Project scheduling using a competitive genetic algorithm

J. MAGALHÃES-MENDES
Department of Civil Engineering
School of Engineering – Polytechnic of Porto
Rua Dr. António Bernardino de Almeida, 431 – 4200-072 Porto
PORTUGAL

Abstract: - The resource constrained project scheduling problem (RCPSP) is a difficult problem in combinatorial optimization for which extensive investigation has been devoted to the development of efficient algorithms. During the last couple of years many heuristic procedures have been developed for this problem, but still these procedures often fail in finding near-optimal solutions. This paper proposes a genetic algorithm for the resource constrained project scheduling problem. The chromosome representation of the problem is based on random keys. The schedule is constructed using a heuristic priority rule in which the priorities and delay times of the activities are defined by the genetic algorithm. The approach was tested on a set of standard problems taken from the literature and compared with other approaches. The computational results validate the effectiveness of the proposed algorithm.

Key-Words: - Project management, scheduling, genetic algorithm, optimization, RCPSP.

1 Introduction

A project management problem consists typically of planning and scheduling decisions. Scheduling involves the allocation of the limited resources to projects to determine the start and completion times of the detailed activities.

The resource constrained project scheduling problem can be stated as follows. A project consists of $n+2$ activities where each activity has to be processed in order to complete the project. Let $J = \{0, 1, \ldots, n, n+1\}$ denote the set of activities to be scheduled and $K = \{1, ..., k\}$ the set of resources. The activities 0 and $n+1$ are dummy, have no duration and represent the initial and final activities. The activities are interrelated by two kinds of constraints:

1. The *precedence constraints*, which force each activity j to be scheduled after all predecessor activities, P_j, are completed.
2. Performing the activities requires resources with *limited capacities*.

While being processed, activity j requires $r_{j,k}$ units of resource type $k \in K$ during every time instant of its non-preemptable duration d_j. Resource type k has a limited capacity of R_k at any point in time. The parameters d_j, $r_{j,k}$ and R_k are assumed to be non-negative and deterministic. For the project start and end activities we have $d_0 = d_{n+1} = 0$ and $r_{0,k} = r_{n+1,k} = 0$ for all $k \in K$.

The problem consists in finding a schedule of the activities, taking into account the resources and the precedence constraints, that minimizes the *makespan* (C_{max}).

Let F_j represent the finish time of activity j. A schedule can be represented by a vector of finish times ($F_1, \ldots, F_m, \ldots, F_{n+1}$). The *makespan* of the solution is given by the maximum of all predecessors activities of activity $n+1$, i.e. $F_{n+1} = Max_{l \in P_{n+1}} \{F_l\}$.

The RCPSP problem belongs to the class of NP-hard optimization problems, therefore justifying the indispensable use of heuristic solution procedures when solving large problem instances. The most competitive heuristics approaches seem to be those of Debels et al. [3], Debels and Vanhoucke [4], Mendes et al. [6], Fleszar and Hindi [8], Palpant et al. [9], Kochetov and Stolyar [11], Valls et al. [12], Valls et al. [13], Ranjbar [14] and Mendes and Gonçalves [15]. A survey provided by Kolisch [16] presents more than eighty models and algorithms for complex scheduling problems and discusses the RCPSP.

2 Application of genetic algorithm

Genetic algorithms (Gas) are search algorithms based on the mechanics of natural selection and natural genetics. They combine 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 [1].

The general schema of GAs may be illustrated as follows (Fig. 1).

```
procedure GENETIC-ALGORITHM

Generate initial population P0;
 Evaluate population P0;
 Initialize generation counter g ←0;

 While stopping criteria not satisfied repeat
    Select some elements from Pg to copy into Pg+1;
    Crossover some elements of Pg and put into Pg+1;
    Mutate some elements of Pg and put into Pg+1;
    Evaluate some elements of Pg and put into Pg+1;
    Increment generation counter: g ← g+1;
  End while

End GENETIC-ALGORITHM;
```

Fig.1 - Pseudo-code of a genetic algorithm.

First of all, an initial population of potential solutions (individuals) is generated randomly. A selection procedure based on a fitness function enables to choose the individuals candidate for reproduction. The reproduction consists in recombining two individuals by the crossover operator, possibly followed by a mutation of the offspring. Therefore, from the initial population a new generation is obtained. From this new generation, a second new generation is produced by the same process and so on. The stop criterion is normally based on the number of generations.

2.1 Decoding

The genetic algorithm uses a random key alphabet which is comprised of real random numbers between 0 and 1.

A chromosome represents a solution to the problem and is encoded as a vector of random keys (random numbers). Each solution chromosome is made of 2*n* genes where *n* is the number of activities:

$$Chromosome = (gene_1, .., gene_n, gene_{n+1}, ..., gene_{2n})$$

2.1.1 Decoding the priorities of activities

The priority decoding expression used the following expression

$$\boldsymbol{PRIORITY_j} = \frac{\boldsymbol{LLP_j}}{\boldsymbol{LCP}} \times \left[\frac{1+\boldsymbol{gene_j}}{2}\right] \qquad \boldsymbol{j} = 1,...,\boldsymbol{n}$$

where $\boldsymbol{LLP_j}$ is the longest length path from the beginning of the activity *j* to the end of the project and $\boldsymbol{LCP}$ is length along the critical path of the project.

