Study of different setup costs in SingleGA to solve a one-dimensional cutting stock problem

Julliany Sales Brandão[1], Alessandra Martins Coelho[2], Felipe do Carmo[3], João Flávio Vasconcelos[3],
[1]Centro Federal de Educação Tecnológica do Rio de Janeiro, [2]Instituto Federal de Educação Ciência e Tecnologia do Sudeste de Minas Gerais, [3]Universidade do Estado do Rio de Janeiro

Abstract - This paper presents the application of new costs for one recent approach, called SingleGA, in solving One-Dimensional cutting stock problem. The cutting problem basically consists in finding the best way to obtain parts of distinct sizes (items) from the cutting of larger parts (objects) with the purpose of minimizing a specific cost or maximizing the profit. The obtained results of SingleGA are compared to the following methods: SHP, Kombi234, ANLCP300 and Symbio, found in literature, verifying its capacity to find feasible and competitive solutions. The computational results show that variations of SingleGA posses good results, improving as setup cost increases.

Keywords: One-dimensional cutting problem; Genetic algorithm; Setup.

INTRODUCTION

The cutting stock problem is a generic term for a class of combinatorial problems which consists in finding the best items arrangement (cutting pattern) of different sizes (items) from cutting larger pieces (objects), aiming a specific purpose [1]. There is, in this case, the importance of geometry, since the items and objects shapes and dimensions determine the possible cutting patterns. This is an important issue in the Operations Research area, being widely studied by the scientific community.

The studies of cutting problems have been stimulated by the companies need to improve their processes due to competition among them, as well as waste and costs reduction, and efficiency in delivery. This study ~~it~~ became critical and very relevant in order to production planning several industries segments such as glass, pulp & paper, textile, chemical, among others.

Kantorovick [2] was a pioneer in the field of cutting stock problems. However, the area breakthrough was the work Gilmore e Gomory ([3], [4]) studying the cutting stock problem through the column generation process.

Haessler (1975) was the first to address the non-linear one-dimensional cutting stock problem this way. The objectives are considered inversely related or partially conflicting, because, as the setup is reduced, the number of processed objects is increased.

It is necessary the cutting patterns and frequency standards to solve a cutting stock problem, i.e. the number of times that these standards will be implemented. Although the overall goal is to minimize losses, several modeling is the problem, namely, profit maximization, the reduction of objects used, the production time and / or a combination thereof.

The cost of preparing the machine is a relevant factor in some cutting processes. Thus, it is interesting to evaluate the effect of minimizing the number of processed objects (input minimization) and the minimizing of the number of cutting patterns (setup), goals which are partially conflicting, for a more general assessment of the cost. The problem discussed here belongs to the class NP-Complete. In this case the use of heuristics or meta-heuristics is justified, generating, for these, good solutions in a short period of time.

This paper will analyze the behavior of SingleGA with varying setup cost for comparison and method efficacy verification.

The paper is organized as it follows: Section 2 formally deals with the cutting problem in order to reduce the number of processed objects and the setup. The basic concepts of genetic algorithms are presented in Section 3. Section 4 presents the computational implementation and sections 5 and show, respectively, the computational results and conclusions.

ONE-DIMENSIONAL CUTTING STOCK PROBLEM

A formal mathematical model that represents these goals is described below (1):

DOI: 10.5176_2010-2283_2.1.118

Minimize $c_1 \sum_{j=1}^{n} x_j + \alpha \sum_{j=1}^{n} \delta(x_j)$

(1)

Subject to : $\sum_{j=1}^{n} a_{ij} x_j \geq d_i \quad i = 1, \ldots, m$

$x_j \in N$

$j = 1, \ldots, n$

where:

$$\delta(x_j) = \begin{cases} 1, \text{ se } & x_j > 0 \\ 0, \text{ se } & x_j = 0 \end{cases}$$

n = number of possible cutting patterns;

m = number of different items;

c_1 = cost of each coil;

c_2 = cost of replacement of standard cutting;

x_j = number of coils processed with the standard cut *j;*

a_{ij} = number of items of type *i* in pattern *j;*

d_i = number of items i demanded;

GENETIC ALGORITHMS

Genetic algorithms are searched and the optimization of solutions algorithms, are based on genetics and evolutionary mechanisms of living beings, such as natural selection and the survival of the fittest, introduced by Charles Darwin in his classic "The Origin of Species" (1859) whose first published known work dates from the late 50's and early 60's.

