A genetic algorithm-based heuristic with unequal individuals to tackle the lot-sizing and scheduling problem.

Marcos Mansano Furlan
Universidade de São Paulo-USP
Av. Trabalhador São-carlense, 400 - Centro
CEP: 13560-970 - São Carlos - Brasil
e-mail: mafurlan@icmc.usp.br

Bernardo Almada-Lobo
Universidade do Porto
Rua Dr. Roberto Frias, s/n
CEP: 4200-465 - Porto - Portugal
e-mail: balobo@fe.up.pt

Maristela Oliveira dos Santos
Universidade de São Paulo-USP
Av. Trabalhador São-carlense, 400 - Centro
CEP: 13560-970 - São Carlos - Brasil
e-mail: mari@icmc.usp.br

Reinaldo Morabito Neto
Universidade Federal de São Carlos
Rod. Washington Luís, Km 235
CEP: 13565-905 - São Carlos - Brasil
e-mail: morabito@ufscar.br

Abstract

In this paper we present a novel genetic algorithm-based approach to tackle a lot-sizing and scheduling problem. Mathematical models, commonly used to represent these problems, have the planning horizon divided in periods and sub-periods. Adjusting the right number of sub-periods becomes a hard task, since higher values create flexible environments and smaller values reduce the problem size and problem complexity. The key idea of our method is to take some advantages about adjusting the number of sub-periods. In this way, a genetic algorithm which allows for the use of individuals with different size was developed. This feature was performed to maintain the flexibility, as well as, reducing the computational load. Preliminary results show the benefits obtained when variable number of sub-periods is used, nevertheless, the tightness of the lower bounds becomes more promising when the number of products increases. More studies are necessary to determine an ideal lower bound rule.

KEYWORDS: **Metaheuristics, OR in Administration & Production Management, OR in industry**.

1 Introduction

The production planning and scheduling aims improving the use of production resources based on pre-defined drivers. These goals can be cost reduction or profit increase. In some cases, special characteristics of the production environment can be considered and desire solution feature may be weighed in the objective function. The main features must be guaranteed by the mathematical model turning it as realistic as possible, however maintaining its tractability.

According to Karimi *et al.* (2003), the production planning covers three time range decisions: long-term, medium-term and short-term. Long-term decisions consist of strategic decisions. Medium-term decisions involve material requirements planning (MRP), lot-size definitions, and other tactical decisions. Short-term decisions involve more detailed planning and is related to operational requests. In this terms, lot-sizing and scheduling problems, as the one tackled in this paper, stay in medium to short term decisions. In other words, it incorporates operational and tactical decisions.

The lot-sizing mathematical models can be divided in two big groups: big bucket and small bucket problems. Big-bucket problems commonly consider the production of a higher number of items in larger periods and the scheduling problem needs some additional constraints to be performed. Small-bucket problems address simultaneously the lot-sizing and the scheduling problems due to the structure of its sub-periods, where the number of items produced is limited (at most two). According to Drexl & Kimms (1997), small bucket problems includes: the discrete lot sizing problem (DLSP), the proportional lot sizing and scheduling problem (PLSP) and the general lot sizing and scheduling problem (GLSP).

The lot-sizing and scheduling problem in an integrated pulp and paper Portuguese mill was tackled in the paper Santos & Almada-Lobo (2012). The authors proposed a mixed integer programming formulation (MIP) based on the GLSP, first presented by Meyr (2000), to deal with a multi-stage scenario. The two-level time-structure of the GLSP synchronizes the resources of the whole process. The process includes the pulp plant, the paper plant and the recovery plant as illustrated in Figure 1. The digester produces the virgin pulp to supply the paper machine, as well as, the weak black liquor to feed the recovery plant. The paper plant can also be supplied by an associated recycled pulp mill, in case the paper produced requires this kind of pulp. The paper machine can process at most one type of paper in each indivisible sub-period, where each type of paper is of a certain grammage and contains a given percentage of recycled pulp. Throughout this paper, and for sake of simplicity, we use the term grammage to refer to a certain paper type with all its characteristics. Intermediate stocks of virgin pulp, recycled pulp and liquors are considered with minimum and maximum limits defined to each tank. The products obtained in the paper plant are the big paper rolls called jumbos; the steam and electric energy are produced at the recovery boiler and associated turbines, respectively. Figure 1 illustrates the considered mill. To a more detailed explanation the underlying manufacturing process and operational production planning and scheduling, the readers are referred to Santos & Almada-Lobo (2012).

