Finding a Hybrid Genetic Algorithm-Constraint Satisfaction Problem Based Solution for Resource Constrained Project Scheduling

Mohammad Amin Rigi, Shahriar Mohammadi
K. N. Toosi University of Technology, Industrial faculty, IT group
Tehran, Iran
amin.rigi@gmail.com, mohammadi@kntu.ac.ir

***Abstract*- Project scheduling has attracted many researchers in recent years. It is about single-item or small batch production where scarce resources have to be met when scheduling dependent activities over time. Because of high complexity of the problem uninformed search strategies cannot solve the problem optimally. This paper proposes a new evolutionary approach to resource constrained project scheduling problem. Hybrid genetic algorithm (GA)-constraint satisfaction problem (CSP) has been applied to solve resource constrained project scheduling (RCPS). GA's task is to find the best schedule. Discussed 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.**

Keywords-project scheduling; genetic algorithm; constraint satisfaction problem; min-conflict heuristic;

I. INTRODUCTION

Scheduling decisions are generally subject to both precedence constraints and resource constraints. Many research works have dealt with a variety of situations in which one or both of these types of constraints are relaxed, or at least simplified. In a sense. the difficulties present in these simpler problems are superimposed in resource constrained project scheduling problem,[1].

The Resource-Constrained Project Scheduling Problem (RCPSP) can be stated as follows: A project consists of a set activities $A=\{a_1,a_2,\ldots,a_n\}$ where each activity has to be processed without interruption.

The dummy activities 1 and n represent the beginning and end of the project. The duration of an activity j is denoted by d_j where $d_1 = d_n = 0$. There are K renewable resource types. The availability of each resource type M in each time period is R_m units, m={1,..,M}. Each activity a_j requires $r_{j,m}$ units of resource m during each period of its duration where $r_{1,m} = r_{n,m} = 0$, m = 1, …,M. All parameters are assumed to be non-negative integer valued[9].

Each activity has some predecessors and some successors except start activity and finish activity. The notation P_i stands for activity a_i's predecessors and S_i shows the successors of activity a_i. The precedence can be shown by a graph. In fig. 1 we can see a simple activity on node (AON) precedence graph of a given project, the vector under name of each node shows the resources needed by that activity.

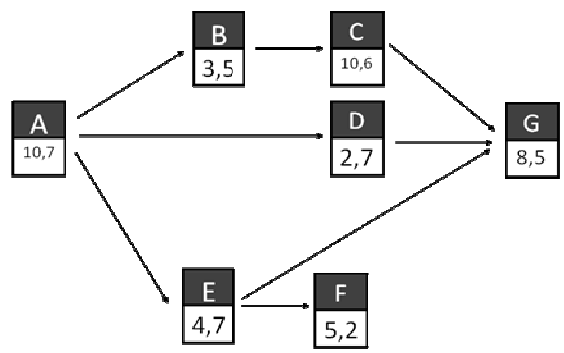

Figure 1. An AON project graph with 7 activities

The RCPS problem can be stated as follows. Minimize the project makespan by finding a schedule which satisfies these two constraints:
The precedence constraint should be satisfied.
The schedule should satisfy the resource constraint which is

$$\sum_{t=start}^{t=finish} r_{t,m} \leq r_m \qquad (1)$$

t stands for time. This inequality indicates that the resources using by activities in each time should not exceed the thresholds determined by resource vector.

Numerous attempts have been made to solve RCPS problem. Some surveys provided by Herroelen et al. [2], Brucker et al. [3] and Kolisch and Padman [4], as well as a book on Project Scheduling by Weglarz [5]. A research by Kolisch et al.[6] indicates that only small-sized problem instances with up to 60 activities can be solved exactly in a satisfactory manner. Therefore, heuristic solution procedures

978-1-4244-5632-1/09/$26.00 ©2009 IEEE

remain as the only feasible method of handling practical resource-constrained project scheduling problems. Recent overviews of heuristic procedures for the RCPSP can be found in Kolisch and Hartmann [6], Valls et al. [7], and Kolisch and Hartmann [8].

In this paper a system is proposed to find the best schedule for a given project with limited non-preemptable resources. Fig. 2 shows high level design of the problem and its solution.

