this research was to minimize the fuzzy software project make span. Firstly, fuzzy set was used to represent the uncertainty of activity duration and the corresponding comparison method of fuzzy number called integral value approach was introduced. Second, three genetic operators were used to search for an approximate shortest software project make span. Therefore, this study provided another metaheuristic method for solving resource-constraint software project scheduling problem with uncertain activity duration.

In 2009 itself Mohammad Amin Rigi, Shahriar Mohammadi K. N. Toosi [8] proposed a new evolutionary approach to resource constrained software project scheduling problem. Hybrid genetic algorithm (GA)-constraint satisfaction problem (CSP) has been applied to solve resource constrained software project scheduling (RCPS). GA's task was to find the best schedule. Their approach has used CSP in order to overcome the existing inconsistencies in activities precedence and resources conflicts. A full state CSP with min-conflict heuristic has been used for solving precedence conflicts and a simple iterative CSP is used to resolve the resource conflicts.

A more realistic resource-constrained software project-scheduling was solved in 2010[7]. A model that is applicable to real-world software projects, with discounted cash flows and generalized precedence relations is investigated under inflation factor such that a bonus–penalty structure at the deadline of the software project is imposed to force the software project not to be finished beyond the deadline. The goal was to find activity schedules and resource requirement levels that maximized the net present value of the software project cash flows. A Genetic Algorithm (GA) is designed using a new three-stage process that utilizes design of experiments and response surface methodology. The results of the performance analysis of the proposed methodology showed an effective solution approach to the problem.

IV. Solution to Resource Constrained Project Scheduling Problem

To implement RCPSP using GA we need to address the following objectives:-

A. method of specifying the relationships between the tasks.

B. description of resources, skill, salary to perform the tasks.

C. The representation of the chromosome.

D. Implementation of selection, crossover and mutation function.

E. Calculation of an objective function to evaluate the best schedule and optimal cost.

F. Class Diagram and Implementation Details of RCPSP.

A. A method of specifying the relationships between the tasks

A project is best represented as a Task Precedence Graph (TPG).A TPG is an acyclic directed graph consisting of a set of tasks and a set of precedence relationships. With the help of Task Precedence Graph we will be able to set the precedence for each task. The Task Precedence graph is shown below in the form of Table.

TABLE I. Task Precedence Graph For RCPSP

	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12
T1	0	0	1	1	0	0	0	0	0	0	0	0
T2	0	0	0	0	1	0	0	0	0	0	0	0
T3	0	0	0	1	0	0	0	0	0	0	0	0
T4	0	0	0	0	0	0	1	0	0	0	0	0
T5	0	0	0	0	0	0	0	0	0	0	0	0
T6	0	0	0	0	0	1	0	1	1	0	0	0

The task precedence graph describes that Task T1 and T2 are not dependent on any task although for task T3 to finish, Task T1 should be completely finished. Similarly for task T4 to complete, Task T1 and T3 should finish and so on. Thus Task

Precedence Graph enables us to set the precedence for various tasks and maintains the order of execution of tasks.

B. Description of resources, skills, salary to perform the tasks

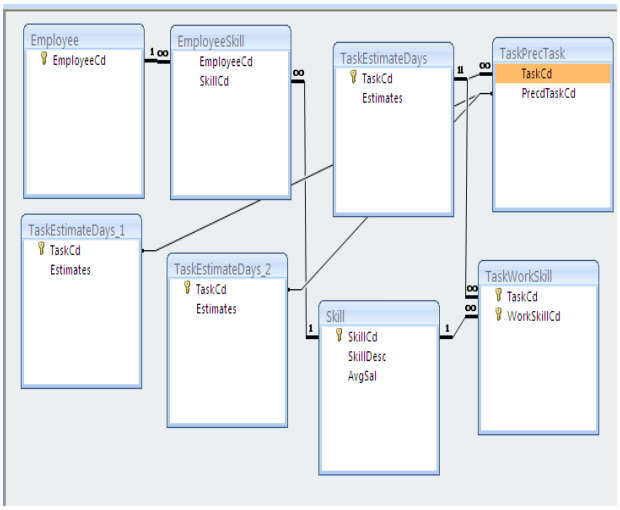

Fig. 1. Data Descriptions for RCPSP

Figure 1 above demonstrates the description of relationship for Resource Constrained database.

