A Multi-Population Genetic Algorithm to Solve the Synchronized and Integrated Two-Level Lot Sizing and Scheduling Problem

Alf Kimms, Claudio F. Motta Toledo, Paulo M. Franca, Reinaldo Morabito

A Multi-Population Genetic Algorithm to Solve the Synchronized and Integrated Two-Level Lot Sizing and Scheduling Problem

Journal:	*International Journal of Production Research*
Manuscript ID:	TPRS-2006-IJPR-0729.R1
Manuscript Type:	Original Manuscript
Date Submitted by the Author:	31-May-2007
Complete List of Authors:	Kimms, Alf; University of Duisburg-Essen, Mercator School of Management Motta Toledo, Claudio; Universidade Federal de Lavras, Departmento de Ciencia da Computacao Franca, Paulo; Universidade Estadual de Campinas, Departamento de Engenharia de Sistemas Morabito, Reinaldo; Universidade Federal de Sao Carlos, Departamento de Engenharia de Producao
Keywords:	LOT SIZING, GENETIC ALGORITHMS, PROCESS INDUSTRY
Keywords (user):	

Multi-Population Genetic Algorithm to Solve the Synchronized and Integrated Two-Level Lot Sizing and Scheduling Problem

C. F. M. TOLEDO†, P.M. FRANÇA‡, R. MORABITO§ and A. KIMMS*
†Departamento de Ciência da Computação, Universidade Federal de Lavras, C.P. 3037, 37200-000, Lavras, MG, Brazil, claudio@dcc.ufla.br

‡Faculdade de Engenharia Elétrica e de Computação, Universidade Estadual de Campinas, C.P. 6101, 13083-852, Campinas, SP, Brazil, franca@densis.fee.unicamp.br

§Departamento de Engenharia de Produção, Universidade Federal de São Carlos, C.P. 676, 13565-905, São Carlos, SP, Brazil, morabito@power.ufscar.br
*Dept. of Technology and Operations Management, University of Duisburg-Essen, 47048 Duisburg, Germany, alf.kimms@uni-duisburg-essen.de

This paper introduces an evolutionary algorithm as a procedure to solve the Synchronized and Integrated Two-Level Lot Sizing and Scheduling Problem (SITLSP). This problem can be found in some industrial settings, mainly soft drink companies, where the production process involves two interdependent levels with decisions concerning raw material storage and soft drink bottling. The challenge is to simultaneously determine the lot-sizing and scheduling of raw materials in tanks and soft drinks in bottling lines, where setup costs and times depend on the previous items stored and bottled. A multi-population genetic algorithm approach with a novel representation of solutions for individuals and a hierarchical ternary tree structure for populations is proposed. **Computational tests include comparisons with an exact approach for small-to-moderate sized instances and with real-world production plans provided by a manufacturer.**

Keywords: lot-sizing and scheduling, production planning, combinatorial optimization, genetic algorithm, soft drink manufacturing.

1. Introduction

The production planning problem studied in this article was motivated by a real situation found in a Brazilian soft drink company. The production process in this kind of company usually has two interdependent levels. At the first level, it must be decided how many raw materials have to be prepared and stored in each one of the available tanks and when. At the second level, it must be decided how many products have to be produced in each one of the available production lines and when. A schematic view of the two-level production process is shown in Figure 1.

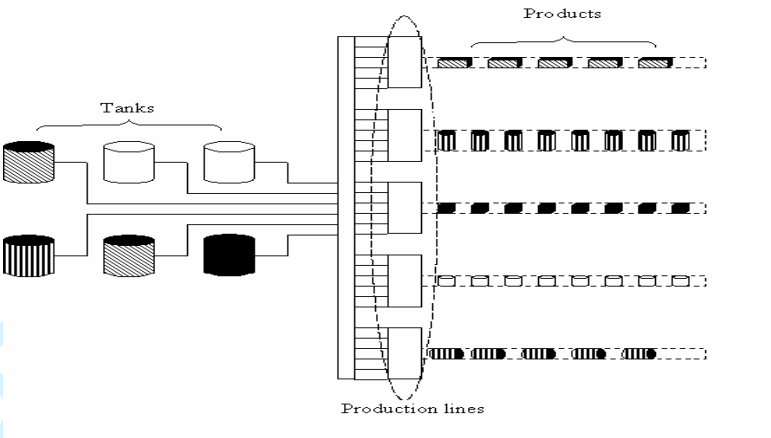


Fig. 1. The two-level production process.

A lot-sizing and scheduling problem has to be solved in each part of these two-levels. The product mentioned is defined by the flavor of the soft drink and the type of container, such as glass bottles, plastic bottles (PET), cans or bag-in-boxes of different sizes. Each line can not produce more than one product at a time, but various products can share a common production line and various lines can produce the same product in parallel. There is a sequence-dependent setup time of up to several hours whenever a different product switches in a line. However, if the production of the same product is **paused** for a while and continued after some idle time, then no setup is required.

The raw material is the soft drink that is bottled on a production line. This soft drink comes from storage tanks with a limited storage capacity. A tank is only filled up again when it is empty. Therefore, a sequence-dependent setup time occurs from time to time to clean and fill up a tank, even if the soft drink replaced is the same as before. Nothing can be pumped to a production line from that tank during the setup time. One tank can be connected to production lines which use the same raw material. Analogously, one production line can be connected with various tanks (containing the same raw material) at the same time.

In general, weekly demands within a time horizon of four weeks have to be met. If an excessive number of products is produced at the end of each week, they are stored which incurs inventory costs. Moreover, there are also costs related to storing raw material in tanks, the production process of each product and the production process of each raw material. Sequence-dependent setup costs occur for products and raw materials which are proportional to the sequence-dependent setup times in lines and tanks, respectively.

The challenging aspect of the problem addressed is the combination of all these issues in an interdependent two-level problem. For these reasons, the present problem is called the Synchronized and Integrated Two-Level Lot Sizing and Scheduling Problem (SITLSP). This problem covers various lot-sizing and scheduling issues that have been dealt with in the literature before. The capacitated lot-sizing and scheduling problem (CLSP) is a topic of broad interest and has attracted many researchers. **Models and algorithms are discussed in depth by Karimi *et al.* (2003) for the single-level lot-sizing problem with uncapacitated and capacitated constraints. A literature review for the single-level, the continuous time and the multi-level lot-sizing and scheduling problems can be found in Drexl and Kimms (1997), where the differences on formal models are presented.** A discussion of lot-sizing and scheduling with sequence-dependent setup costs or sequence-dependent setup times is reported by e.g. Clark and Clark (2000), Fleischmann (1994), Gupta and Magnusson (2005), Haase and Kimms (2000) and Meyr (2000). Publications addressing multi-level lot-sizing (scheduling) problems are e.g. Berreta *et al.* (2005), França *et al.* (1997), Kimms (1997). Studies regarding these problems with parallel machines are described in e.g. Clark and Clark (2000), Kang *et al.* (1999), Kuhn and Quadt (2002) and Meyr (2002). **The planning of a canning line at a drink manufacturer is presented by Clark (2003) where the author proposes various heuristic approaches.**

A first mixed-integer mathematical model for the SITLSP is described in Toledo (2005) and Toledo *et al.* (2006). The underlying idea to create a model for the SITLSP combines issues from the General Lot-sizing and Scheduling Problem (GLSP) and the Continuous Setup Lot-sizing Problem (CSLP). In the SITLSP model, a planning horizon is divided into T (macro-) periods of the same length. A maximum number of slots (S for each line and $\overline{S}$ for each tank) is fixed for each macro-period $t = t_1, t_2, ..., t_T$. **The limited number of slot assignments is an idea taken from the GLSP models presented in Fleischmann and Meyr (1997), Meyr (2000) and Meyr (2002).** This enables us to determine in the SITLSP for which product (raw material) a particular slot in a particular line (tank) is reserved, and which lot size (a lot of size zero is possible) should be scheduled. As well as the assignment of raw material to tanks and products to lines in each macro-period, it is also necessary to synchronize the slots scheduled in a two-level problem like this. **This is done by introducing the concept of micro-period, an idea found in the CSLP model (Bitran and Matsuo 1986, Drexl and Kimms 1997), where each macro-period t is divided into T^m micro-periods with the same length.** According to the CSLP assumptions, the capacity of each micro-period can be **totally** or partially occupied and only one product type can be produced per micro-period.