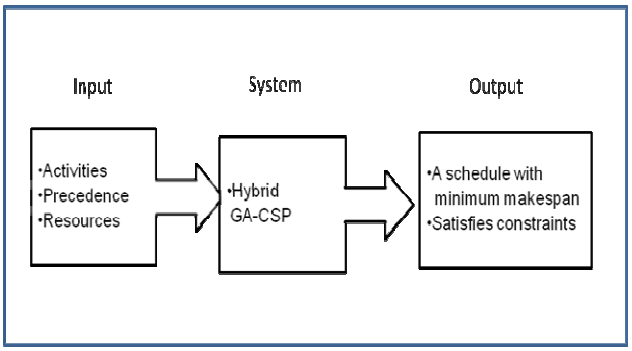

Figure 2. High level diagram of the RCPS problem and its solution

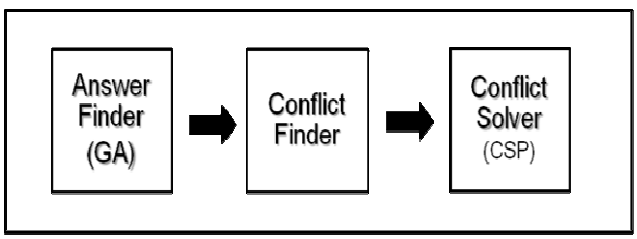

Figure 3. GA and CSP's role

Genetic algorithm is used to find the best schedule, but genetic algorithm operators cause some problems in the precedence and resource constraints. Then it is CSP's job is to solve these conflicts (like fig. 3).

The rest of paper is organized as follows. First the hybrid GA-CSP will be discussed, then CSP's role will be shown, after that we will see the complete solution to the problem and finally we have the results of our simulations.

II. HYBRID GENETIC ALGORITHM-CSP

Random search techniques (e.g., genetic algorithms and simulated annealing) try to overcome the shortcomings of the calculus-based and enumeration techniques (i.e., the problem of getting stuck at local optima and the lack of efficiency of the search). Although in the long run a random search technique can do no better than enumerative technique, it exploits information from the search space in order to guide the random selection process.

Consequently, it usually takes fewer objective function evaluations in order to get close to the optimal solution than complete enumeration. This is particularly useful when a computationally intensive simulation is used to generate the value of the objective function at each point. A random search technique also allows the search to move away from a local optimum because it does not require improvement in the objective function at every iteration.

Genetic algorithms, which were developed in the mid 1960's, are randomized search techniques that are based on the mechanics of genetics and natural selection and are designed for problems that require efficient and effective searches [10]. They attempt to "breed" good solutions using a paradigm based on natural evolution (survival of fittest).Genetic algorithm (GA) is a probabilistic optimization algorithm (or adaptation procedure) guided by the mechanics of natural evolution (according to Darwinian Theory of evolution).

Genetic algorithm characteristics in this specific problem (RCPS) are as follows:

A. *Chromosomes encoding*

Firstly, an initial population containing N_{pop} chromosomes is generated. Each gene in the binary chromosome is randomly assigned as either 1 or 0, with a probability of 0.5. The jth chromosome ($1 \le j \le N_{pop}$) is represented by $l_j^1 l_j^2 l_j^3 \cdots l_j^n$, where *l* denote the substrings that encode an activity in j^{th} schedule. And "n" is number of activities need to be scheduled, respectively. So each chromosome shows a way of project scheduling. For example l_j^1 will be encoded as the precedence of first activity (in the jth chromosome).

B. *Fitness Function*

Fitness function can be easily computed with the project makespan. Previously we find out that each chromosome is a schedule itself, therefore for determining the fitness of a given chromosome we only need to compute the project makespan.