2.1.2 Decoding the delay times

The genes between *n*+1 and 2*n* are used to determine the delay times used when scheduling an activity. The delay time used by each scheduling iteration *g*, $Delay_g$, is given by the following expression:

$$Delay_g = gene_{n+g} \times 1.5 \times MaxDur$$

where MaxDur is the maximum duration of all activities. The factor 1.5 was obtained after some experimental tuning.

2.1 Evolutionary strategy

To breed good solutions, the random key vector population is operated upon by a genetic algorithm.

There are many variations of genetic algorithms obtained by altering the reproduction, crossover, and mutation operators.

Reproduction is a process in which individual (chromosome) is copied according to their fitness values (*makespan*).

Reproduction is accomplished by first copying some of the best individuals from one generation to the next, in what is called an elitist strategy.

In this paper the roulette wheel selection method is used to select the individuals for reproduction. The characteristic of the roulette wheel selection is stochastic sampling. The survival probability of each individual is determined by its fitness. A roulette wheel model is established to represent the survival probabilities for all the individuals in the population. Then the roulette wheel is rotated for several times [1].

After reproduction, crossover may proceed in two steps. First, members of the newly reproduced chromosomes in the mating pool are mated at random. Second, each pair of chromosomes undergoes crossover as follows: an integer position *k* along the chromosome is selected uniformly at random between 1 and the chromosome length *l*. Two new chromosomes are created swapping all the genes between *k*+1 and *l* [1], see Fig. 2.

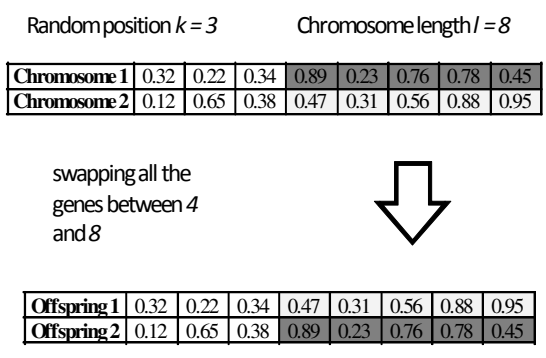

Fig.2 – Crossover operator example.

The mutation operator preserves diversification in the search. This operator is applied to each offspring in the population with a predetermined probability. We assume that the probability of the mutation in this paper is 0.001. With 60 genes positions we should expect 60 x 0.001 = 0.06 genes to undergo mutation for this probability value.

3 Constructive heuristic

The constructive heuristic used to construct active schedules is based on a scheduling generation scheme that does time incrementing.

This heuristic makes use of the priorities and the delay times defined by the genetic algorithm and constructs active schedules. This heuristic is described by Gonçalves et al. [5] and Mendes et al. [6].

4 Local Search

Local search algorithms move from solution to solution in the space of candidate solutions (the search space) until a solution optimal or a stopping criterion is found. In this paper was applying backward and forward improvement based on Klein [10].

Initially is constructed a schedule by planning in a forward direction starting from the project's beginning. After is applying backward and forward improvement based on Klein [10].

5 Computational results

The experiments were performed on an Intel Core 2 Duo CPU T7250 @2.00 GHz. The algorithm was coded in Visual Basic 6.0. The GA-RKV (Genetic Algorithm - Random Key Variant) was tested on the instance sets J30 and J60 available in PSPLIB.

5.1 Genetic algorithm configuration

Though there is no straightforward way to configure the parameters of a genetic algorithm, we obtained good results with values: **population size** of 5 × number of activities in the problem; **mutation probability** of 0.001; **top** (best) 1% from the previous population chromosomes are copied to the next generation; **stopping criterion** of 250 generations.

5.2 Experimental results

Table 1, column 3, summarizes the average deviation percentage from the optimal *makespan* (D_{OPT}), for the instance set J30.

Table 2, columns 3, summarizes the average deviation percentage from the well-known critical path-based lower bound (D_{LB}) for the instance set J60. This lower bound values are reported by Stinson et al. [7].

The maximum computational time dispended is 120 seconds for each instance of J60.

Algorithm	*Reference*	*J30* D_{OPT}
MAPS	Mendes and Gonçalves [15]	0.00
GA-TS path relinking	Kochetov and Stolyar [11]	0.00
Decomp. & local opt.	Palpant et al. [9]	0.00
F&F(5)	Ranjbar [14]	0.00
GA - RKV	***This paper***	**0.01**
GAPS	Mendes et al. [6]	0.01
Scatter Search - FBI	Debels et al. [3]	0.01
VNS-activity list	Fleszar and Hindi [8]	0.01
GA - DBH	Debels and Vanhoucke [4]	0.02
GA – hybrid, FBI	Valls et al. [12]	0.02
GA - FBI	Valls et al. [13]	0.02

Table 1: Top-ten computational results for J30 instances.