An example helps to understand these ideas. Consider T=2 macro-periods, 5 raw materials (*RmA*, *RmB*, *RmC*, *RmD* and *RmE*), 6 products (*P1*, *P2*,..., *P6*), 3 tanks (*Tk1*, *Tk2* and *Tk3*) and 3 lines (*L1*, *L2* and *L3*). Suppose that the raw material *RmA* produces product *P1*, *RmB* produces *P2* and *P3*, *RmC* produces *P4*, *RmD* produces *P5* and *RmE* produces *P6*. The total number of slots is $S=\overline{S}=2$ and it can not be exceeded in each macro-period. The same happens with the total number of micro-periods T^m=5. Figure 2 shows how the slot assignments and micro-periods can help to synchronize and integrate the two-levels of the problem.

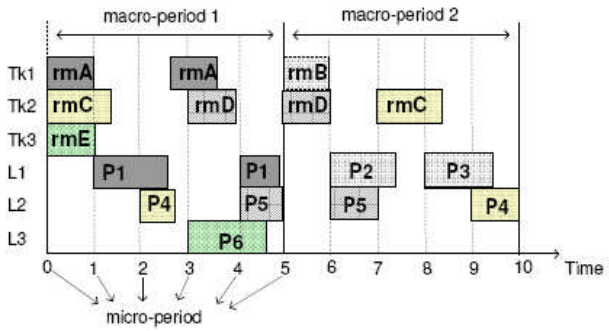


Fig. 2. Synchronization between lines and tank slots

The method proposed in this paper to solve the SITLSP is a multi-population genetic algorithm where each population is conceived as a hierarchical ternary tree structure. The Genetic Algorithm (GA) is a very popular search procedure inspired by biological evolution processes (Goldberg 1989, Holland 1975, Michalewicz 1996). There is vast literature concerning the use of GAs to solve lot-sizing and scheduling problems. A lot-sizing and scheduling problem with sequence-dependent setup times is solved using GA by Sikora (1996), where an individual is represented as a string of paired values (type of product and lot size) in each scheduling period. The capacitated lot-sizing and loading problem with setup times, parallel facilities and overtime, proposed by Özdamar and Birbil (1998), uses another direct representation of solutions for individuals in GA. An individual is also a string of paired values, where the first value is the lot size and the second is the facility.

A more elaborate representation of a solution is proposed in Dellaert *et al.* (2000), where a binary matrix *PxT* (*P* products and *T* periods) represents an individual for the multi-level lot-sizing problem. Each binary entry $y_{i,t}$ is set to 1 if a setup for product *i* occurs in *t*; otherwise $y_{i,t}$=0. Another elaborate representation is given by Dorndorf and Pesch (1995) where the genes in the GA are rules to select

tasks to be scheduled in a job shop scheduling problem. All the GA papers mentioned above have specific genetic operators (crossover, mutation, and selection) designed to deal with these individual representations (solutions) for lot-sizing and scheduling problems. A multi-population GA was developed in this paper because populations that evolve separately will have different characteristics according to the genetic drift idea (Weiner 1995). This can allow for a more effective exploration in the solution space of the problem. As reported by Mendes (2003) better results solving different optimization problems using a structured multi-population GA instead of a non-structured single population GA were attained. In addition, França *et al.* (2001) found better results solving the total tardiness single machine scheduling problem using GA with hierarchically structured populations.

This paper is organized as follows. In Section 2, the GA approach with the individual representation and its decoding procedure are fully described, as well as the tailor-made genetic operators and the multi-population structure. Computational results using instances, whose data were provided by a soft drink company, are reported in Section 3. The conclusion follows in Section 4.

2. Genetic Algorithm

In this section a multi-population hierarchical GA procedure is developed. In França *et al.* (2001), Mendes *et al.* (2001) and Mendes (2003) similar and well succeeded approaches are reported. These findings have also been observed in the present work with the SITLSP. Moreover, the adoption of the multi-population approach has enhanced the convergence features of the GA, postponing premature convergence and improving its whole effectiveness (Toledo 2005).

The GA proposed here was implemented using the NP-Opt (Mendes *et al.* 2001, Mendes 2003) an object-oriented framework written in JAVA (Java Sun 2007) code which contains procedures based on evolutionary computation techniques to address NP-hard problems. The next pseudo-code (Figure 3) describes the multi-population GA available in the NP-Opt.

```
Method MultiPopulationGeneticAlg
begin
 repeat
  for i = 1 to numberOfPopulations do
    initializePopulation(pop( i ));
    evaluatePopulationFitness(pop( i ));
    structurePopulation(pop( i ));
    repeat
     for j = 1 to numberOfCrossover do
       selectParents(individualA,individualB);
       newInd=crossover(individualA, individualB);
       if (executeMutation newInd) then
            newInd=mutation(newInd);
       evaluateFitnessIndividual(newInd);
       insertPopulation(newInd, pop(i ));
     end
     structurePopulation(pop( i ));
    until(populationConvergence pop(i ));
  end
  for i = 1 to numberOfPopulations do
    executeMigration pop(i );
  end
 until(stop criterion)
end
```

Fig.3. Pseudo-code for the Multi-Population Genetic Algorithm.

The algorithm executes a fixed number of crossovers in each population while there is no convergence. This process involves a parent selection (**selectParents**(individualA, individualB)), a new individual creation by crossover (**crossover**(individualA, individualB)), a possible mutation application to the new individual (**mutation**(newInd)), its fitness evaluation (**evaluateFitnessIndividual**(newInd)) and its insertion or not into the population (**insertPopulation**(newInd, pop(i)). The population convergence occurs when there are no new individuals inserted after a fixed number of crossovers has taken place. A migration among populations (**executeMigration** pop(i)) is executed when all populations have converged and the stop criterion has not been satisfied yet. In case a new initialization of the populations (**initializePopulation**(pop(i))) occurs, the best individual and the migrated individuals are kept. Details on the constitution of populations and the migration policy will be presented later.

2.1 Individual

Two interdependent lot-sizing and scheduling problems have to be solved by the GA approach. An elaborate coding of solutions is proposed in this work. This representation is close to the one presented in Kimms (1999) which uses assignment rules in a multi-level proportional lot-sizing and scheduling problem with multiple machines. An individual in Kimms (1999) is a matrix with *M* lines and *T* columns and each matrix entry (gene) is a rule $\theta_{m,t}$ used to select the setup state for machine *m* at the end of period *t*. Figure 4 introduces the individual representation proposed for the SITLSP.

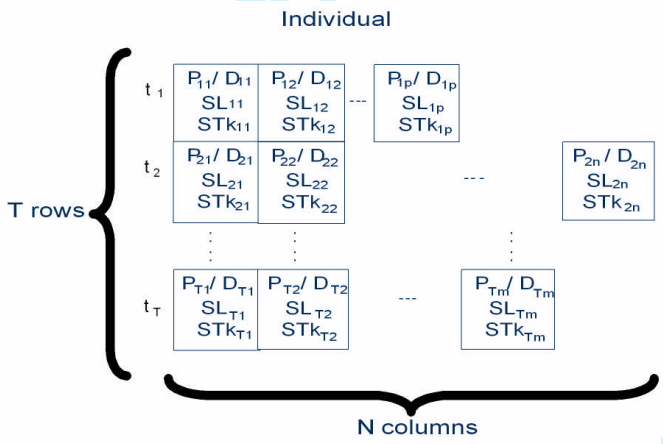


Fig. 4. Individual representation

The rows represent macro-periods t_1, t_2,...,t_T. The column represents the genes and there can be a different number of genes per macro-period. Each gene corresponds to a cell in the individual representation that contains the following data:

- P_{mn}: product in gene n to be produced in macro-period m.
- D_{mn}: lot size of product P_{mn}.
- SL_{mn}: sequence of lines where D_{mn} can be produced.
- STk_{mn}: sequence of tanks where the raw material of D_{mn} can be stored.