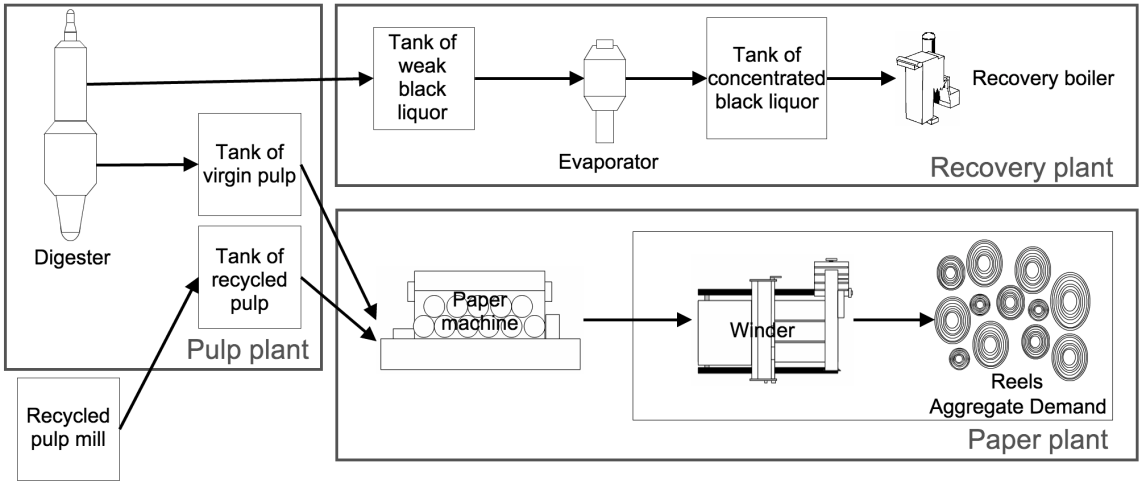


Figure 1: The integrated pulp and paper mill.

This paper focus on the lot-sizing and scheduling problem from the pulp and paper industry and has the main purpose of proposing a novel heuristic that takes advantage from the characteristics of the GLSP models. The key idea is to design a genetic algorithm-based method that can handle individuals with different number of sub-periods in their chromosome, benefiting from the problem size reduction. Some computational experiments are performed and the solutions with and without this feature are compared. This kind of approach is not common in the literature and presents promising preliminary results. The genetic algorithm proposed is detailed in Section 2. Tests instances and preliminary results are presented in Section 3 . Finally, Section 4 draws the main conclusions and some appointments for future research.

2 Genetic Algorithm

This section describes a genetic algorithm developed to tackle the lot-sizing and scheduling problem appearing at the pulp and paper mill presented in Santos & Almada-Lobo (2012). The problem representation takes care of two simultaneous scheduling problems. Well known operators of selection and crossover were developed together with three mutation operators. Our approach allows for individuals that are decoded in solutions with different number of sub-periods overcoming the traditional representation of solutions that tend to overestimate this number.

Genetic algorithm became popular after the book of John Holland in 1975 entitled *Adaptation in Natural and Artificial Systems*. In the following decades, several papers applied and developed the theory of genetic algorithms as, for example, Goldberg (2002) that studied the building blocks theory and Whitley (1994) that analysed the disruption caused by some operators.

A genetic algorithm is basically composed of initial population, selection and crossover operators. Mutation operators are very common, but not mandatory. The selection operator guides the search process across promising regions and crossover operator blends chromosomes looking to generate better individuals (offspring). The mutation operator introduces some variability and enables the method to produce better solutions also.

A review of genetic algorithms applied to lot sizing problems was conducted in Goren *et al.* (2010). The authors discuss some characteristics of the genetic algorithms, such as the types of chromosome representation, evaluation, selection, and so on. Some recent papers (Toledo *et al.* , 2009; Amorim *et al.* , 2011; Mohammadi *et al.* , 2011; Toledo *et al.* , 2012; Camargo *et al.* , 2012; Chan *et al.* , 2012) address different lot-sizing and scheduling problems using genetic algorithm-based methods. For instance, Mohammadi *et al.* (2011) propose a hybrid method composed of a genetic algorithm embedded in a rolling horizon heuristic to tackle the lot-sizing and scheduling problem in a permutation flow shop environment with sequence-dependent setups. The authors propose a method that solves the problem iteratively in a rolling horizon fashion using a genetic algorithm to solve the mixed integer sub-problems and relaxing the integrality in the reminder periods.

Our genetic algorithm uses the *2*-point crossover and performs *2*-tournament to chose individuals to survive and to become parents. The adjustment of the number of sub-periods of each individual is a new feature brought up in this paper. In some situations the number of sub-periods is overestimated, generating difficulties in terms of computational load without gains in terms of quality solution. The acceptance of individuals with different number of sub-periods may overcome those barriers. Note that the number of production slots of any solution is limited by the number of sub-periods. The complete pseudo-code is presented in Algorithm 1 and each component is detailed afterwards.

The procedure *Initialize* constructs the initial population. The *Greedy Sequencing Heuristic* improves the sequence of the initial solutions by rearranging the sequence of changeovers inside a period to reduce the setup cost. It is done for every period in a rolling horizon fashion and in the majority of the cases, this method reduces the setup costs without changing other objective function terms. In extreme situations it may change production rates, in case the solution stays in the feasibility frontier.