C. Representation of the chromosome for RCPSP

Before Implementation any application with genetic algorithm, the most important part of genetic algorithm is to decide the structure for a genome. The genome is an essential part of genetic algorithm as it will be generated randomly. The genome for our problem is a two-dimensional array consisting of employees and tasks. We will randomly generate the employees who can work on these tasks as the random numbers generated between 0 and 1. The chromosome structure is shown below.

TABLE II. CHROMOSOME STRUCTURE FOR RCPSP

	T1	T2	T3	T4
Emp1	0	1	0	1
Emp2	1	0	1	0
Emp3	1	0	0	1
Emp4	0	0	1	1

D. Design of operators for Genetic Algorithm

The three critical functions of genetic algorithm are *selection, crossover and mutation*. These are to be designed for a specific problem. We have designed these operators for RCPSP and they are explained below.

SELECTION:

We initially generate a 2 dimensional array of the above mentioned genome of employee who can work on various tasks. First we check the validity of genome by checking the following

1) *Have obeyed the task precedence relationship*
2) *Have fitness better than death fitness variable*
3) *Obeys employee skill matrix*

Here the death fitness variable signifies the fitness of the genome. If the fitness of the genome is -1 , (value of death fitness) then the genome is an invalid genome. We calculate the fitness of the genome. We select only those genome which are good reproducers i.e. which can reproduce. If the fitness is better, only then it will reproduce otherwise it is removed from the genome list and if it is able to reproduce it will be added to the list of genomes which will be further utilized for crossover and mutation. This way we will be able to select the genomes which have the capability to reproduce further.

Crossover

The crossover operator mimics the way in which bisexual reproduction passes along each parents good genes to the next generation. Normally, two parents Genomes create two new offspring Genomes by combining their “genes” using one point crossover. Let’s take an example of two genomes which are successfully randomly generated and passed the first operator of genetic algorithm.

Before crossover

Randomly we have chosen two genomes from the list of selected genomes which can reproduce well. They are represented as Genome1 and Genome2.

TABLE III. GENOME 1

	E1	E2	E3	E4	E5	E6	E7
T1	1	0	0	0	0	0	0
T2	0	0	0	0	0	0	1
T3	1	0	0	0	0	0	1
T4	1	1	1	0	1	0	0
T5	1	1	1	0	1	0	0
T6	1	1	1	0	1	0	0

The crossover can be performed row wise as well as column wise. Let’s take we have randomly generated the row wise crossover point as 2. So we will swap genome 1 and genome2 after T2, the two baby genomes will be as follows:

After Crossover

TABLE IV. GENOME 2

	E1	E2	E3	E4	E5	E6	E7
T1	1	0	0	0	0	0	0
T2	0	0	0	0	0	0	1
T3	1	0	0	0	0	0	1
T4	1	1	1	0	1	0	0
T5	1	1	1	0	1	0	0
T6	1	1	1	0	1	0	0

TABLE V. BABY GENOME1

	E1	E2	E3	E4	E5	E6	E7
T1	1	0	0	0	0	0	0
T2	0	0	0	0	0	0	1
T3	1	0	0	0	0	0	1
T4	1	1	1	0	1	0	0
T5	1	1	1	0	1	0	0
T6	0	0	1	1	1	1	1

We have thus four genomes after crossover. They are Genome1, Genome2 and two 2 baby genomes. They will be sorted according to the fitness values and best of the two are stored in the list of genomes. We have used row wise crossover, we will also perform column wise crossover. We will generate a random number and based on that random we will decide for row wise or column wise crossover operator.

TABLE VI. BABY GENOME 2

	E1	E2	E3	E4	E5	E6	E7
T1	1	0	0	0	0	0	0
T2	0	0	0	0	0	0	1
T3	1	0	0	0	0	0	1
T4	1	1	1	0	1	0	0
T5	1	1	1	0	1	0	0
T6	1	1	1	0	1	0	0

MUTATION

Following the crossover operator the offspring may be mutated by the mutation operator. Mutation is basically to get some variation in the result. Similar to random mutation in the biological world, this function is intended to preserve the diversity of the population, thereby expanding the search space into regions that may contain better solutions. Here for problem of Resource Constrained Project Scheduling, we have a two dimensional array of genome.