The demand d_{it} of product i in period t is divided into many lots (D_{mn}) and randomly distributed among the genes in periods t, t-1, t-2, ..., 1. The sequences SL_{mn} and STk_{mn} are randomly generated with length k. Sequence SL_{mn} is defined below, where L is the number of lines:

$$SL_{mn}=(\alpha_1,...,\alpha_k) \text{ with } \alpha_i \in \{1,...,L\},$$

α_i is the line number in position i of the sequence. Sequence STk_{mn} is defined below, where $\bar{L}$ is the number of tanks:

$$STk_{mn}=(\beta_1,..., \beta_k) \text{ with } \beta_i \in \{1,2,...,2\bar{L}\},$$

β_i tells where and how the raw material will be stored. The β_i is taken from $2\bar{L}$ possible values and the reasons to do this will be explained next. The tank number j where the raw material will be stored is obtained from β_i doing:

$$j=\begin{cases}\beta_i & 1\le \beta_i \le \bar{L} \\ \beta_i-\bar{L} & \bar{L}< \beta_i \le 2\bar{L}\end{cases} \quad (1)$$

Let Rm_a be the raw material of lot size D_{mn}. The assignment of Rm_a to tank j obeys the following criteria:

- **Criterion 1**: If tank j stores lots of raw materials Rm_b and $Rm_a \neq Rm_a$, the Rm_a lot will occupy the next assignment to j. Two different raw materials cannot be stored at the same time in j.
- **Criterion 2**: If tank j is full, Rm_a will occupy the next assignment to j.
- **Criterion 3**: If tank j is empty, it will be immediately occupied by Rm_a.
- **Criterion 4**: If tank j is already occupied by other lots of Rm_a and the minimum tank capacity is not filled yet, the new Rm_a lot has to be assigned to j.
- **Criterion 5**: If $1\le \beta_i\le \bar{L}$, Rm_a occupies a new assignment to tank $j = \beta_i$, as the previous conditions have been regarded.
- **Criterion 6**: If $\bar{L} < \beta_i\le 2\bar{L}$, Rm_a integrates the previous lot already assigned to tank $j = \beta_i-\bar{L}$, as the previous conditions have been regarded.

Criteria 1 to 4 ensure that the problem constraints (such as different raw materials in the same tank, minimum tank capacity, etc) are not violated. They have priority over criteria 5 and 6 which allow for solutions with a partial (criterion 5) or total (criterion 6) tank occupation. The individuals in each initial population are generated following the pseudo-code next (Figure 5). There are T macro-periods, J products, L lines and $\bar{L}$ tanks.

IndividualInitializationAlg.
For $t=t_1, t_2,..., t_T$ **do**
 repeat
  Select a product $P_i \in \{P_1, P_2,...,P_J\}$randomly with $d_{Pi,t} > 0$.
  Set Dem = $d_{Pi,t}$
 while Dem>0 **do**
  Select the matrix row $m \in \{ t_1, t_2,...,t\}$ of the individual randomly.
  Determine n as the first gene available in line m of the individual.
  Determine $D_{mn} \neq 0$ with $D_{mn} \in [0, \text{Dem}]$ randomly generated.
  Insert D_{mn} and P_i in the gene position (m,n) of the individual.
  Generate $SL_{mn}= (\alpha_1,...,\alpha_k)$ with $\alpha_i \in \{1,...,L\}$ randomly selected.
  Generate $STk_{mn}= (\beta_1,..., \beta_k)$ with $\beta_i \in \{1,2,...,2\bar{L}\}$ randomly selected.
  Set Dem = Dem - D_{mn};
 end
 until All demands have been distributed among the genes.
end

Fig. 5. Individual initialization algorithm

Individuals which are randomly generated make a certain level of diversity to the initial population possible. First, the variable *Dem* receives the total demand $d_{Pi,t}$ of product P_i in the macro-period t. This demand is randomly divided and distributed among the genes. Of course, each piece of demand is not positioned after its demand's due date (macro-period t). The following example clarifies the solution representation as an individual. Suppose two products (P_1 and P_2) where each product has a demand of 100 units to be filled in macro-period t_1 and another 100 units to be filled in macro-period t_2. The products use different raw materials. Moreover, there are two lines available to produce both products and two tanks available to store both raw materials. Figure 6 gives us two possible representations, both based on the individual initialization algorithm.

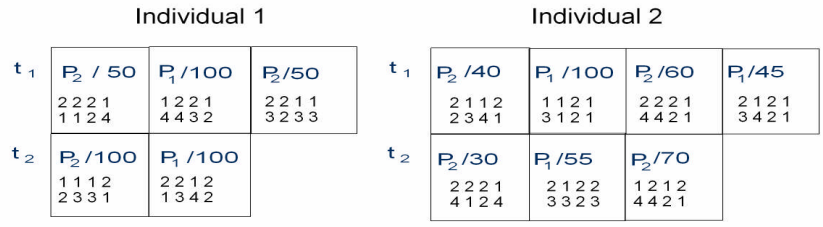


Fig. 6. Two possible individuals

The demands are distributed in their respective macro-periods in individual 1. However, the demand of P_2 in t_1 is split between two genes. In individual 2, part of the P_1 demand in t_2 is produced in t_1. Notice that the sequence of lines and tanks can repeat values of $\alpha_i \in \{1,2\}$ and $\beta_i \in \{1,2,3,4\}$ for $L = \overline{L} = 2$ and length $k = 4$ in the sequences SL_{mn} and STk_{mn}.

2.2 Decoding and fitness

The decoding procedure is responsible for determining a solution from the data encoded in each gene of an individual. The procedure starts from the first gene in the last macro-period up to the last gene in the first macro-period. This backward procedure allows us to postpone setups and processing time of products and raw materials in lines and tanks. However, there is no guarantee that all demands will be produced at the end and a penalty in the fitness is taken into account for that. An example illustrates the decoding procedure. Consider the same data used in the example of section 2.1 and individual 2 shown there. The process begins by the first gene in the last macro-period. A lot of product P_2 ($D_{21} = 30$) has to be produced using the first pair (α_1, β_1) = (2, 4) from sequences SL_{21} and STk_{21} (Figure 7). Product P_2 is produced in line 2 because $\alpha_1 = 2$ and let's assume that its processing time takes 1 micro-period. The other processing times used in this example are suppositions as well. According to equation (1) of section 2.1, raw material *Rm2* of P_2 has to be stored in tank $j = 4\text{-}2 = 2$ because $\beta_1 = 4$ and $2 < \beta_1 \leq 4$ with $\overline{L} = 2$. Given the criteria defined in section 2.1, criterion 3 has priority over the others in this case because tank $j = 2$ is empty and must be occupied immediately. A tank should be ready at least one micro-period before the production starts in lines. Therefore, the setup time of *Rm2* occurs in Figure 7 one micro-period before the P_2 production starts in *L2*. Let us suppose that the setup time of raw materials will take 1 micro-period in any tank in this example.

