HEURISTIC IDEAS OF USING GENETIC ALGORITHM FOR SOLVING LOT-SIZE PRODUCTION SCHEDULING PROBLEMS

By Viktor GORELIK[1], Wim DE BRUYN[2] and Dmitriy BORODIN[2]

[1] Dorodnicyn Computing Centre of Russian Academy of Sciences, Moscow

[2] University College of Ghent, Belgium

dk.borodin@gmail.com

Genetic algorithms need special representation of solutions to be efficient in this or that problem case. This paper analyzes some particular cases of the lot-size production scheduling problems where solutions may be represented as binary variables, and, moreover, on each solution string one and only one component is 'true'. This gives two heuristic ideas to apply the genetic algorithm.

Introduction

The lot-size production scheduling problem addresses the problem of cyclic scheduling and lot sizing of multi-products in one facility as to minimizing the production duration, setup costs, fulfil customer time demands, etc. This problem is practically important and has been investigated since 1950s. A number of good reviews on the problem was provided by international researchers [eg Rogers, 1958; Elmaghraby, 1978; Lopez and Kingsman, 1991; Yao, 1999].

In this paper we assume that that the scheduling problem is formulated in a very generic way: Minizime an Objective function, subject to production capacity, resource and other constraints, formulated as equations and inequalities.

We also assume that decision variables are binary, ie $x_{i,j,k} \in \{0,1\}$.

There are many techniques developed for solving such problems, including analytical algorithms (eg Branch-and-Bound [Theo C. Ruys, 2001]) and heuristic/Metaheuristic algorithms (eg local search, tabu search, ant colony optimization, evolutionary algorithms etc). The time cost of analytical algorithms growth significantly with the size of the problem and it becomes practically impossible and unreasonable to use them for real-life problems. To avoid this, more and more heuristic ideas are being developed trying to reduce the computation time.

Here we explain the heuristic ideas for using genetic algorithm (GA) to solve some of the above mentioned problems.

Genetic Algorithm

A population of abstract representations (called chromosomes or the genotype of the genome) of candidate solutions (called individuals, creatures, or phenotypes) to an optimization problem evolves toward better solutions. Traditionally, solutions are represented in binary as strings of 0s and 1s, but other encodings are also possible. The evolution usually starts from a population of randomly generated individuals and happens in generations. In each generation, the fitness of every individual in the population is evaluated, multiple individuals are stochastically selected from the current population (based on their fitness), and modified (recombined and possibly randomly mutated) to form a new population. The new population is then used in the next iteration of the algorithm. Commonly, the algorithm terminates when either a maximum number of generations has been produced, or a satisfactory fitness level has been reached for the population. If the algorithm has terminated due to a maximum number of generations, a satisfactory solution may or may not have been reached.

Heuristic Ideas

Two ideas are offered within the mentioned above problem.

<u>Idea 1: Use vectors of binary solutions as chromosomes without encoding</u>

We have a matrix of binary variables like $x_{i,j,k} \in \{0,1\}$, it is possible to represent it as a set of vectors. Such a vector looks as following:

$$x_{i,j} = (0,0,0,1,0...,0,0,0,1,0,0,0,0,0)$$

Each vector is a chromosome, and set of chromosomes is a solution in the terms of GA, and, in this case, respectively a solution of the mentioned above problem.

Thus, it is possible to save computation time by excluding from GA encoding and decoding of solutions.

In order to handle the constraints [P.-T. Chang et al., 2006], it is possible to discard strings on the generation step of GA: once the string is generated, it is checked whether the string is feasible or not; if not, the string is discarded, and new one is generated.

The reproduction process produces offspring or children to the next generation. It may be executed by using the classic roulette wheel technique with the respective probabilities of the strings to be reproduced.

The classic one-point crossover may be illustrated by the following example [P.-T. Chang et al., 2006]: with the respective probability for crossover of two strings randomly selected in which the genes of the products in the two strings after a randomly selected product are swapped, is used.

String 1	Product 1_1	Product 2_1	Product 3_1	Product 4_1	Product 5_1	Product 6_1
String 2	Product 1_2	Product 2_2	Product 3_2	Product 4_2	Product 5_2	Product 6_2

String 1′	Product 1_1	Product 2_1	Product 3_1	Product 4_2	Product 5_2	Product 6_2
String 2′	Product 1_2	Product 2_2	Product 3_2	Product 4_1	Product 5_1	Product 6_1

For mutation with the respective mutation probability, the procedure of randomly selecting a product, from which randomly selecting a gene, changing the value to one and reassigning zero to all the other genes of this product, is used.

Product 1	Product 2	Product 3	Product 4	Product 5	Product 6

Product 3																							
0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Product 3																							
0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0

Moreover, after crossover and mutation operators infeasible strings may also be created. This problem can be resolved by replacing the infeasible one(s) with the highest total-cost one(s) in the last population instead of the lowest total-cost one(s) as to maintaining the diversity of chromosomes.

Idea 2: Use vectors of binary solutions as chromosomes without encoding

The decision variables of the problem described above may be represented as a matrix, let it be a three-dimensional matrix $x_{i,j,k} \in \{0,1\}$. In many cases the problem is in finding the values of decision variables equal to 1, in other words, to determine places of 1s in matrix sub vectors. So, we are interested only in indexes of our variables where the values are equal to 1.

Instead of searching the space of 0s and 1s, we search the vectors of integer numbers, each giving the respective place (index) of 1 for the original formulation. For this, we can use standard GA and define boundaries of our index variables, and for the evaluation of generated solutions on each stage we assume the objective function and the constraints to be dependent on the index variable(s).

Tools for Computational Experiments

There are a number of tools which help perform computation experiments to check the algorithm and/or solve the real-life problem.

For the case of this paper it is reasonable to use either Mathcad (www.ptc.com/products/**mathcad**/) [Dyakonov, 2007] for small and average sized problems or Matlab (www.mathworks.com/products/**matlab**/) [Dyakonov, 2008] for the problems of bigger (ie industrial) size.

Moreover, Matlab has a special GA toolbox which can help solve implement the described heuristic ideas.

Authors use Mathcad to test their heuristic ideas; it gives the solution within the reasonable time period for test instances of the problem under study.

Special Thanks

Authors want to thank Professor Vladimir DYAKONOV for the help with implementation of GA in Mathcad and for the fruitful cooperation.

References

Rogers, J. (1958) A computational approach to the economic lot scheduling problem. Management Science 4, 264–291.

Elmaghraby, S.E. (1978) The economic lot scheduling problem (ELSP): Review and extensions. Management Science 24, 587–598.

Lopez, M.A., Kingsman, B.G. (1991). The economic lot scheduling problem: Theory and practice. International Journal of Production Economics 23, 147–164.

Yao, M.-J. (1999) The economic lot scheduling problem with extension to multiple resource constraints. Ph.D. Dissertation, North Carolina State University, Raleigh, NC, USA.

Theo C. Ruys (2001). Optimal Scheduling using Branch and Bound with SPIN 4.0, 16 p.

P.-T. Chang et al. (2006) A genetic algorithm for solving a fuzzy economic lot-size scheduling problem / Int. J. Production Economics vol. 102, pp. 265–288.
Dyakonov, V. (2007) Mathcad 11/12/13 in Mathematics. Guide. Published in Moscow by Goryachaya Liniya. Telecom (*in Russian*).
Dyakonov, V. (2008) MATLAB R2006/2007/2008+Simulink 5/6/7. Implementation Guide. Second edition. Published in Moscow by SOLON-Press, 800 pages (*in Russian*).