The Genetic Algorithms, rigorously introduced by John Holland [6], work with a population of individuals, in which each one represents a possible solution to a given problem. Each individual has fitness, that is, a value that quantifies the individual's adaptability to the environment (treated problem). Individuals with higher fitness have higher chances of being selected for reproduction through the intersection, and thereby spread over their characteristics for future generations, allowing the most promising areas of research to be explored, taking the genetic algorithm, in most cases, to the convergence to the problem optimal solution.

COMPUTING CONSTRUCTION SINGLEGA

The SingleGA developed by [7], was based on Symbio [8]. However, the use SingleGA is just a genetic algorithm to solve the same goal, while the Symbio makes use of concepts of symbiosis [10], specifically, a mutualistic relationship between two genetic algorithms [11] that evolve beneficial way.

The SingleGA is a genetic algorithm composed of two kinds of population: the solutions and the patterns, for instance, have the ability to build cutting patterns, regardless of the solution. The population patterns, generated randomly, are static, in this case, not suffering from any kind of evolution.

The solution population gene was represented by two elements. The first refers to the amount of time (frequency) that the standard indicates by the second element processed. The second element only served as a reference to the pattern population.

The pattern population gene was represented by a real number that indicates the item length range from requirements list the clients.

The maximum amount of genes from the solution individual (maximum size of the genome) used was equal to the number of setups (number of different items). However, if the largest item to be cut presents a length less than or equal to 50% of the coil length, the following procedure would be adopted, experimentally determined and presented in Figure 1, which is an adaptation of [8].

The pattern genome size adopted was equal to the greatest integer, less than or equal to the result of the division between the size of the default coil and the smaller item of the pattern.

The items were added to the pattern from left to right, if and only if, the pattern had sufficient free space to accommodate it (approach based on the work of [8].

The solutions population (individuals formed by genes from the solutions population) was formed by 600 individuals and the patterns population (individuals formed by genes from the patterns population) by 400, in both cases, all individuals were generated randomly, with no direction. Populations sizes were defined experimentally, after simulations with different values.

```
if (number of items> 30) then
    SolutionGenes = 16
else
    if (Number of items> 15) then
        SolutionGenes = 12
            else
        SolutionGenes = 8
    end if;
end if
```

Figure 1 - Number of Genes of the Solution Individual

The structure selection adopted was the elitism with steady state. It was chosen because of the fact that this feature increases the performance of genetic algorithm, since it ensures that the best solution found so far is maintained in future generations. The steady state used is available in the package GALib [11] form the class of genetic algorithms GASteadyStateGA. According to the GALib's tutorial (Wall, 1996), the method generates, in each generation, a temporary population of individuals by cloning, and they are inserted in the population of the current generation, removing the worst individuals. The replacement rate, found experimentally, was 25%, which means that 75% of the solution population best individuals will be selected and remain in the population.

The fitness calculation (*Fs*) of the individual solution in order to meet the objective of minimizing both the number of processed objects and *setup,* based and described on [8], was performed as follows:

$$F_s = c_1 \sum x_j + \alpha \sum \delta(x_j) + \sum \tau(x_j) + \rho \qquad (2)$$

where:

τ: is the relative loss and can be calculated by $\tau(x_j) = \frac{t_j}{w \sum_{j=1}^{n} x_j}$, being t_j the waste of standard j.

ρ: are penalties if the solution is not feasible. Is proportional to the sum of the infeasibilities, i.e., the value remaining to meet the demand of the item, multiplied by 1000 (value experimentally chosen).

Values c_1 and α, respectively, are the costs of processed objects and setup, treated explicitly in the objective function, not requiring any other modifications to achieve different objectives, i.e., the cost is already involved directly in the minimization of the objective function.

To calculate the objective function an interpretation of the patterns was performed, i.e., it was determined which were the active genes of the pattern. The active genes are those that fit in the master-piece without blowing its size.

The recombination operator (crossover) used was the uniform, which consists on randomly choose a value between 0 and 1. If the number drawn was less than or equal to 0.7 the recombination was carried out with the individual who owned the greatest fitness, otherwise, it was carried out with the worst individual. The recombination rate, found experimentally, was 30%.