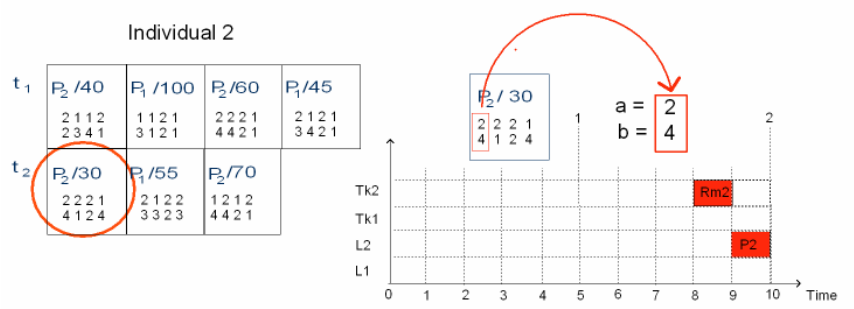


Fig. 7. Decoding of the first gene in t_2.

The demand of this gene has been completely scheduled, so the next gene in t_2 is decoded (Figure 8). The first pair of rules (α_1, β_1) = (2,3) is selected to schedule 55 units of P_1 which are also produced in *L2* (α_1 = 2). There is a setup time because P_2 is scheduled next. Value β_1 = 3 means j = 3-2 = 1 ($2<\beta_1 \leq 4$) by equation (1). Tank j = 1 is empty and, according to criterion 3, must be occupied immediately by *Rm1* of P_1. Figure 9 presents the third gene decoding in t_2. The lot size of P_2 has to be produced in *L1* (α_1 =1) and its raw material has to be stored in tank j = 2 (β = 4 implies j = 4-2 = 2). There is a lot of *Rm2* in j = 2 (criterion 1 does not apply) and we are supposing that there is available capacity in this tank (criterion 2 does not apply as well). This tank is not empty and, according to criterion 6, the *Rm2* must integrate the lot of *Rm2* already stored in j=2. The insertion of the new lot in j = 2 causes a setup time anticipation. The tank now has to be ready one micro-period before the P_2 production starts in *L1*.

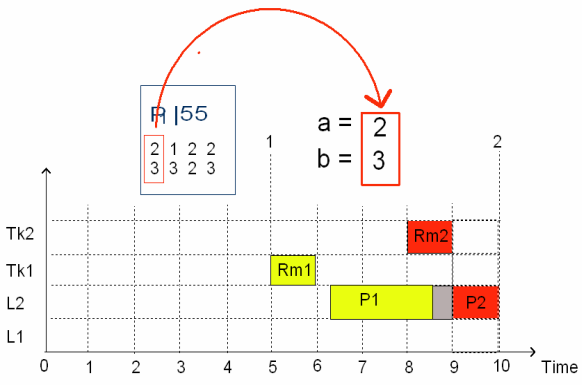


Fig. 8. Decoding of the second gene in t_2

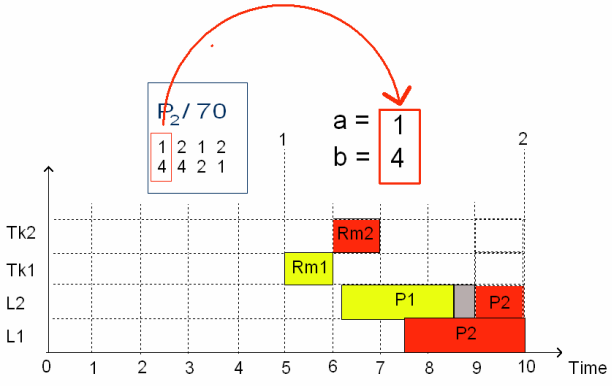


Fig. 9. Decoding of the third gene in t_2.

The decoding process continues in the first gene in t_1 (Figure 10). Product P_2 is produced in *L2* (α_1 = 2) and ***Rm2* is assigned to tank j = 2 (β_1 =2)**. Let's suppose that the minimum tank capacity has already been filled. In this case, criterion 4 does not apply as well as the criteria 1 to 3. Therefore, criterion 5 can be applied (β_1 =2 with $1<\beta_1 \leq 2$), and a new lot is scheduled for tank j = 2. The next gene decoding is shown in Figure 11. Line *L1* (α_1 =1) is selected to produce D_{12} = 100 units of P_1. Raw material *Rm1* is assigned to j = 4-3 =1 (β_1 = 3) and it must occupy the previous lot already assigned to this tank (criterion 6). However, we are supposing at this point that there is available capacity in tank j = 1 to store raw material sufficient to produce only 45 units of P_1. In this case, the setup time of *Rm1* is anticipated for the third micro-period and part of D_{12}=100 is scheduled in line *L1*. Thus, the next pair (α_2, β_2) is selected to schedule the remaining lot D_{12} = 100-45 = 55. The remaining lot is produced in *L1* (α_2=1) and a new lot of *Rm1* must be used in tank j=1 (β_2 =1) by criterion 5.

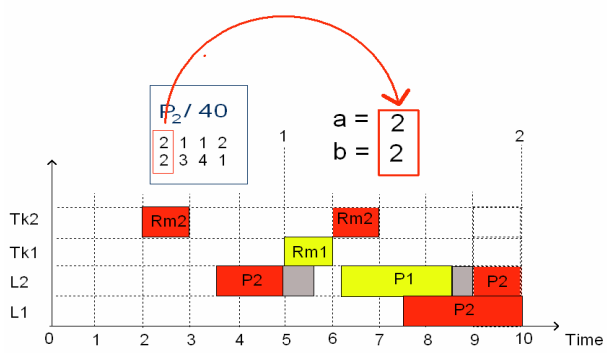
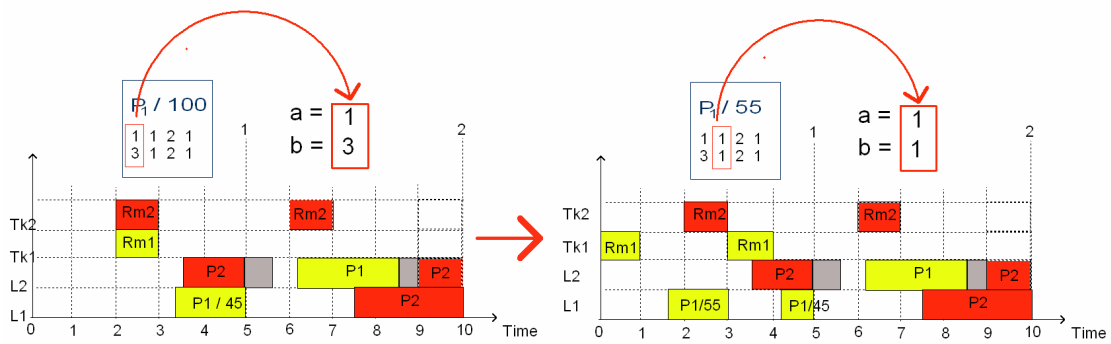


Fig. 10. Decoding of the first gene in t_1.



.Fig. 11. Decoding of the second gene in in t_1.

The next gene decoding is illustrated in Figure 12. Line *L2* produces P_2 (α_1 =2) and there is no setup time because the same product is scheduled next. Raw material *Rm2* is integrated into the lot already stored in tank j = 4-2 = 2 (β_1 = 4), so there is anticipation in the setup time of this tank. Figure 13 has the last gene decoding. Product P_1 is scheduled in *L2* (α_1 = 2) and we have a setup time because P_2 is produced next. *Rm2* will integrate the lot previously stored in tank j = 3-2 = 1 because β_1 = 3 and the conditions of criterion 6 are satisfied.

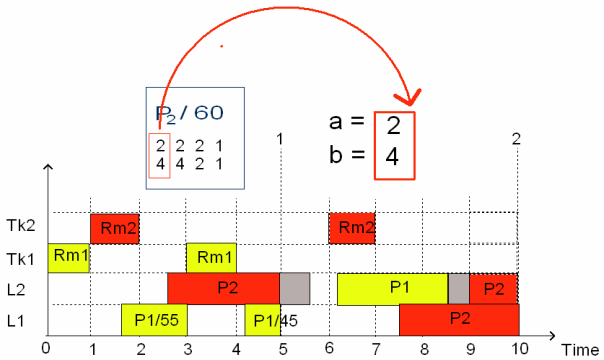


Fig. 12. Decoding of the third gene in t_1.