TABLE VII. GENOME BEFORE MUTATION

	E1	E2	E3	E4	E5	E6	E7
T1	1	0	0	0	0	0	0
T2	0	0	0	0	0	0	1
T3	1	0	0	0	0	0	1
T4	1	1	1	0	1	0	0
T5	1	1	1	0	1	0	0
T6	1	1	1	0	1	0	0

TABLE VIII. GENOME AFTER MUTATION

	E1	E2	E3	E4	E5	E6	E7
T1	1	0	0	0	0	0	0
T2	0	0	0	0	0	0	1
T3	1	0	0	0	0	0	1
T4	1	1	0	0	1	0	0
T5	1	1	1	1	1	0	0
T6	1	1	1	0	1	0	0

We randomly pick a genome. We randomly generate an index value, pick any array value randomly, e.g. (4, 3). Presently the array value of this cell is one. We flip this value to zero. And index (5,4) which is 0 is flipped to 1. This way we can have variation in the genome results. After mutation we again calculate their fitness and put it in the final list of genomes.

E. Calculation of an objective function to evaluate the best schedule

Our objective is to find a schedule which should finish in minimum duration and should have an optimal cost. Another important objective is that no task should be undone. Our project will not be complete if any of the tasks is left incomplete, so we have maintained a check that no task is left, it should be managed by at least one of the employee. We have made functions like calculate project duration () that is to calculate the duration of the entire project which is shown below with the help of an example. Let say we have chosen this genome to calculate the project duration and project cost which is mentioned below.

TABLE IX. GENOME 1

	E1	E2	E3	E4	E5	E6	E7
T1	1	0	0	0	0	0	0
T2	0	0	0	0	0	0	1
T3	1	0	0	0	0	0	1
T4	1	1	1	0	1	0	0
T5	1	1	1	0	1	0	0
T6	1	1	1	0	1	0	0

TABLE X. TASK VS ESTIMATED DAYS

T1	T2	T3	T4	T5	T6
20	10	15	25	7	10

As mentioned earlier the task and Estimated man days are also stored in the above Table. Based on the Task and Estimate days, we have calculated the Task Duration of Genome as shown in Table 11.

TABLE XI. TASK DURATION FOR RANDOM CHROMOSOME

	E1	E2	E3	E4	E5	E6	E7	Task Duration
T1	1	0	0	0	0	0	0	20
T2	0	0	0	0	0	0	1	10
T3	1	0	0	0	0	0	1	7.5
T4	1	1	1	0	1	0	0	6.25
T5	1	1	1	0	1	0	0	1.75
T6	1	1	1	0	1	0	0	2.5

TABLE XII. PROJECT DURATION

	20	10	7.5	6.25	1.75	2.5
	t1	t2	t3	t4	t5	t6
t1	0	0	1	1	0	0
t2	0	0	0	0	1	0
t3	0	0	0	1	0	0
t4	0	0	0	0	0	1
t5	0	0	0	0	0	0
t6	0	0	0	0	0	0
	20	10	27.5	33.75	11.75	36.25

On the basis of Table 11, the project duration is calculated and shown in Table 12.

Thus the project Duration comes out to be 36.25 Mandays Based on the salary of various skills, Project cost is calculated as the summation of these entire task cost.

F. Fitness Function

Genetic Algorithm mimics the survival of the fittest Principle of nature to make a search process. Therefore, GAs is naturally suitable for maximization problems, minimization problems are usually transformed into maximization problems by some suitable transformation. In general fitness function F(x) is first derived from objective function and used in successive genetic operations. For maximization problems, the fitness function can be considered to be the same as objective function F(x) =f(x). Where F(x) is the fitness function and f(x) is the objective function. For minimization problems:

$$F(x)= 1/(1+f(x))$$

In our case for RCPSP, Since we have to minimize the objective function ,the fitness function will be same as described in the above equation.

Therefore

Fitness= 1/1+functionvalue

Where function value = w1*projectduration+w2*projectcost.

Where w1 and w2 are the weights attached to project duration and project cost respectively. Depending on which is more crucial for our organization whether cost or duration we can decide the weights. If we have to give more weight to duration, the weight of duration will be increased and similarly for cost also and vice versa.

So we compare the fitness of this genome, with the previous generated genome and iterate this process and generate various generation and looks for best fitness that can be achieved. The best fit genome will be displayed by the console application.