The mutation operator adopted consisted on randomly choosing a position in the genome of the gene to be mutated and then randomly determining which of its elements (frequency or default_index) would suffer mutation. If the chosen element of the gene were the frequency, a value between the minimum and maximum limits were drawn, otherwise, one individual was randomly chosen in the pattern population and a pointer would be created. For the standard index, component of the solution population's gene, the limits were between zero and the maximum number of setups and for the frequency, the limits were between zero and the value of the highest demand [8].The mutation rate was given by the ratio between 1 and different number of items (*1/m).*

Regarding the stopping criteria, three were used:

- maximum number of generations: 1000;
- maximum execution time: 500s;
- convergence: 500 generations, ie, if the algorithm does not improve the solution by 500 generations, it stops.

At this point is presented the pseudo code of the proposed method called SingleGA:

```
Procedure SingleGA
1 Generate the individuals of the patterns population randomly;
2 Generate the individuals of the solutions population randomly;
3 Calculate the objective function of the solutions individuals;
4 Select the solutions individuals parents;
5 Use recombination and mutation operators to generate new solutions;
6 Evolve the population;
7       if some stopping criterion is satisfied STOP the execution of the algorithm;
8       else return to step 3.
End-Procedure SingleGA
```

The similarity with the Symbio consists on the objective function, in the way patterns are added and some limits cited. However, besides using only one genetic algorithm, the other procedures, genetic operators, rates and values adopted are different, as presented in pseudo-code and stopping criteria described above.

COMPUTATIONAL RESULTS

The problems used for the computational tests with the SingleGA were generated randomly by CUTGEN1 [12]. We generated 18 classes characterized by different values of input parameters, each class containing 100 problems, totaling 1800 tests for evaluating the quality of the proposed method.

The parameters and the seed used to generate the 1800 problems in CUTGEN1 were the same used by Foerster and Wascher [13], Salles Neto and Moretti [1] and Golfeto et al. [8].

The results of SingleGA will be presented with two new values for the cost of setup: $\alpha = 1$ which was termed SingleGA01, $\alpha = 5,$ called SingleGA05 and cost $\alpha = 10,$ presented in [7] which was called SingleGA10. In all cases, the cost of the number of objects processed was of a unit $(c_1 = 1).$ The value of α is intended to penalize the number of different patterns.

For the purpose of assessing the quality of SingleGA, in solving the cutting stock problem mentioned in section 2, was compared with four different methods in the literature: SHP [5], Kombi234 [13], ANLCP300 [1], Symbio [8].

The average total cost was calculated to assess the quality and performance of the proposed method, at the same time, the two goals for comparison of the value of the objective function with the other methods.

The total cost is the cost of the objective function and its calculation was performed by the following formula represented below:

$$.cost_{total} = c_1\, obj_{average} + \alpha setup_{average} \qquad (3)$$

where:

$obj_{average}$: is the average number of objects processed in the 100 problems in each class;

$setup_{average}$: is the average of the number of setup among the 100 problems of each class.

The expression used for calculating the variation of the total cost is represented by (4) below:

$$100\ x\ \left(\frac{CostTotalSingleGA - CostTotalcomparemethod}{CostTotalSingleGA}\right) \qquad (4)$$

For the average total cost c_1 e $\alpha = 1$, the SingleGA01 was better in six classes compared to the SHP and other six had a difference less than 3.5%, with respect to ANLCP300 was better in four classes and six presented in a less than 3, 5%. Although it not shown better results when compared with the Kombi234 and Symbio01, the percentage difference the first in five classes was 3%, while the latter this difference has fallen to 1.5% in six classes.

For costs c_1 e $\alpha = 5$, the total cost of SingleGA05 showed better results in nine classes of SHP and two class obtained almost the same result (0.006841%). In the Kombi234 it was higher in seven classes and ANLCP300 in three classes and other five classes the percent difference was 2.3%.

Finally, when confronted with Symbio05, was better in one class in two other class (13 and 14) had nearly identical results and more three showed a difference of only 0.7%.