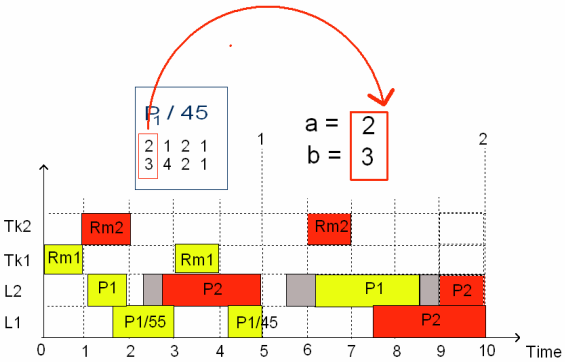


Fig. 13. Decoding of the last gene in t_1.

The final schedule established by the decoding procedure for lines and tanks is a possible solution for the problem which is evaluated using the fitness function. The objective function of the mathematical formulation described in Toledo (2005) and Toledo *et al.* (2006) is the fitness function in the GA. To sum up, this value is determined adding up all costs that happen in the final schedule: setup, production

and inventory costs for products and raw materials in lines and tanks, respectively. If some demands are not satisfied, a high penalty cost per unit also occurs in the objective function. The best individual of the population is the one whose decoding process determines a final schedule with a minimum cost value.

2.3 Crossover and mutation

Previous computational experiments with several crossover operators revealed that the best behavior was attained by the uniform crossover. In this recombination operator, genes of two different parents that occupy the same position in the individuals have some probability of being inherited by the child. Individuals 1 and 2 (Figure 6) are used to show how the uniform crossover operator works. Sequences SL_{mn} and STk_{mn} are not relevant because the new individual (Child) will inherit these sequences without changes. Figure 14 illustrates the crossover of Ind1 and Ind2. For each gene in the same position in Ind1 and Ind2, a random value $\lambda \in [0,1]$ is generated. If $\lambda < 0.5$, Child inherits the gene of Ind1; otherwise, Child inherits the gene of Ind2. The genes selected following this procedure are shaded in Figure 14. Notice that there are more genes in macro-periods t_1 and t_2 of Ind2 than in the same macro-periods of Ind1. In this case, the procedure continues in Ind2 selecting those genes where $\lambda \geq 0.5$. We do not allow excessive demands in a new individual. For example, the gene of Ind1 marked by a circle in Figure 14 is not inherited because it would exceed the total demand of P_1 in Child. If there is a gene in the same position in Ind2, this gene must be inherited by Child if the same problem does not occur. On the other hand, a lack of demand can occur at the end of the crossover. For this reason, a repair procedure is necessary and the demand deficits in some macro-period are inserted (Figure 15).

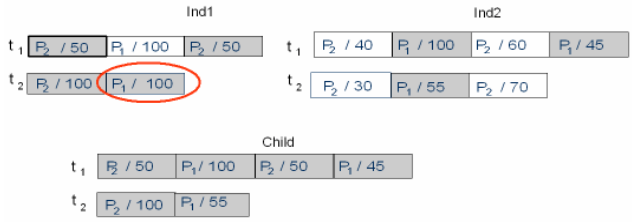


Fig. 14. Uniform crossover example.

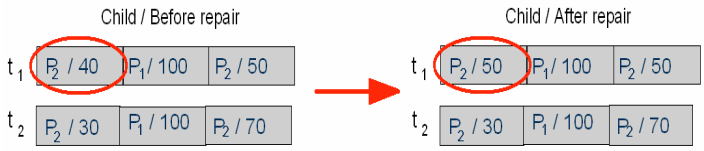


Fig. 15. Repair in Child.

Mutation aims to keep diversity in a population avoiding premature convergence. A mutation rate determines the number of individuals that will be changed. The mutations adopted in this paper basically swap gene positions. Figure 16 shows an example of the three mutation operators used. The first type swaps the positions of two selected genes in the same macro-period. In the second type, the selected gene is removed and inserted into another position which is also randomly selected. The third type swaps the positions of two chosen genes that are in different macro-periods. The new gene positions have to respect the macro-period demand of each product. A product swap will not take place if it can violate the demand satisfaction. The mutation procedure randomly chooses which mutation type will be applied to the individual.

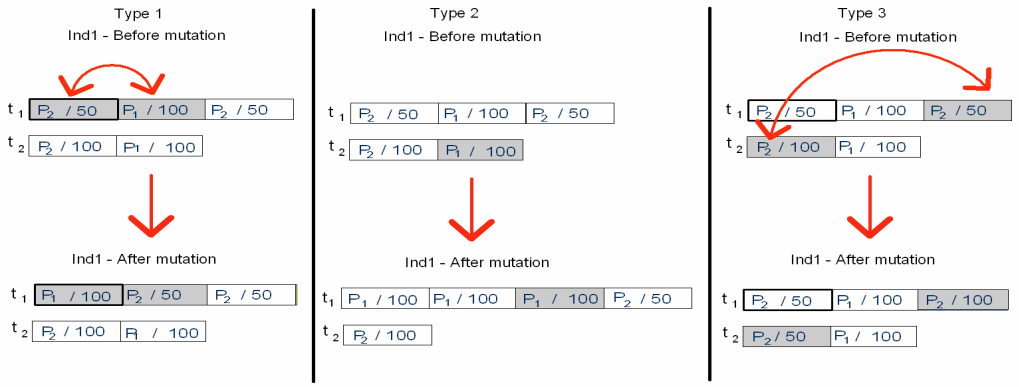


Fig. 16. Mutation types.

2.4 Populations and migration

A multi-population genetic algorithm is proposed here where each population is constituted by a hierarchical two-level ternary tree structure. Each structure is formed by 4 clusters of 4 individuals and arranged in two levels (1 cluster in level 1 and 3 clusters in level 2). The cluster has a node leader and 3 other nodes called supporters (Figure 17).

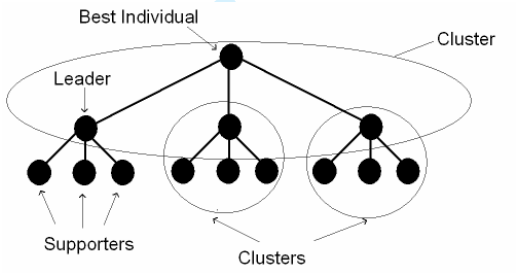


Fig. 17. A population as a hierarchical ternary tree structure.

Note that the leader of the whole population is called the best individual. In the algorithm MultiPopulationGeneticAlg explained in the beginning of this section, the parameter *numberOfCrossover* is used to determine the maximum number of recombination executed in each population. This parameter is provided by the equation:

$$numberOfCrossover = \rho * PopulationSize \quad (2)$$

where ρ is the crossover rate. A crossover is carried out over a cluster (selected at random) and always selects a supporter node as parents, also randomly chosen, and its respective leader node. If better than some parents, the new individual (Child) will replace the parent with the worst fitness value (Figure 18). Otherwise, the new individual is not inserted into this population. After every crossover, adjustments are necessary to keep the cluster structure well ordered where the best is the leader (Figure 19).