V. CLASS DIAGRAM AND IMPLEMENTATION DETAILS

A. GASolver.Core

GASolver .Core is the main component of the solution. It is responsible for implementing all the three operators namely selection, crossover and mutation on various generations. It also provides a contract IGenome to be implemented in different genomes who wish to use GASolver for optimizing their problem. Following is the class diagram of GASolver.Core. It has a population (generation) class which essentially is collection of similar genomes.

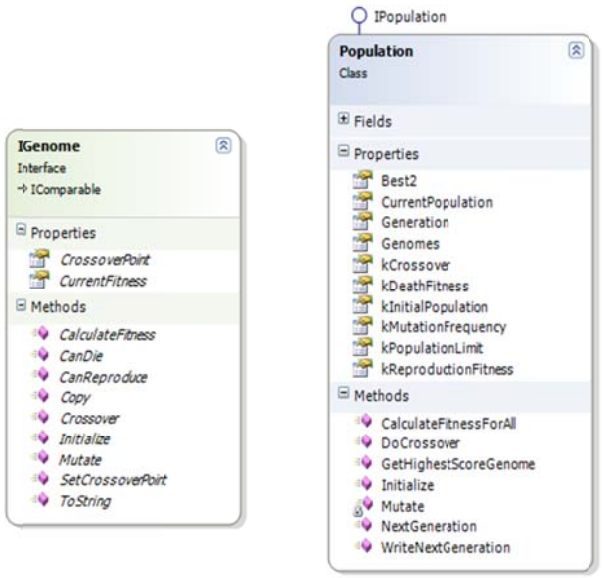

Fig. 2. Class diagram for GASolver

RCProjectSchedulingGenome implements a common contract IGenome, as shown in Figure 2 below, it does not worry about actual implementation of genome.

Hence with the use of interface IGenome, the solution is loosely coupled and there is no direct coupling between RCProjectSchedulingGenome and population class.

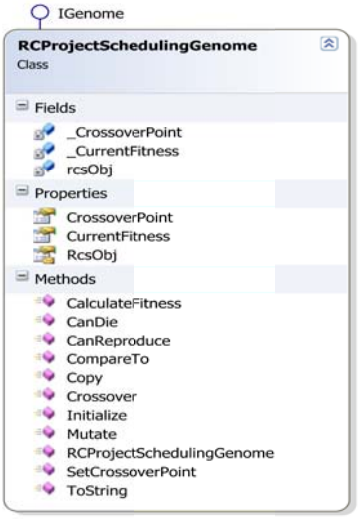

Fig. 3. RCProjectSchedulingGenome Class

GASolver.Core population class is also responsible for creating collection of genomes objects, it must know about the actual implementation of IGenome e.g RCProjectSchedulingGenome. But we cannot instantiate the actual genome object since it will tightly couple the population class to that genome implementation and the population class could not be used for other Genomes. We have used dependency injection object oriented programming principle to overcome this problem.

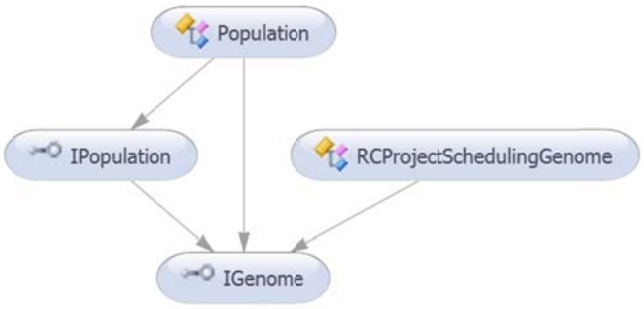

Fig. 4. GASolver's Dependencies Diagram

B. GASolver.RCPSP

GASolver.RCPSP is implementation of Resource constrained project scheduling problem. RCProjectSchedulingGenome class implements interface IGenome. This class is representation of genome and has methods for mutation, crossover and calculating fitness of genome. RCPSPDataConnection class is responsible making the connection to database and fetching different data from RCPSP database.

Above diagram is dependency graph of GASolver solution. GASolver.Core component instantiate RCProjectSchedulingGenome using unity dependency injection container.

VI. Test Results And Analysis

A. Test Case 1

Find a valid schedule that has lowest (PC), irrespective of the duration of the Project.

This is the first test case in which, we wish to optimize cost. We had two variables in the code, cost and duration. We changed the weight factor of cost to one and for duration it is made to zero i.e. the total focus is on project cost.