C. *Initialization*

The next important step in genetic algorithm is to initialize the population. As noted by Davis and Steenstrup, the initialization process can be executed with either a randomly created population or a well adapted population [1]. For many combinatorial problems, it is relatively easy to create reasonably good initial population structure. For resource constrained project scheduling problem, because of the existence of both precedence and resource constraints, the search space is just part of all possible permutations. The random way to initializing evolution program will yield a considerable amount of infeasible schedules. So the initial chromosomes (individuals) are generated in this way, first for each chromosome a random schedule will be generated, in the next stage two CSPs will be applied to them. Here is a definition of a CSP with the graph coloring example. Formally speaking, a CSP is defined by a set of variables, XI, X2,…, Xn, (in graph coloring problem the nodes represent the variables and they can have colors as their

values) and a set of constraints, C1…Cp,(in graph coloring adjacent nodes cannot have same color). Each variable Xi has a nonempty domain Di of possible values (in graph coloring each node must choose a value between a set of predefined colors). Each constraint Ci involves some subset of the variables and specifies the allowable combinations of values for that subset. A state of the problem is defined by an assignment of values to some or all of the variables. An assignment that does not violate any constraints is called a consistent or legal assignment. A complete assignment is one in which every variable is mentioned, and a solution to a CSP is a complete assignment that satisfies all the constraints [11] (in graph coloring is a state that each node has a color and adjacent nodes have different colors). The first CSP is a complete state CSP that uses min conflict heuristic for solving precedence constraints. Fig. 4 shows the general algorithm for the min-conflict heuristic.

```
function MIN-CONFLICTS(csp, max_steps) return
                                    solution or failure
  inputs: csp, a constraint satisfaction problem
          max_steps, the number of steps allowed before
                     giving up
    current ← an initial complete assignment for csp
    for i = 1 to max_steps do
      if current is a solution for csp then return current
      var ← a randomly chosen, conflicted variable from
                VARIABLES[csp]
      value ← the value v for var that minimizes
                CONFLICTS(var,v,current,csp)
      set var = value in current
  return failure
```

Figure 4. CSP min-conflict algorithm[11]

For example in fig. 1 we have a precedence graph and let the activities order: A-B-E-F-G-D-C be a random schedule (chromosome) for first population which obviously has conflicts in precedence, then table 1 shows how the problem can be solved with the CSP min conflict heuristic. In table 1 we have seven variables (A,…,G), each variable has a domain (D_A,…,D_G), and each variable has a value in its domain.

The min conflict heuristic chooses a conflicted variable randomly and changes its value in order to solve the conflicts. For example variable G is chosen and its value will be changed from 5 to 7. After one step we can see the problem state in table 2. And finally after two more steps solution can be seen in table 3.

TABLE I. FIRST STATE IN SOLVING PRECEDENCE CONFLICT

Order	D_A	D_B	D_E	D_F	D_G	D_D	D_C
1	●						
2		●					
3			●				
4				●			
5					●		
6					↓	●	
7							●
conflicts	0	0	0	0	2	1	1

After solving the precedence conflicts now it is resource conflicts' turn to be solved. Since the number of activities is finite a simple iterative CSP with depth first search (DFS) is used to find an answer to the resources conflicts. It is obvious that the first CSP (precedence solver) will decrease the search space effectively.

TABLE II. SECOND STATE IN SOLVING PRECEDENCE CONFLICT

Order	D_A	D_B	D_E	D_F	D_G	D_D	D_C
1	●						
2		●					
3			●				
4				●			
5							↑
6						●	
7					●		●
conflicts	0	0	0	0	0	1	1

TABLE III. FINAL STATE IN SOLVING PRECEDENCE CONFLICT

Order	D_A	D_B	D_E	D_F	D_G	D_D	D_C
1	●						
2		●					
3			●				
4				●			
5						●	●
6							
7					●		
conflicts	0	0	0	0	0	0	0

D. *Reproduction*

For reproduction $N_{pop}/2$ with higher fitness are selected to make next population. We use weighted average method [12] for generating the next population.

E. *Crossover*

Since the chromosomes are binary coded, simple one point crossover has been used as crossover operation. The

basic elements of our implements consist of two parts, firstly perform a partially mapped crossover (PMX) operation and then perform a repair procedure to solve precedence conflict and resource conflict in the generated schedules.

F. Mutation

It is incorporated with the neighborhood technique to try to find an improved offspring. A lot of definitions may be considered for the neighborhood of a schedule.

Here, the set of schedules transformable from a schedule x by exchanging not more than x genes is regarded as the neighborhood of schedule x. The scheme of the crossover operator is depicted below [1].