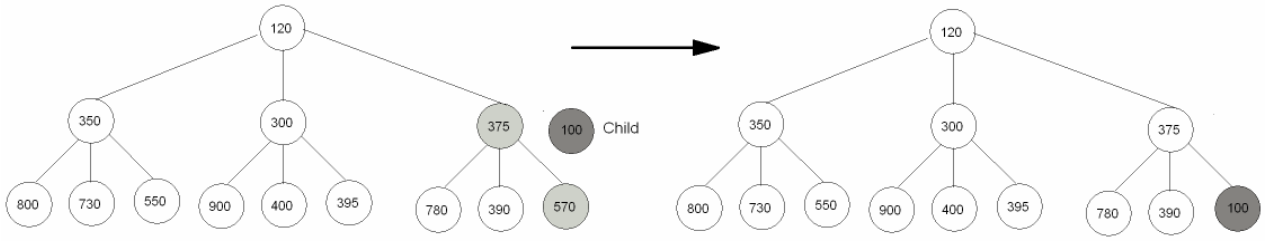


Fig. 18. Population before and after Child insertion.

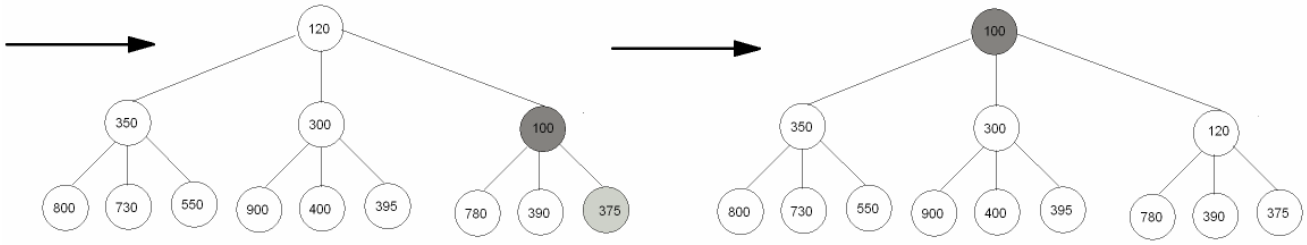


Fig. 19. Adjustments in the population.

The populations converge when no new individuals are inserted in any population after the *numberOfCrossover* executions are reached. In this case, a copy of each best individual is inserted into the next population replacing some individual randomly selected - except the best one. This is the migration process and only one new individual is inserted into each population. New initialization of the population occurs after the migration process. However, the best individuals and the individuals that have just migrated are kept (Figure 20).

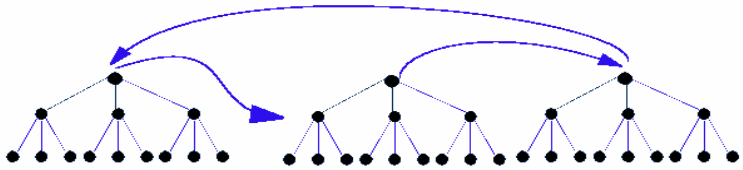


Fig. 20. Migration between populations.

3. Computational Results

In this section, computational results for a set of instances are presented. These instances are based on data provided by a soft drink company.

3.1 Computational results: small-to-moderate size instances

The parameters adopted to create small-to-moderate size instances are in Table 1. Some parameter values are fixed and others are chosen from a set of values.

Table 1. Parameter values and meanings

Param.	Values	Meaning	Param.	Values	Meaning
L	{2;3;4}	Number of lines	h_j	1($/$u$)	Unity inventory cost of product j
$\overline{L}$	{2;3}	Number of tanks	$\underline{h}_j$	1($/$u$)	Unity inventory cost of raw material j
J	{2;3;4}	Number of products	v_{jl}	1($/$u$)	Unity production cost of product j in

					line l
$\overline{J}$	{1;2}	Number of raw materials	$\bar{v}_{jk}$	1($/$u$)	Unity production cost of raw material j in tank k
T	{1;2;3; 4}	Number of macro-periods	Q_k	5000l	Maximum capacity of tank k (liters)
C	5	Capacity of each macro-period	Q^k	1000l	Minimum capacity of tank k (liters)

A certain combination of parameters L, $\overline{L}$, J, $\overline{J}$ defines a set of instances. Table 2 presents the values used in each combination.**These particular parameter combinations were chosen because most of the variables and constraints in the mathematical model (see Toledo 2005, Toledo *et al.* 2006) are indexed by these parameters. Increments in these parameter values may result in the exact approach to failing in solving the instances.**

Table 2. Combination of parameters per macro-period for small-to-moderate sized instances

Comb.	$L/\overline{L}/J/\overline{J}$	Comb.	$L/\overline{L}/J/\overline{J}$	Comb.	$L/\overline{L}/J/\overline{J}$
1	2/2/2/1	4	3/3/2/1	7	4/4/2/1
2	2/2/3/2	5	3/3/3/2	8	4/4/3/2
3	2/2/4/2	6	3/3/4/2	9	4/4/4/2

A total of 9 parameter combinations per macro-period (T={1,2,3,4}) were chosen, limiting the simulations to 4x9=36 possible set of instances. **These sets are considered of small-to-moderate size if compared to the industrial instances found in soft drink companies and also in terms of the number of variables and constraints in the related mathematical model (see Table 4)**. In each set of instances, 10 replications are randomly generated resulting in 4x9x10 = 360 instances in total. The other parameters used to create instances are shown in Table 3.

Table 3. Parameter ranges

Param.	Ranges	Meaning
σ_{ijl}	U[0.5; 1]	Setup time (hours) from product i to j in line l
$\overline{\sigma}_{ijk}$	U[1; 2]	Setup time (hours) from raw material i to j in tank k
p_{jl}	U[1000; 2000]	Processing time (units/hour) of product i in line l
d_{jt}	U[500; 10000]	Demand (units) of product j in period t.
r_{ji}	U[0.3; 3]	Conversion factor

The conversion factor r_{ji} determines how many liters of raw material i are necessary to produce one unit of product j. These parameters are generated from a uniform distribution in the interval [a,b]. The setup costs in lines and tanks are calculated as:

$$s_{ijk} = f\,\sigma_{ijk} \quad (3)$$

$$\overline{s}_{ijk} = f\,\overline{\sigma}_{ijk} \quad (4)$$

respectively, where s_{ijl} is the setup cost when one shifts from product i to j in line l and $\overline{s}_{ijk}$ is the setup cost when one shifts from raw material i to j in tank k. These setup costs are proportional to the setup times as determined by parameter f in equations (3) and (4). The same approach is used in Haase and Kimms (2000), where a similar procedure establishes setup costs from setup times in a lot-sizing and scheduling problem for a single-stage and single-machine production system. According to the results reported in Toledo (2005) and Toledo *et al.* (2006), the most adequate value for f is 1000 given that it provides a suitable trade off between the different terms of the objective function.

The mathematical model developed in Toledo (2005) and Toledo *et al.* (2006), and the instances generated wherein were coded using the modelling language GAMS IDE, version 2.0.10.0, and solved by the solver

CPLEX, version 7 (Gams 2007). The CPLEX handles mixed integer problems using a branch & cut algorithm which solves a series of LP subproblems (Hoffman and Padberg 1991). GAMS/CPLEX ran on each instance only once during the time limit of 1 hour. Two kinds of problem solutions can be returned: the optimal solution or the best feasible solution achieved up to the time limit.

The other series of experiments involving the small-to-moderate size instance set was to find heuristic solutions using the multi-Population genetic algorithm (GA). In the combinations where the GAMS/CPLEX spent up to 0.5 hour (CPU time) on average to find the optimal solution (see Table 4), the GA ran during 360 seconds and this execution was repeated 5 times over each instance. Taking this into account, the GA had a total execution time of 0.5 hour per instance. In the combinations where GAMS/CPLEX spent more than 0.5 hour on average to return a solution (see Table 6), the GA ran on each instance during 1200 seconds. This GA execution was repeated 3 times over each instance, that is, a total execution time of 1 hour per instance. The GA consists of 3 populations structured in ternary trees of 13 individuals each, which means a total of 39 individuals. The crossover rate ρ was set to 1.5 which leads the procedure to execute 19 crossovers over each population. The mutation rate is fixed at 0.7. A number γ is randomly chosen in the interval [0,1] with uniform distribution. If $\gamma < 0.7$, the mutation operator is executed on the new individual.