Project Cost is fully optimized while duration may vary high or low. The results are shown below in Table 13.

TABLE XIII. Optimized Project Cost

Generations	Project Duration	Project cost
1	65	2910910
3	65	2810910
61	31.67	2795520
66	51.67	2710040
73	47	2664760
94	29.35	2545550

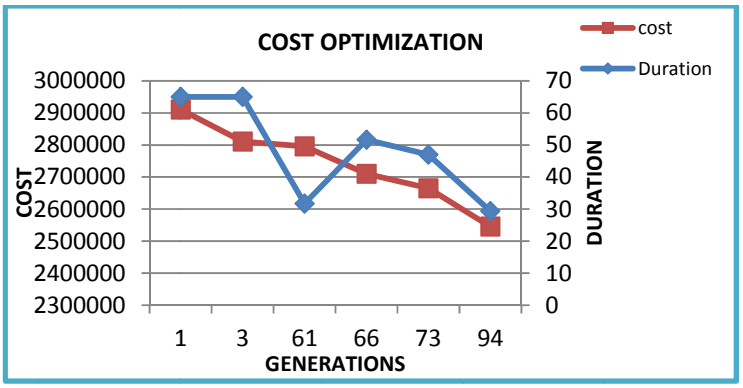

Fig. 5. RCPSP-Cost Optimization

The above Figure 5 shows the graph of cost optimization. We can analyze from the graph that Project Cost is constantly decreasing during higher generations while Project Duration is varying between higher and lower values as the weight factor for Project Cost is kept one.

B. Test Case 2

Find a valid schedule that has optimized duration, irrespective of the cost of the project.

This is the test case in which, we wish to optimize duration. We had two variables in the code, cost and duration. We changed the weight factor of duration to one and for cost it is made to zero i.e. the total focus is on Project Duration. Project Duration is fully optimized while Cost may vary high or low. The results are shown below in Table 14

TABLE XIV. RCPSP DURATION OPTIMIZATION

Generation	Duration	Cost
1	53.83	3153140
2	50.5	3109650
44	47.5	2758670
190	44.14	2589405
194	40.14	2565615
210	37.08	2702700
308	33	2667710

The Figure 6 below shows the graph of Duration Optimization. We can analyze from the graph that Project Duration is constantly decreasing during higher generations while Project Cost is varying between higher and lower values as the weight factor for Project duration is kept one and project cost is kept at zero.

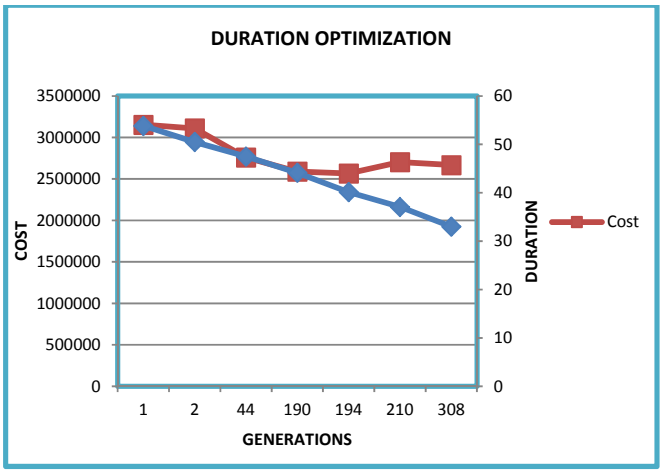

Fig. 6. RCPSP Duration Optimization

C. Test Case 3

Find the optimum valid schedule, satisfying a composite function including cost and duration.

This is the test case in which, we wish to optimize Cost and Duration both. We had two variables in the code, Cost and Duration.

We changed the weight factor of cost to 0.5 and for Duration it is made to 0.5 i.e. focus is on optimizing both Cost and Duration. The results are shown below in Table 15.

The Figure 7 shows the graph of Cost and Duration Optimization. We can analyze from the graph that Project Duration and well as Project Cost both are constantly decreasing during higher generations as the weight factor for Project duration is kept at 0.5 and Project Cost is also kept at 0.5.