Selection, *Mutation*, *Evaluation* and *Crossover* are common procedures adopted by genetic algorithms. The first is used to select the individuals that survive to the next generation, according to a predefined pressure of selection (percentage of population replaced by the offspring). The second is used to mutate the offspring, introducing the variability and the third evaluates the individual fitness according to the quality solution (objective function) and the sum of infeasibility. The fourth

Algorithm 1: The hybrid genetic algorithm pseudo-code.

```
1  Initialize(pop);
2  GreedySequencingHeuristic(pop);
3  Evaluate(pop,β);
4  Selection(pop);
5  while time limit is not reached do
6  |  Crossover(pop);
7  |  Mutation(pop.offspring, mutprob);
8  |  Evaluate(pop,β);
9  |  Selection(pop);
10 end
11 Return the best solution;
```

selects and combines parents to produce new solutions. The *Crossover* procedure is run as many time as necessary to maintain the population size. For example, if the *Selection* reduces the population by 10 solutions, the procedure *Crossover* is called as many times as necessary to produce 10 new individuals.

2.1 Individual

The underlying problem includes two integrated scheduling problems (digester and paper machines). The first problem defines the digester speeds in each sub-period in a discrete range of values between minimum and maximum values allowed. The discrete steps are defined by the predefined parameter Φ (in RPM). For example, using a digester that has minimum speed of 10 RPM and $\Phi = 0.25$, the set of possible speeds is $\{10, 10.25, 10.5, ...\}$. The second scheduling problem defines the production sequence on the paper machine. In both cases, the idea is to define the grammage/speed chosen in every sub-period in the whole planning horizon for the paper machine/digester.

Integer vectors are used to treat this integrated problem, representing partially the solution (binary variables relate to the digester speed and paper machine setup). The resulting sub-problem is solved by an LP solver. To ensure that all individuals respect the smooth shift on the digester speed, the related vector defines the variation of the speeds between consecutive sub-periods instead of explicitly their indices. This gradient is limited by parameter Δ. For example, considering $\Phi = 0.25$ and $\Delta = 1$ (which denotes that speed can vary at most one position), it means that speed can vary at most 0.25 RPM between consecutive sub-periods. To avoid negative values in the chromosome, we relocate the range of values, starting by the bigger allowed speed decrease represented by zero. In the last example, a value zero in the chromosome means a speed reduction of 0.25 RPM, a value one means that the speed is kept and, finally, value two means a speed increase of 0.25 RPM. Furthermore, when a calculated speed exceeds the maximum or the minimum allowed velocities, it is replaced by the violated limit. On the other hand, the other scheduling vector defines the grammage produced in each sub-period of the planning horizon. Smooth changeovers are also desirable in this case, however, positive setup costs and times reinforce this smoothing.

Figure 2 illustrates one individual decoding a problem with 3 periods, 12 sub-periods in total, $\Delta = 2$, $\Phi = 0.5$ and initial speed of 12.0 RPM. One characteristic that could be viewed in this figure is the variation in the number of sub-periods. The light gray cells define the active genes, that affect the solution scheduling, dark gray cells define the "passive" genes, that are overlooked in the decode process, and the white cell that are illustrative values and not required to be store, but can be obtained implicitly (sub-periods) or can be calculated (speeds). We assume that disabled genes could transmit information to the next generations and, when an offspring with greater number of sub-periods is produced, some improvement could be reached. The number of active genes in each period is defined by a "dominant gene" that ranges from a minimum number of sub-periods to the original number of sub-periods. In the example of Figure 2, this dominant gene has value three and the maximum possible value is four.

Periods		0	1				2				3			
Sub-periods		0	1	2	3	4	5	6	7	8	9	10	11	12
Digester	Variation	*	2	3	4	2	1	0	2	3	1	1	3	4
	Speeds	12.0	12.0	12.5	13.5	*	13.0	12.0	12.0	*	11.5	11.0	11.5	*
PM (grammages)		*	1	1	3	3	5	4	3	1	3	2	6	4

Figure 2: Individual representation with the digester speed variation and the paper machine scheduling. The digester speed is calculated to illustrate the decode of the first vector.

2.2 Initial population

The initial population is commonly randomly generated in the literature. For each individual and each sub-period a random type of paper is chosen to be produced, independently of the others. The vector of speed variation is also randomly generated within the allowed range. This kind of initial population produces a good variability and guarantees a good initial diversity.

2.3 Greedy sequencing heuristics

A greedy heuristic was implemented to improve the initial production sequences on the paper machines. This procedure reorders the production sequences between sub-periods in the same period to reduce the setup costs without increasing inventory and backlog costs. The method starts from the first period onwards by removing all the gene values and then re-inserting the value with the smallest setup cost, according to the last grammage allocated. This process is done iteratively to each period in a rolling horizon fashion. For example, consider an individual with production sequence [1,4,2], initial grammage 1 and setup costs proportional to the difference between the grammage indices. From the set 1,4,2, the method selects, for instance, item 1 in the first iteration (null setup cost), then item 2 and, finally, item 4.

2.4 Decoding and fitness evaluation

The decoding of individual is defined for both scheduling problems. The machine papers scheduling are directly defined by the product indices. For example, in Figure 2 the paper machine (PM) is assigned to produce grammage [1,1,3] in the first period, in this sequence. The speed definition for the digester requires some considerations. The initial speed is an important information, since the vector defines the variation of the speed during the time with the maximum variation of Δ. All speeds are defined according to the last velocity, the respective variation and the speed limits. The resulting sub-problem is a linear programming problem, obtained by fixing the integer variables in the mathematical model (see Santos & Almada-Lobo (2012)). These sub-problems are addressed by an LP solver.