All the computational tests were executed on a Pentium IV microcomputer with 2.8 GHz. For each combination of parameters (Comb.), Table 4 presents the number of constraints (Const.) and binary (Bin.) and continuous (Cont.) variables of the mathematical model generated by GAMS/CPLEX with T=1,2. The number of optimal solutions (Opt.) returned by the solver and the average CPU time (seconds) are presented as well.

Table 4. GAMS/CPLEX model sizes and optimal solutions for the sets of instances with T=1,2

Comb.	T=1					T=2				
	Bin.	Cont.	Const.	Opt.	CPU	Bin.	Cont.	Const.	Opt.	CPU
1	100	263	281	10	2	210	533	575	10	28
2	136	499	454	10	1	282	1004	919	6	1703
3	142	615	512	10	1	294	1235	1039	6	1545
4	150	452	419	10	573	315	916	862	1	3366
5	204	880	673	10	152	423	1771	1368	0	3600
6	213	1098	764	10	46	441	2206	1552	1	3260
7	176	555	498	2	3069	367	1122	1013	0	3600
8	246	1100	821	8	989	507	2211	1641	1	3255
9	258	1390	924	7	1357	531	2790	1865	0	3600

The GAMS/CPLEX solved most of the problem instances with T = 1 (77 out of 90 instances) optimally. However, GAMS/CPLEX had problems to solve instances with T=2 optimally within the time limit (25 out of 90 instances). This was expected, because of the increasing number of constraints and variables of the model for each instance with T = 1 and T = 2. The following relative deviation was used to compare the GA and GAMS/CPLEX solutions:

$$Dev(\%) = 100\left(\frac{Z - \overline{Z}}{\overline{Z}}\right)$$

where, Z is the final solution returned by GA and $\overline{Z}$ is the final solution returned by GAMS/CPLEX. The average (Avg), maximum (Max) and minimum(Min) deviation values are listed for T=1,2 in Table 5. The Avg determines an average deviation of the final GA solution from the solution returned by GAMS/CPLEX. This average considers the final solutions of all (three or five) GA ran on each instance. Max is the same average

deviation, but it considers only the worse solution found by GA in its runs in each instance. Min is the average deviation considering only the best solution found by GA, after the executions in each instance.

Table 5. GA deviation values from GAMS/CPLEX solutions for T=1,2

Comb.	T=1 – Dev(%)			T=2 – Dev(%)		
	Avg.	Min.	Max.	Avg.	Min.	Max.
1	0.00	0.00	0.00	0.27	0.23	0.31
2	0.00	0.00	0.00	0.18	0.11	0.28
3	0.34	0.34	0.34	0.65	0.38	0.83
4	0.00	0.00	0.00	0.79	0.66	1.03
5	0.00	0.00	0.00	-0.11	-0.27	0.16
6	0.00	0.00	0.00	0.04	-0.08	0.10
7	0.00	0.00	0.00	1.30	1.01	1.72
8	0.00	0.00	0.00	1.84	0.86	2.53
9	0.00	0.00	0.00	-1.86	-1.88	-1.82

The GA was able to solve most of the problem instances with T=1 optimally. For T = 2, notice that some of the deviation values are negative, meaning that the GA was able to find better solutions than the (non-optimal) solutions returned by GAMS/CPLEX. In the sets where GA did not outperform GAMS/CPLEX, its average deviations were less than 1% in most of the cases. Table 6 shows the model dimensions, the number of optimal solutions and the average CPU time for the instance sets with T=3,4.

Table 6. GAMS/CPLEX model sizes and optimal solutions for the sets of instances with T=3,4

Comb.	T=3					T=4				
	Bin.	Cont.	Const.	Opt.	CPU	Bin.	Cont.	Const.	Opt.	CPU
1	320	803	863	2	3147	430	1073	1163	1	3377
2	574	2014	1835	3	3077	574	2014	1827	1	3245
3	598	2475	2075	3	2912	598	2475	2075	0	3600
4	480	1380	1303	0	3600	645	1844	1752	0	3600
5	642	2662	2071	0	3600	861	3553	2764	0	3600
6	669	3314	2350	0	3600	897	4422	3140	0	3600
7	558	1689	1536	1	3281	749	2256	2039	0	3600
8	768	3322	2466	0	3600	1029	4433	3335	0	3600
9	804	4190	2799	0	3600	1077	5590	3735	0	3600

GAMS/CPLEX was unable to find optimal solutions in most of these problem instances within 1 hour. The GA deviations regarding GAMS/CPLEX final solutions for T=3,4 are shown in Table 7. GA was able to find better final solutions than the best feasible solutions provided by GAMS/CPLEX in many sets of instances. In the other instance sets, the GA deviation was less than 5% in most of the cases.

Table 7. GA deviation values from GAMS/CPLEX solution for T=3,4

Comb.	T=3 – Dev(%)			T=4 – Dev(%)		
	Avg.	Min.	Max.	Avg.	Min.	Max.
1	0.39	0.37	0.43	1.77	1.32	2.27
2	-0.40	-0.43	-0.39	0.10	0.03	0.13
3	-0.73	-0.83	-0.53	-1.63	-1.64	-1.62
4	2.26	1.52	2.95	2.61	1.68	3.71
5	-0.96	-1.27	-0.48	-2.41	-3.34	-1.51
6	0.06	-0.74	0.70	-2.56	-3.34	-1.75

7	5.01	3.90	6.13	8.12	6.57	9.50
8	-3.62	-4.14	-3.01	-2.22	-3.31	-1.13
9	-0.56	-0.81	-0.34	-3.59	-4.54	-2.63

The evolutionary approach reveals itself to be a good alternative to solve more complex instances. Considering all instances and all average results (Avg, Min and Max), the conclusion is that in 103 out of 108 results, the GA deviations differ less than 5% from GAMS/CPLEX solutions. Similarly, in 37 out of 108 average results, the GA outperforms GAMS/CPLEX best feasible solutions (31 out of 54 for T=3,4 and 6 out of 54 for T=1,2).

3.2 Computational results: industrial instances

This section reports computational results that were obtained solving some industrial instances. These instances were provided by a Brazilian soft drink manufacturer whose production planning problem motivated this work. The data of instances are based on the schedules executed by the company in their lines and tanks during different time periods. The parameters which define each instances are presented in Table 8.

Table 8. Parameters of industrial instances

Comb.	$L/\bar{L}/J/\bar{J}/T$	Comb.	$L/\bar{L}/J/\bar{J}/T$
A1	5/9/33/11/1	B1	6/10/52/19/1
A2	6/9/49/14/2	B2	6/10/56/19/2
A3	6/9/58/15/3	B3	6/10/65/21/3

Notice that each combination represents now only one instance. The demands must be met at the end of each period. In the A instances, each planning covers 7 days with 24 hours of available capacity per day. In the B instances, each period covers 10 days with 24 hours of available capacity per day. Table 9 presents the setup values.