TABLE XV. RCPSP- DURATION AND COST OPTIMIZATION

Generation	Project Duration	Project Cost
1	72	3038030
5	68	3013045
8	73.75	3012420
13	73	2946170
127	41.67	2614705
137	40	2507620
157	44.17	2347185

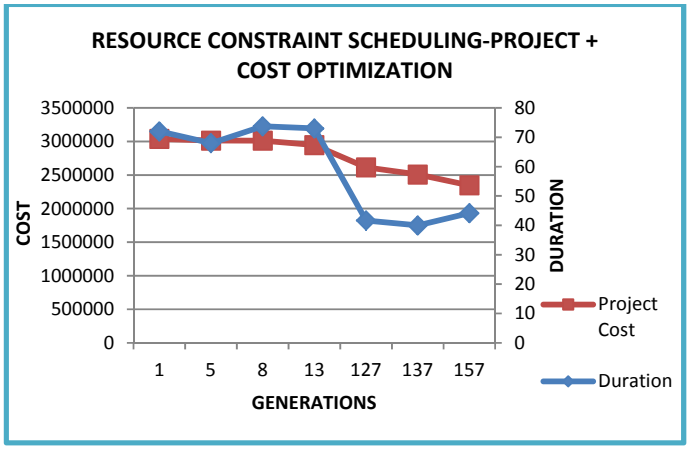

Fig. 7. RCPSP- Duration and Cost Optimization

VII. CONCLUSION AND FUTURE DIRECTIONS

Resource Constrained Project scheduling is an important problem as studied in literature survey. We have implemented this with Genetic Algorithm using C#.net. Most of the solutions that existed earlier for RCPSP were not extendable. We have implemented GASolver .core using which any specific problem domain genome can be constructed. The fitness function is only to be specified by the project manager for their own specific domain. The same GASolver .core can be extended to other important research areas like Time Cost trade off, Payment Scheduling problem etc. Once all these areas will be part of GASolver, it will be the complete solution to project scheduling problems.

List of abbreviation:

RCSP - Resource Constrained Scheduling Problem

PS - Payment Scheduling

DI - Dependency Injection

CT - Cost Trade off

GA - Genetic Algorithm

PC - Project Cost

REFERENCES

[1] Chambers LD (ed.) (1999) Practical handbook of genetic algorithms: complex coding

systems. CRC Press, Boca Raton

[2] David E. Goldberg " Genetic Algorithm, in search Optimisation and Macine Learning

[3] Davis L (1991) Handbook of genetic algorithms. Van Nostrand Reinhold, New York

[4] Dumond, J. and Mabert, V.A., "Evaluating Software project Scheduling and Due Date Assignment Procedures: An Experimental Analysis", Management Science, Vol. 34 No. 1, 1988, pp. 101-18

[5] J.Alcaraz, C.Moroto, " A Robust Genetic Algorithm for resource allocation in software project scheduling, Annals of operations Research 102(2001) 83-109.

[6] R.Mohring, A.Schulz, F.Stork, M.Uetz, " Solving Software project scheduling problems by minimum cut computations, Management science 49 (3) (2003) 330-335

[7] Moslem Shahsavar a,1, Seyed Taghi Akhavan Niaki b,*, Amir Abbas Najafi c,2 "An efficient Genetic Algorithm to maximize net present value of software project payments under inflation and bonus–penalty policy in resource investment problem", 2010 Elsevier

[8] Mohammad Amin Rigi, Shahriar Mohammadi K. N. Toosi Finding a Hybrid Genetic Algorithm-ConstraintSatisfaction Problem basedSolution for ResourceConstrained Software project Scheduling University of Technology, Industrial faculty, IT group Tehran, Iran, 2009 International Conference on Emerging Technologies.

[9] Stinson, J.P., Davis, E.W. and Khumawala, B.M., "Multiple Resource-constrained Scheduling Using Branch-and-Bound", AIIE Transactions, Vol. 10 No. 3, 1978, pp. 252

[10] Xinggang Luo 1,2, Dingwei Wang 2, Jiafu Tang 2, Yiliu Tu 3Resource-Constrained Software project Scheduling Problem , Proceedings of the 6th World Congress on Intelligent Control and Automation, June 21 23, 2006, Dalian, China.

[11] Yan Liu1,2,Sheng-Li zharo2, Xi-Ping Zhang2, Guang-Qiandu2, A GA-Based Approach for solving fuzzy siftware project scheduling Proceedings of the Sixth International Conference on Machine Learning and Cybernetics, Hong Kong, 19-22 August 2007.