The fitness function is composed by two parts and is presented in (1). The first part is composed by the objective function from the individual p, called $f(p)$ which is defined in (2), divided by the average objective function ($\overline{f}$) of the whole population. The second part of the fitness function defines the relative unfitness or infeasibility calculated through the division of the individual infeasibility $u(p)$ by the biggest unfitness found in the same generation. The objective function (2) is composed by four parts: the first part is the sum of holding costs, the second part is the sum of backlogging costs, the third is the sum of changeover costs and the last part creates a stimulus to maximize the steam production.

$$fitness(p) = \frac{f(p)}{\overline{f}} \cdot \left(1 + \frac{u(p)}{\max_{q \in Pop} u(q)}\right) \tag{1}$$

$$f(p) = \sum h^{+}_{jt} \cdot IG^{+}_{jt} + \sum h^{-}_{jt} \cdot IG^{-}_{jt} + \sum sc_{kjm} \cdot Z_{kjsm} - \sum \gamma \cdot O^{steam}_{s} \tag{2}$$

2.4.1 Infeasibility

Many methods in the literature allow for infeasible solutions to maintain variability (methods based on solution pools) and/or to overcome local optima. The global optima solution could stay in the border of the feasibility region and/or the solution set could have disjunct areas and each area may

have one or more local optima. In these cases, the infeasible treatment increases the chances to reach the global (near global) optimum solution.

Our genetic algorithm allows for some penalized infeasibility to maintain the population variability. Furthermore, the majority of the initial population solutions are infeasible, and a recovery procedure could require high computational time. The infeasibility was calculated by relaxing the constraints that controls inventory of virgin pulp (3), inventory of recycled pulp (4), inventory weak black liquor (5) and the minimum lot-size requirement (6). The intermediate inventories bounds can be a production limit, according to the production scheduling and the digester speeds. With the relaxation we can measure these violations. The minimum lot-size constraint is also relaxed, because in the case of a high number of changeovers on the paper machine in one period, the sum of times to produce lots to reach the minimum and the setup times could be greater than the capacity (number of hours) in this period.

$$X_s^{dig} + I_{s-1}^{virg} = O_s^{virg} + I_s^{virg} \qquad s \in S \tag{3}$$

$$X_s^{recy} + I_{s-1}^{recy} = O_s^{recy} + I_s^{recy} \qquad s \in S \tag{4}$$

$$X_s^{liquor} + I_{s-1}^{liquor} = O_s^{liquor} + I_s^{liquor} \qquad s \in S \tag{5}$$

$$X_{js} \geq m_j \cdot (Y_{js} - Y_{j,s-1}) \qquad j \in K,\ s \in S, \tag{6}$$

where X_s^{dig}, X_s^{recy} and X_s^{liquor} are the amount of virgin pulp, recycled pulp and weak black liquor, respectively. The inventories levels are presented by I_s^{virg}, I_s^{recy} and I_s^{liquor} and the outputs are defined by O_s^{virg}, O_s^{recy} and O_s^{liquor}. The paper machine productions are defined by X_{js} and Y_{js} define the setup of the machine.

2.5 Crossover

The crossover operator blends the parents chromosome to produce offspring with expected better individuals. The building blocks theory presented by Goldberg (2002) defines that blocks of genes which benefit the fitness of an individual tends to be maintained in the population and grow throughout the generations. Other papers discuss the schemata theory with binary, integer and real alphabets (Goldberg, 1990; Whitley, 1994).

We choose the 2-point crossover operator to blend the solutions in our genetic algorithm supported by Spears & De Jong (1990), which shows that uniform crossover is more disruptive than the 2-points crossover for order-3 schemata. Furthermore, 1-point crossover and other n-point crossover operators are also more disruptive than 2-point operator. The Figures 3 illustrates an example of 2-points operator.

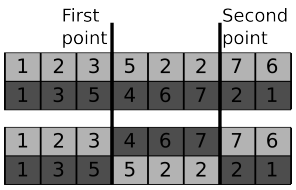


Figure 3: An example of 2-points crossover operator.

To develop solutions blending with different number of sub-periods, we consider that this combination is performed considering both passive and active genes, i.e., the whole vectors are blended. An integer value that defines the number of sub-periods is also passed to the offspring. For example, mating parents with 3 and 4 sub-periods will generate a child that considers 3 sub-periods and other that contains 4 sub-periods. This approach is used within the selection process to adjust the sizes that have better solutions during the whole process, i.e., offspring with varied size are compared in terms of the solution quality, and the best individuals remains, regardless of their number of sub-periods.

The crossover points are chosen independently to both speed digester and paper machine vectors. The idea is to have a more general operator to blend the solutions, since one parent can have better encode to one scheduling and worse to others and, therefore the chances that both parents contribute

mostly with their better parts exists. The solution set with the same cut points for both vectors is a subset from the operator with different cut points.

2.6 Mutation

We use three mutation operators in this paper. The first and the second are of general purpose and well known in the literature. The third is related to the use of individuals with different size. It was developed, as a selection process counterpart, to increase the individuals size, while the selection tends to choose small individuals (smaller setup costs). All the mutation operators are presented below.