Algorithm	*Reference*	*J60* D_{LB}
F&F(5)	Ranjbar [14]	10.56
MAPS	Mendes and Gonçalves [15]	10.64
GAPS	Mendes et al. [6]	10.67
GA - DBH	Debels and Vanhoucke [4]	10.68
Scatter Search - FBI	Debels et al. [3]	10.71
GA – hybrid, FBI	Valls et al. [12]	10.73
GA, TS – path relinking	Kochetov and Stolyar [11]	10.74
GA - FBI	Valls et al. [13]	10.74
Decomp. & local opt.	Palpant et al. [9]	10.81
GA - RKV	***This paper***	**10.88**
VNS-activity list	Fleszar and Hindi [8]	10.94

Table 2: Top-ten computational results for J60 instances.

6 Conclusion

This paper presents a new genetic algorithm for the resource constrained project scheduling problem. The chromosome representation of the problem is based on random keys. Reproduction, crossover and mutation are applied to successive chromosome populations to create new chromosome populations. These operators are simplicity itself, involving random number generation, chromosome copying and partial chromosome exchanging.

The schedules are constructed using a priority rule in which the priorities are defined by the genetic algorithm. Schedules are constructed using a constructive heuristic.

The approach was tested on a set of 960 standard instances taken from the literature and compared with the best state-of-the-art approaches. The algorithm produced good results when compared with other approaches therefore validating the effectiveness of the proposed algorithm.

Acknowledgements

This work has been partially supported by the Polytechnic of Porto, IPP/PADInv2007.

References:

[1] D. E. Goldberg. *Genetic Algorithms in Search, Optimization & Machine Learning*, Addison-Wesley, 1989.

[2] D. Beasley, D.R. Bull and R.R. Martin, *An Overview of Genetic Algorithms: Part 1, Fundamentals*, University Computing, Department of Computing Mathematics, University of Cardiff, UK, Vol. 15(2), 1993, pp. 58-69.

[3] D. Debels, B. De Reyck, R.Leus and M.Vanhoucke, A Hybrid Scatter Search/Electromagnetism Meta-Heuristic for Project Sheduling, *European Journal of Operational Research,* Vol. 169, 2006, pp. 638-653.

[4] D. Debels and M. Vanhoucke. *A Decomposition-Based Heuristic for the Resource-Constrained Project Scheduling Problem*. Working Paper 2005/293, Faculty of Economics and Business Administration, University of Ghent, Ghent, Belgium, 2005.

[5] J.F. Gonçalves, J.M. Mendes, and M.C.G. Resende. A hybrid genetic algorithm for the job shop scheduling problem. *European Journal of Operational Research*, Vol. 167, 2005, pp. 77-95.

[6] J.J.M. Mendes, J.F. Gonçalves 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.

[7] J.P. Stinson, E.W. Davis and B.M. Khumawala, Multiple Resource-Constrained Scheduling Using Branch and Bound, *AIIE Transactions*, Vol. 10, 1978, pp. 252-259.

[8] K. Fleszar and K.S. Hindi, Solving the resource-constrained project scheduling problem by a variable neighbourhood search, *European Journal of Operational Research*, Vol. 155, 2004, pp. 402-413.

[9] M. Palpant, C. Artigues and P. Michelon, LSSPER: Solving the resource–constrained project scheduling problem with large neighbourhood search, *Annals of Operations Research*, Vol.131, 2004, pp. 237-257.

[10] R. Klein, Bidirectional planning : improving priority rule-based heuristics for scheduling resource-constrained projects, *European Journal of Operational Research,* Vol. 127, 2000, pp. 619-638.

[11] Y. Kochetov and A. Stolyar. Evolutionary local search with variable neighborhood for the resource constrained project scheduling problem. In *Proceedings of the 3rd International Workshop of Computer Science and Information Technologies*, Russia, 2003.

[12] V. Valls, F. Ballestin and M.S. Quintanilla. *A hybrid genetic algorithm for the RCPSP*. Technical report, Department of Statistics and Operations Research, University of Valencia, 2003.

[13] V. Valls, F. Ballestin and M.S. Quintanilla, Justification and RCPSP: A technique that pays, *European Journal of Operational Research*, Vol.165, 2005, pp. 375-386.

[14] M. Ranjbar, Solving the resource-constrained project scheduling problem using filter-and-fan approach, *Applied Mathematics and Computation*, Vol. 201, 2008, pp. 313–318.

[15] J.J.M. Mendes and J.F. Gonçalves, A Memetic Algorithm-Based Heuristics for the Resource Constrained Project Scheduling Problem, Proceedings of *II International Conference on Computational Methods for Coupled Problems in Science and Engineering*, Spain, 2007, pp. 644-648.

[16] R. Kolisch and S. Hartmann, Experimental investigation of heuristics for resource-constrained project scheduling: an update, *European Journal of Operational Research*, Vol.174 (1), 2006, pp. 23-37.