Mutation Procedure

Step 1. Random pick up a substring with X genes

Step 2. Generate the neighboring schedules by pairwise interchange

Step 3. Solve the precedence conflict for the illegal neighbors

Step 4. Evaluate all neighbors and the best neighbor selected as the new schedule.

III. SIMULATION AND COMPARISON

Many researches had solved RCPS problem with genetic algorithm [1,9], but very rare of them had tried to solve the inconsistencies (that caused by GA's operator) with good heuristics. They tried simple uninformed searches for this phase of problem. In here the first CSP will cut off the search space effectively; so many irrelevant states in resources conflicts are ignored.

The proposed approach has been implemented with C# programming language exactly like the algorithm in fig. 5 for a RCPS problem with 10 renewable resources. The answer of hybrid GA-CSP method had been compared to the brute force method for problems with less than 20 activities; the answers were the same respectively. This shows the validity of our approach. As we noticed before this problem is an NP problem, therefore it was hard to test the validity of the approach for more than 20 activities with brute force method. The convergence of genetic algorithm shows that the genetic algorithm has been converged on an optimal solution. Genetic algorithm characteristics are shown in table 4.

The plots of genetic algorithm's best individuals' fitness (project makespan) with respect to generation for project with 50 and 200 activities is shown in fig. 6 and fig. 7.

TABLE IV. GA'S CHARASTERESTICS

Number of Individuals	100
Number of Genetic Algorithm Iterations	50
P_{Xover}	0.9
$P_{mutation}$	0.02

1. Create an initial population
2. Solve inconsistencies of precedence with CSP
3. Solve conflicts in resources with CSP
4. Compute fitness function
5. For [i=1 to niter]
    1. Select pair of individuals
    2. Crossover
        i. Apply partially mapped crossover operator
        ii. Solve resource and precedence conflicts
    3. Mutation
        i. Apply mutation
        ii. Solve resource and precedence conflicts
    4. Evaluate fitness of children
    5. Select best individuals for reproduction
6. Return the best solution

Figure 5. The algorithm

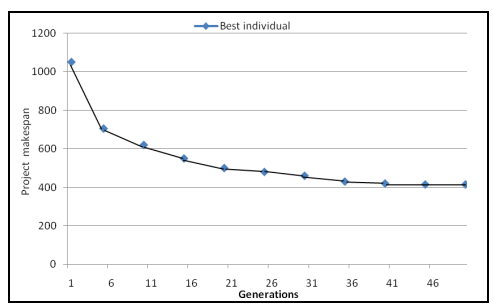

Figure 6. Project duration value of the best individuals in GA with respect to generation for 50 activities problem

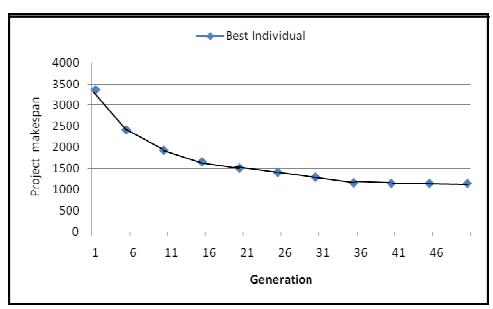

Figure 7. Project duration value of the best individuals in GA with respect to generation for 200 activities problem

IV. Discussion and Future work

This paper is focused to find a fast solution for RCPS problem, the process consists of a hybrid GA-CSP search algorithm. As shown in figs. 2 and 3, genetic algorithm will find the answers and CSP solves conflicts. The convergence of the GA shows the validity of reaching to an optimal solution with GA-CSP method. The flowchart of the hybrid GA-CSP algorithm is shown in fig. 8.

There is still one problem, as we said we use a CSP for cutting off the search space by deleting many irrelevant states with min conflict CSP. But in the second phase to overcome the resources conflicts we still use a kind of brute force solution which is an iterative depth first search (DFS) based CSP.

The future work might be on how to construct a full state CSP for whole of the problem. We are concentrating on how to import the whole structure of the problem (both precedence and resource) in CSP min conflict algorithm. Another approach for optimizing the structure might be using genetic programming (GP) because the search space is finite tree.