2.6.1 Simple mutation

When a gene is selected to mutate, a different value from the original is randomly generated. Consider, for example, the scheduling of 8 different papers and that a gene with value equal to 3 is chosen to mutate. The mutation generates a random value in the set $[1, 2] \bigcup [4, 8]$. It can be seen as the integer version for the flip bit mutation.

2.6.2 Swap mutation

The swap mutation exchanges values between two genes. When a gene is selected to mutate, the partner is chosen randomly and the values are swapped only in case they have different values to each other. Otherwise, a simple mutation is applied to the first gene. This process off change is done between two genes of the same vector, since the range of values can be different between vectors.

2.6.3 Sub-periods mutation

This mutation, in general, increases the number of active genes, i.e., increment by one the number of sub-periods considered by an individual in each period. If the original number is the maximum value, it is reduced by one, i.e., the mutation always occur when an individual is chosen to mutate, even when it has all the genes actives. When the number of sub-periods is fixed (maximum and minimum are equal) this mutation is skipped.

2.7 Selection

All the selections performed in this genetic algorithm are done by *2*-tournament operators. Tournament operator is attributed to Brindle (1981) and consists in performing tournaments with individuals randomly chosen to obtain the best solution among the k solutions compared. Some papers in the literature use tournament as the selection process.

In this paper, survival and crossover selection variate in the number of tournaments performed: in the survival selection this process is done until the number of survivors reaches a pre-defined number of individuals; in the crossover selection two parents are selected to produce two new solutions until the population reduced by the selection is newly fulfilled.

Other relevant differences: in the survival selection, we are interested also in saving the best solution, that is copied to the new population without tournament; in the crossover selection, one individual can produce children more than one time, however, in survival selection, individuals that were already copied to the survival population cannot be selected again, i.e., an individual can only be selected once to survive at each generation.

3 Computational experiments

The instances generator is a modification from the generator proposed by Figueira *et al.* (2013). The instances are constructed based on the data from the study case of Santos & Almada-Lobo (2012) and random variations are done on the demand profiles and initial stocks of intermediate items. The demand profile ranges from 65% to 85% of the paper machine capacities in a uniform distribution. The initial stocks also vary according to a uniform distribution; they range from 80% to 120% of the original data values, always considering the process capacity bounds.

In this generator, we include a procedure to calculate the price of each grammage, based on the average sales prices from the original data. The paper prices are used to calculate the setup, inventory and backlog costs. Table 3 shows the instance features that were generated. For each number (#) of periods we use all the values in the set of grammages and of periods totalling in 16 classes of tests. Five instances were generated to each class.

# of periods	# of grammage	# of sub-periods	# of instances
7	4,8	3,4,5,6	5
15	8,16	3,4,5,6	5

Table 1: Instances characteristics.

3.1 Computational results

All the computational tests were performed in a computer with a intel Core i5 2300 2.8 GHz CPU with 4 GB of RAM memory. The method was coded in C++ with the support of OPL Cplex interface and using the MIP/LP solver Cplex to tackle the linear sub-problems. All the computational tests were limited to 3600 seconds and the method was run 5 times per instance and variant. The genetic algorithm parameters were determined by preliminary tests and the values used were: Population size = 100, Selection pressure=10%, Mutation Probability=0.8%, Crossover Probability= 99%.

In this paper, we test 3 variants of our method that are compared below. The first variant has the Number of Sub-periods Fixed (NSF), i.e., the original size ($\mid S \mid$) in maintained for all individuals. The second has Number of Sub-periods limited at 3 (NS3) which means that individuals size ranges from 3 to $\mid S \mid$. The third has Number of Sub-periods Released (NSR), which means that the number of sub-periods ranges from 1 to $\mid S \mid$.

We present below the computational results with the support of performance charts developed by Dolan & Moré (2002). With these charts, we can compare some approaches or variants of the same approach for many instances. The key idea is, for each instance, to define the gap between the method solutions and the smallest value; and then to calculate the percentage of instances with at most τ gap. The τ values are expressed in the abscissa and the percentage in the ordinate. A performance curve with greater values determines that a method delivers better results than others. For example, the values in $\tau = 1$ determine the percentage of best solutions for each method. It is important to remember that draws are computed to both methods.

Figure 4 presents the aggregate results for all the instances tested. In general, the NSR provides the best results, since its curve is on top of all the others. The number of best solutions (see values $\tau = 1$) is smaller than the obtained by NS3, however the number are quite similar. The results of NSF are clearly lower than the other variants. It supports our assumption that allowing individuals with different size is promising approach, and may be applied to other GLSP-based problems.