The average total cost of SingleGA10 (c_1 e $\alpha = 10$) found better results when compared with setup cost $\alpha = 1$ and $\alpha = 5$. The results obtained by SingleGA10 were better than SHP in 12 classes and in more one, class 16, showed a difference of only 0.4%. The SingleGA10 exceeded the Kombi234 in ten classes, the ANLCP300 in three classes, and in the latter, and in further fix classes showed a difference less than 2%. In relation to Symbio10, it was superior in four classes and shown the difference was less than 3% in other seven classes.

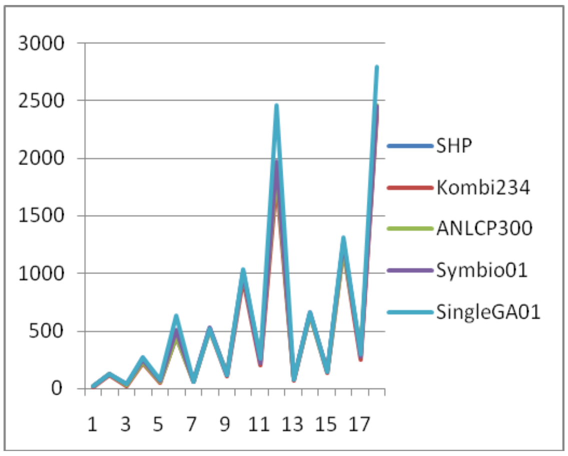
The SingleGA10 was better than Kombi234 and SHP in most classes, a fact that in relation to ANLCP300 and Symbio10 has not occurred.

Considering changes in the setup cost noted that as the cost of setup increases, also the method performance increases. Furthermore, it was observed that for smaller setup cost, the method was similar to the behavior of the SHP and ANLCP300. However, with this increased cost, the method has worsened their performance in relation to ANLCP300 and has been improving steadily in comparison with other methods.

Due to small differences in the percentage changes found, we analyzed graphically (Graphics 1 to 3) the behavior of SingleGA with other methods, and found that the performance of the methods are almost equal. The major differences provided by SingleGA for all the others are in class 12 and 18. The same is true for the comparison between the variations of SingleGA (Graphic 4).

When comparing the results obtained with the SingleGA with other methods, it was noted that the SingleGA presented its best results to the extent that this value (setup cost) was increasing.

The computational time was not considered in this work, since each method has been implemented in different languages and machines, making it impossible to compare them.



Graphic 1: SingleGA01 x other methods

CONCLUSIONS

This paper showed that the study with additional setup cost in the SingleGA method presented a good behavior when compared with other methods, however, the higher the setup cost, the higher the method performance Increasing the setup, method performance worsened compared to ANLCP300 and has been improving steadily in comparison with other methods.

Graphical analysis carried out showed good performance of the method tested with all costs, and make clear that the percentage changes are small for most classes when compared with other methods, thus validating the SingleGA.

The method met the intended goals, however, noted that it is possible to refine it and found to provide superior results with a more detailed study of parameters such as population size, genetic operators and their respective rates.

Based on the computational results presented, it can be said that the new method is promising and competitive in the environment of one-dimensional cutting stock problems.

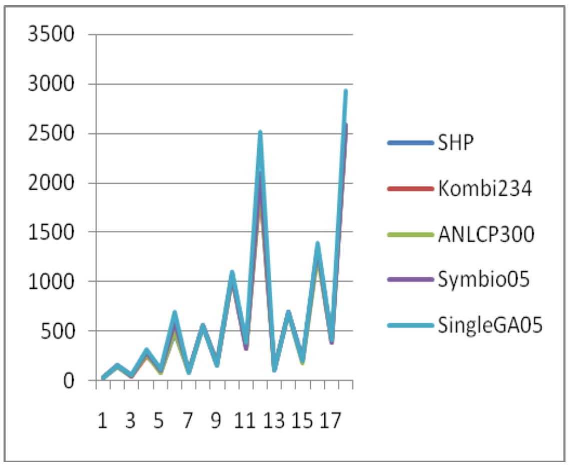
REFERENCES

[1] SALLES NETO, L.L. Modelo não linear para minimizar o número de objetos processados e o setup num problema de corte unidimensional. 2005. Tese (Doutorado) - Instituto de Matemática, Estatística e Computação Científica, Universidade Estadual de Campinas, Campinas, 2005.
[2] KANTOROVICH, L.V. Mathematical methods of organizing and planning production. Management Science, v. 6, p 366-422, 1960.