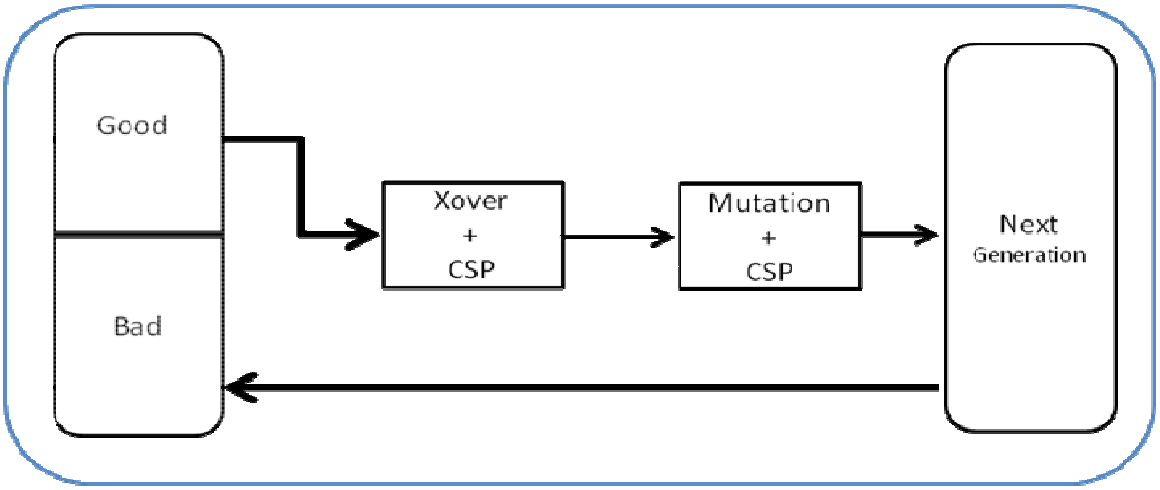

Figure 8. The flowchart of the GA-CSP system for RCPS

References

[1] Runwei Cheng and Mitsuo Gen, "Evolution program for resource constrained project scheduling problem", International Conference on Evolutionary Computation, Orlando, Florida, USA , pp. 736-741, June 1994.

[2] Herroelen, W., De Reyck, B., Demeulemeester, E.,. "Resource-constrained project scheduling: A survey of recent developments.", Journal of Computers and Operations Research, vol. 25 (4), pp. 279−302, 1998.

[3] Brucker, P., Drexl, A., Mo¨ hring, R., Neumann, K., Pesch, E., "Resource-constrained project scheduling: notation, classification, models, and methods.", European Journal of Operational Research, vol. 112, pp. 3−41, 1999.

[4] Kolisch, R., Padman, R, "An integrated survey of deterministic project scheduling", Omega vol. 29, No. 3, pp. 249-272, June 2001.

[5] Weglarz, J., Project Scheduling. Recent Models, Algorithms and Applications. Kluwer Academic Publishers. 1999, pp. 254-300.

[6] Kolisch, R., Sprecher, A., Drexl, A., "Characterization and generation of a general class of resource-constrained project scheduling problems", Management Science, vol. 41, pp. 1693−1703, 1995.

[7] Valls, V., Quintanilla, S., Ballestı ´ n, F., "Resource-constrained project scheduling: A critical activity reordering heuristic", European Journal of Operational Research, vol. 149, pp. 282−301, 2003.

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

[9] Vicente Valls a, Francisco Ballestn b, Sacramento Quintanilla, "A hybrid genetic algorithm for the resource-constrained project scheduling problem", European Journal of Operational Research, vol. 185, pp. 495−508, 2008.

[10] D.E. Goldberg, Genetic Algorithms: in Search, Optimization, & Machine Learning, Addison-Wesley Publishing Company, Inc., Reading, MA, 1989.

[11] Russel S., Norrvige P., Artificial Intelligence a Modern Approach, 2nd ed. ,printice hall, 2003.

[12] R. L. Haupt S. E. Haupt–Practical Genetic Algorithm, John Wiley & Sons, 2004.

[13] Julia Pet-Edwards, "A simulation and genetic algorithm approach to stochastic research constrained project scheduling" IEEE Southcon/96. Conference Record , Issue 25, pp. 333 – 338, Jun 1996.