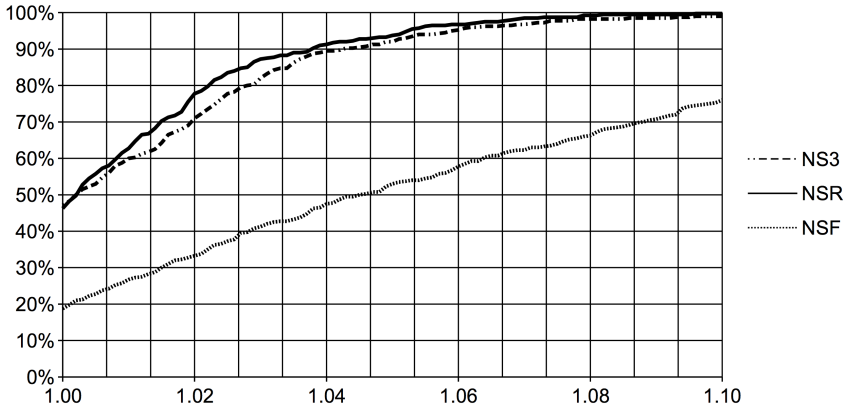
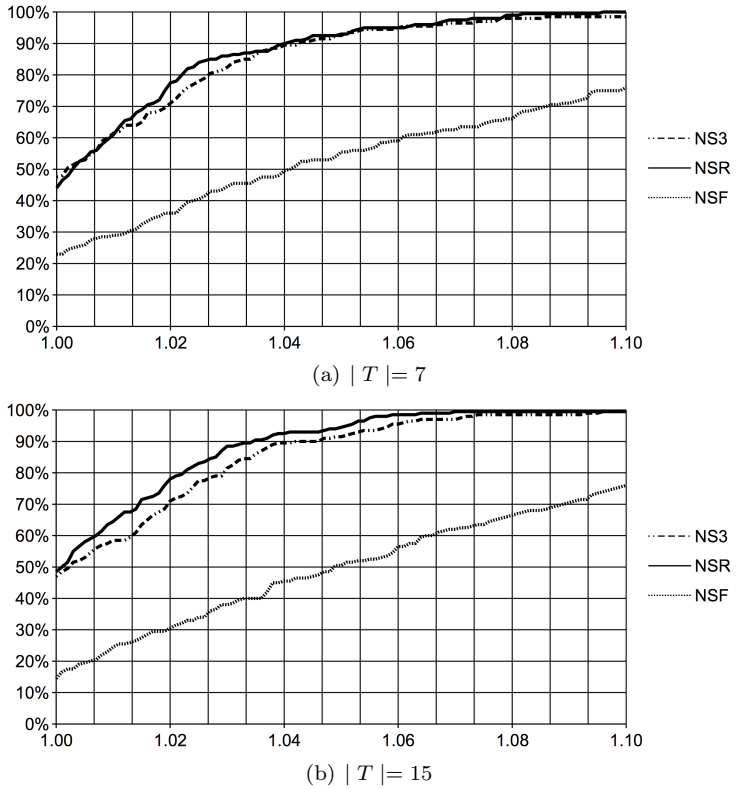


Figure 4: Performance chart considering all instances.

Figure 5(a) and Figure 5(b) present the results separated by periods size. Figure 5(a) shows the

performance chart for instances with planning horizon of 7 days. These results are similar when compared with those of Figure 4, however the results from variants NS3 and NSR seem to be closer. On the other hand, Figure 5(b) presents a bigger difference between both NS3 and NSR, maintaining the same tendency.



(a) $|T| = 7$

(b) $|T| = 15$

Figure 5: Performance charts per number of periods.

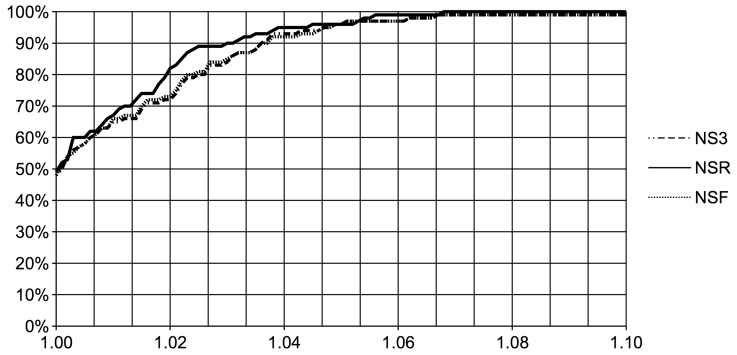
Figure 6 shows the performance charts split by number of sub-periods. The results present the growing difference between the variant NSF and the other two with the increasing number of sub-periods. It is an expected characteristic, since the difficult associated to the problem grows with the number of sub-periods from the planning horizon. Another interesting result, in this chart, is that NSR performs better with small number of sub-periods, however for instances with 6 sub-periods (see Figure 6(d)) the variant NS3 is more competitive. These results demonstrate that, in some cases, a better lower bound in the size can improve the method.

Figure 7 presents the results divided by quantity of grammage. The interesting result appears in Figure 7(c), where NR3 has a significant superior performance compared to NSR. For example, NS3 has around 65% of best solution, while NSR has less than 31% and NSF around 20%. These results indicate that a bigger minimum amount of sub-periods become more relevant when the number of grammage grows, however the flexibility introduced by individuals with different size stills exists.