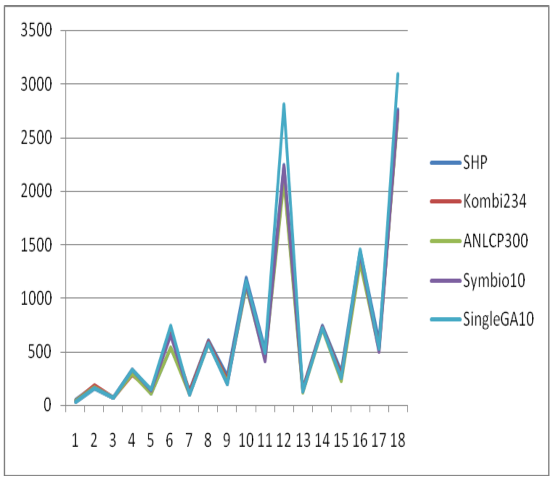
[3] GILMORE, P.C.; GOMORY, R.E. A linear programming approach to the cuting stock problem I. Operations Research, v. 9, p. 849-859, 1961.
[4] ______________________________. A linear programming approach to the cutting stock problem II. Operations Research, v. 11, p. 863-888, 1963.
[5] HAESSLER, R. Controlling cutting pattern changes in one-dimensional trim problems. Operations Research, v. 23, p. 483-493,1975.
[6] HOLLAND, J.H. Adaptation in natural and artificial systems. Michigan: University of Michigan Press, 1975.
[7] BRANDÃO, J. S., COELHO, A. M., VASCONCELOS, J. F., SALLES NETO L. L., PINTO, A. V., Application of Genetic Algorithm to Minimize the Number of Objects Processed and Setup in a One-Dimensional Cutting Stock Problem. International Journal of Applied Evolutionary Computation (IJAEC), Vol 2, Issue 1, 2011. 15 pages: DOI: 10.4018/jaec.2011010103, ISSN: 1942-3594, EISSN: 1942-3608.

[8] GOLFETO, R. R.; MORETTI, A. C.; SALLES NETO, L. L. Algoritmo genético simbiótico aplicado ao problema de corte unidimensional. In: SIMPÓSIO BRASILEIRO DE PESQUISA OPERACIONAL, 39., 2007. Anais do Simpósio Brasileiro de Pesquisa Operacional 2007a .

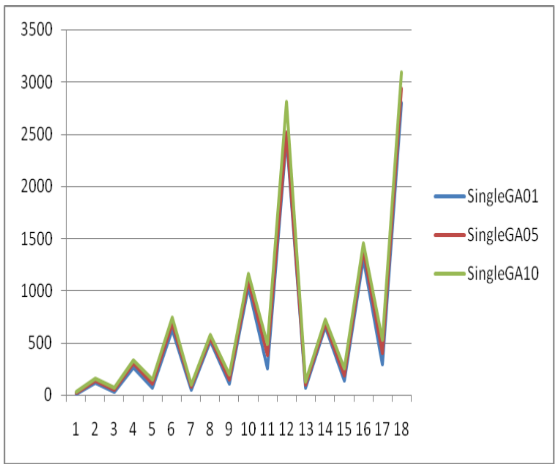
[9]ALLABY, M. Dictionary of ecology. New York: Oxford University Press, 1998.
[10] PIANKA, E. R. Evolutionary ecology. New York: HarperCollins,1994.
[11] WALL, M. A C++ library of genetic algorithm components. Massachusetts Institute of Technology. Mechanical Engineering Department, 1996.
[12] GAU, T.; WASCHER, G. CUTGEN1: A problem generator for the standard one-dimensional cutting stock problem. European Journal of Operational Research, v. 84, p. 572-579, 1995.
[13] FOERSTER, H.; WASCHER, G. Pattern reduction in one-dimensional cutting-stock problem. International Journal of Prod. Res., v. 38, p.1657-1676, 2000.



Graphic 2: SingleGA05 x other methods



Graphic 3: SingleGA10 x other methods



Graphic 4: Variations SingleGA