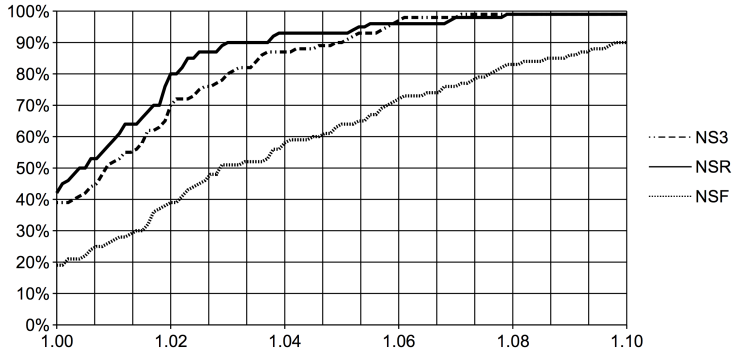
4 Conclusions

In this paper we study the impact of dynamically adjusting the number of sub-periods by the solution approach in execution time. The problem consists in defining the production of different types of paper, meeting the limited resource capacities, considering the mill characteristics and attending the clients demand with the smallest sum of inventory, backlogging and setup costs. The main challenge is to reduce the problem difficulty by reducing the number of sub-periods, without loss of detail and maintaining the capacity to attend the demands properly.

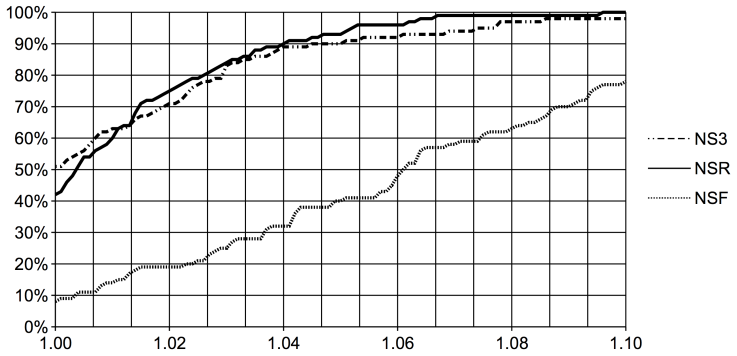
The computational results demonstrate the advantages in use of individuals with different size. Even in cases where the number of products is superior for the maximum number of sub-periods,



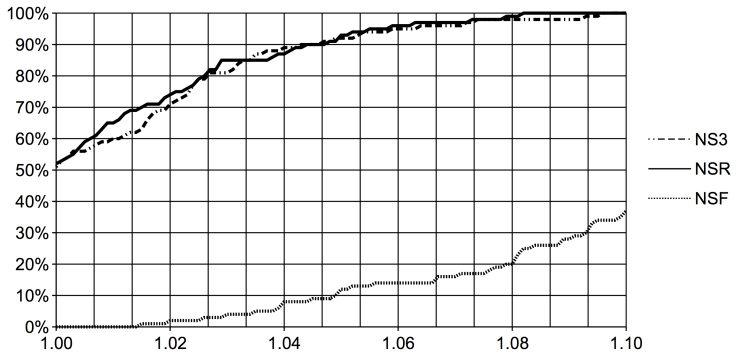
(a) $|S| = 3$



(b) $|S| = 4$



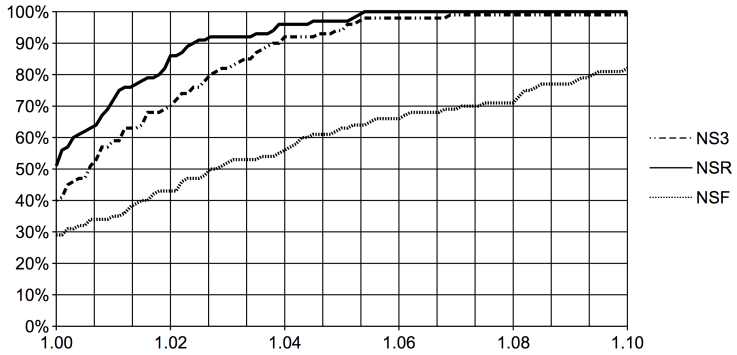
(c) $|S| = 5$



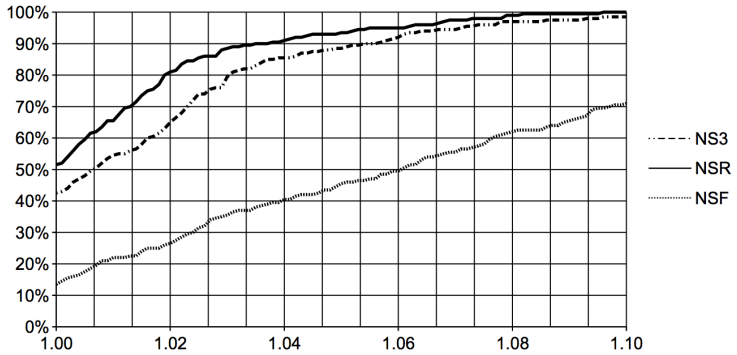
(d) $|S| = 6$

Figure 6: Performance charts per number of sub-periods.

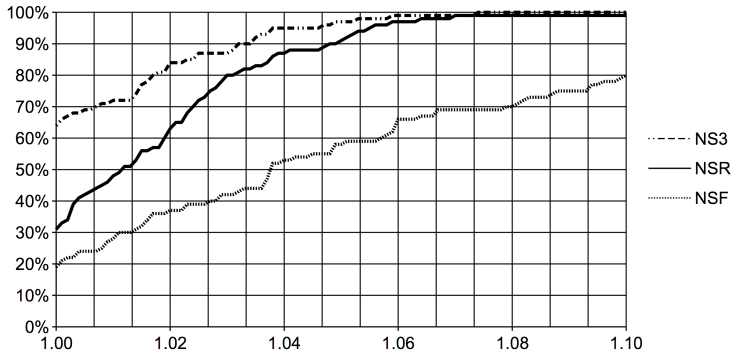
flexible variants have better results. More computational experiments are required to support other analyses and comparisons with literature method are mandatory. We expected to compare the complete results, in a future publication, with the methods proposed by Santos & Almada-Lobo



(a) $|K| = 4$



(b) $|K| = 8$



(c) $|K| = 16$

Figure 7: Performance charts per number of grammage.

(2012) and Figueira *et al.* (2013) and a commercial solver.

Some studies about the convergence process and more flexible ways to represent individuals can be addressed in future research, such as considering periods with different number of sub-periods in one individual. This approach could be more interesting in cases where more variation in the demand between consecutive periods exists. The use of other features in the genetic algorithm is also relevant and needs more research as, for example, using the problem knowledge and diversification process associated with population-based approaches. Furthermore, this size variation can be performed with other solution methods.

Acknowledgements

We are grateful to Gonçalo Figueira for his support with the instance generator. This paper was partially funded by Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) and by Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP).

References

Amorim, Pedro, Antunes, Carlos H, & Almada-Lobo, Bernardo. (2011), Multi-objective lot-sizing and scheduling dealing with perishability issues, *Industrial & Engineering Chemistry Research*, 50, 3371–3381.

Brindle, A., *Genetic Algorithm for Function Optimization*, Ph.D. thesis, University of Alberta, 1981.

Camargo, Victor CB, Mattiolli, Leandro, & Toledo, Franklina. (2012), A knapsack problem as a tool to solve the production planning problem in small foundries, *Computers & Operations Research*, 39, 86–92.

Chan, Hing Kai, Chung, Sai Ho, & Chan, Tak Ming. (2012), Combining genetic approach and integer programming to solve multi-facility economic lot-scheduling problem, *Journal of Intelligent Manufacturing*, 23, 2397–2405.

Dolan, Elizabeth D., & Moré, Jorge J. (2002), Benchmarking optimization software with performance profiles. *Mathematical Programming*, 91, 201–213.

Drexl, Andreas, & Kimms, Alf. (1997), Lot sizing and scheduling – survey and extensions, *European Journal of Operational Research*, 99, 221–235.

Figueira, Gonçalo, Santos, Maristela Oliveira, & Almada-Lobo, Bernardo. (2013), A hybrid VNS approach for the short-term production planning and scheduling: A case study in the pulp and paper industry. *Computers & Operations Research*, 40, 1804–1818.

Goldberg, David Edward. (1990), Real-coded genetic algorithms, virtual alphabets, and blocking, *Urbana*, 51, 61801.

Goldberg, David Edward, *The design of innovation: Lessons from and for competent genetic algorithms*. Kluwer Academic Publishers, Norwell, MA, USA, 2002.

Goren, Hacer Guner, Tunali, Semra, & Jans, Raf. (2010), A review of applications of genetic algorithms in lot sizing, *Journal of Intelligent Manufacturing*, 21, 575–590.

Holland, John Henry, *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*, University of Michigan Press, Ann Arbor, MI, USA, 1975.

Karimi, B., Ghomi, S. M. T. Fatemi, & Wilson, J. M. (2003), The capacitated lot sizing problem: a review of models and algorithms, *Omega*, 31, 365 – 378.

Meyr, Herbert. (2000), Simultaneous lotsizing and scheduling by combining local search with dual reoptimization, *European Journal of Operational Research*, 120, 311–326.

Mohammadi, Mohammad, Ghomi, SMT Fatemi, & Jafari, Niloofar. (2011), A genetic algorithm for simultaneous lotsizing and sequencing of the permutation flow shops with sequence-dependent setups, *International Journal of Computer Integrated Manufacturing*, 24, 87–93.

Santos, Maristela Oliveira, & Almada-Lobo, Bernardo. (2012), Integrated pulp and paper mill planning and scheduling, *Computers & Industrial Engineering*, 63, 1 – 12.

Spears, William M, & De Jong, Kenneth A. (1990), *An analysis of multi-point crossover*, Technical report DTIC Document.

Toledo, Claudio Fabiano Motta, França, Paulo Morelato, Morabito, Reinaldo, & Kimms, Alf. (2009), Multi-population genetic algorithm to solve the synchronized and integrated two-level lot sizing and scheduling problem. *International Journal of Production Research*, 47, 3097–3119.

Toledo, Claudo Fabiano Motta, Arantes, Márcio da Silva, Oliveira, Renato Resende Ribeiro, & Almada-Lobo, Bernardo. (2012), Glass Container Production Scheduling through Hybrid Multi-population based Evolutionary Algorithm, *Applied Soft Computing*, 13, 1352–1364.

Whitley, Darrell. (1994), A genetic algorithm tutorial, *Statistics and computing*, 4, 65–85.