Table 9. Setup values

Meaning	Value	Meaning	Value
Line setup time (hours)	$\sigma_{ijl} = \begin{cases} 0.5 & i \neq j \\ 0.0 & i = j \end{cases}$	Line setup cost ($/u)	$s_{ijl} = \begin{cases} 3000 & i \neq j \\ 0.0 & i = j \end{cases}$
Tank setup time (hours	$\bar{\sigma}_{ijl} = \begin{cases} 1 & i \neq j \\ 0.5 & i = j \end{cases}$	Tank setup cost ($/u)	$\bar{s}_{ijl} = \begin{cases} 12000 & i \neq j \\ 6000.0 & i = j \end{cases}$

The inventory cost of products (h_j), the production costs for products (v_{jl}) and raw materials ($\bar{v}_{jk}$) keep the values as used in the small-to-moderate sized instances. There is no inventory cost of raw material ($\bar{h}_j$=0). All these costs values were used to evaluate the production plan elaborated by the company. The value achieved is the cost of the company schedule (Z^I) in Table 10. The GA ran 3 times over each instance with 3600 seconds assigned to each run. The GA parameters are the same used to solve the small-to-moderate sized instances. Table 10 compares the cost (in 10^6 monetery units) of the company schedule (Z^I) with the better GA final solution (GA-Min), the average GA solution (GA-Avg), and the worst GA final solution (GA-Max) returned after three executions. The corresponding relative deviations of the GA solutions from Z^I are also shown.

Table 10. GA solutions compared to solutions provided by a manufacturer

Comb.	Solutions				Deviation – Dev(%)		
	Z^I	GA–Min	GA-Avg	GA-Max	GA-Min	GA-Avg	GA-Max
A1	1692098	1662098	1667098	1671098	-1.8	-1.5	-1.2
A2	3511910	3381357	3388059	3398510	-3.7	-3.5	-3.2
A3	5002677	4837433	4847207	4859924	-3.3	-3.1	-2.9
B1	3378205	3303500	3317500	3345500	-2.2	-1.8	-1.0
B2	4278521	4174522	4199463	4222860	-2.4	-1.8	-1.3
B3	7943402	7735818	7796636	7839039	-2.6	-1.8	-1.3

The GA solutions are always better than the solutions found by the company. As many metaheuristics, the GA might find better results in more than 1 hour of execution time. Needless to say the company's solution made use of the experience of the scheduler, but not the GA's solution. Viewed as a toll to support decision making, the GA seems to provide a good starting point from which the scheduler can improve lot sizes and schedules for line and tanks. It is worth mentioning that a scheduler normally takes two days to conclude a production plan.

4. Conclusion

In this paper we present a two-level production planning problem where, on each level, a lot-sizing and scheduling problem with parallel machines, capacity constraints and sequence-dependent setup costs and times have to be solved. **Two instance sets were used for comparisons: small-to-moderate sized and industrial instances. All these instances are based on data provided by a real soft drink company in Brazil. In the first part of the computational experiments,** the software GAMS/CPLEX was used to obtain optimal solutions or the best feasible solution at least. This exact solution approach was able to return an optimal solution only for small-to-moderate sized instances with 1 and 2 macro-periods. Only the best feasible solutions were returned at the end of the time limit for small-to-moderate sized instances with 3 and 4 macro-periods T=3,4. The main contribution of this article is a **multi-population genetic algorithm which seems to be the only alternative for the exact method in order to solve real world instances. The GA approach has a new representation of solutions and deals with a hierarchically structured multi-population. A tailor-made decoding procedure is used to evaluate the solution encoded in the gene of each individual. Moreover, a tailor-made recombination is carried out on population clusters. Migrations between different populations are allowed. In the small-to-moderate sized instances, the GA found many optimal solutions or yielded small mean deviations from the best feasible solutions returned by GAMS/CPLEX. Furthermore, GA returned better final solutions for various small-to-moderate sized instances where GAMS/CPLEX returned only a best feasible solution (not optimal) within the time limit. In the industrial instances, after 1 hour of CPU time, the GA was able to return a better solution than the production plan generated by the industry. These results show that the proposed GA can help the decision maker to generate improved schedules.**

Acknowledgement

The authors are indebted to Flávio A.M.Veronese from Companhia de Bebidas Ipiranga, Brazil, for his kind collaboration. This research was partially supported by FAPESP (grant no.00/02609-2) and CNPq (grant 303956/2003-8 and 522973/95-7). **The authors wish to thank the threes anonymous referees for their helpful comments and suggestions on the previous version of this paper.**

References

Berreta, R. E., França, P.M., Armentano, V.A., Metaheuristic approaches for the multilevel resource-constrained lot-sizing problem with setup and lead times. *Asia-Pacific Journal of Operational Research*, 2005, **22**(2), 261-286.

Bitran, G. R. and Matsuo, H., Approximation formulations for the single-product capacitated lot size problem. *Operations Research*, 1986, **34**, 63--74.

Clark, A. R. and Clark, S. J., Rolling-horizon lot-sizing when set-up times are sequence-dependent. *International Journal of Production Research*, 2000, **38**(10),2287--2307.

Clark, A. R., Hybrid heuristics for planning lot setups and sizes. *Computers and Industrial Engineering*, 2003, **45** (4), 545-562.

Dellaert, N., Jeunet, J. and Jonard, N., A genetic algorithm to solve the general multi-level lot-sizing problem with time-varying costs. *International Journal of Production Economics*, 2000, **68**, 241-257.

Dorndorf, U., Pesch, E., Evolution based learning in job shop scheduling enviroment. *Computer & Operations Research*, 1995, **22**, 25--40.

Drexl, A., Kimms, A, Lot-sizing and scheduling - survey and extensions. *European Journal of Operational Research*, 1997, **99**, 221-235.

Fleischmann, B., The discrete lot-sizing and scheduling problem with sequence-dependent setup costs. *European Journal of Operational Research*, 1994, **75**, 395--404.

Fleischmann, B. and Meyr, H., The general lot-sizing and scheduling problem. *OR Spektrum*, 1997, **19**, 11--21.

França, P. M., Armentano, V., Berretta, R. E.,Clark, A. R., A heuristic method for lot--sizing in multi--stage systems. *Computers and Operations Research*, 1997, **24** (9), 861--874.

França, P.M.; Mendes, A.S.; Moscato, P., A memetic algorithm for the total tardiness single machine scheduling problem. *European Journal of Operational Research*, 2001, **1**(132), 224--242.

Gams, Web site. Available online at http://www.gams.com (accessed 11 May 2007).

Goldberg, D.E., Genetic Algorithms in search, optimization, and machine learning. Reading, MA: Addison-Wesley, 1989.

Gupta, D. and Magnusson, T., The capacitated lot-sizing and scheduling problem with sequence-dependent setup costs and setup times. *Computers & Operations Research*, 2005, **32**, 727--747.

Haase, K.; Kimms, A., Lot-sizing and scheduling with sequence dependent setup costs and times and efficient rescheduling opportunities. *International Journal of Production Economics*, 2000, **66**, 159--169.

Hoffman, K. L. and Padberg, M., Improving LP-representations of zero-one linear programs for branch-and-cut. *ORSA Journal of Computing*, 1991, **3**(2),121--134.

Holland, J.H., Adaptation in natural and artificial systems. The University of Michigan Press, 1975.

Java Sun, Web site. Available online at http://java.sun.com (accessed 11 May 2007).

Kang, S., Malik, K., Thomas, L.J., Lotsizing and scheduling on parallel machines with sequence-dependent setup costs. *Management Science*, 1999, **45**, 273--289.

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

Kimms, A. Multi-level lot sizing and scheduling: methods for capacitated, dynamic and deterministic models, 1997 (Physica-Verlag: Heidelberg).

Kimms, A., A genetic algorithm for multi-level, multi-machine lot sizing and scheduling. *Computers & Operations Research*, 1999, **26**, 829-848.

Kuhn, H. and Quadt, D., Lot sizing and scheduling in semiconductor assembly - A hierarchical planning approach, in *Proceedings of the International Conference on Modeling and Analysis of Semiconductor Manufacturing*, 2002, pp.211--216.

Mendes, A. S., França, P. M. and Moscato, P., NP-Opt: An optimization framework for NP problems, in *Proceedings of POM2001 - Internatinal Conference of the Production and Operations Mnagement Society*, 2001, pp.82--89.

Mendes, A. S., The framework NP-Opt and its applications to optimization problems. Doctoral Thesis, State University of Campinas, 2003.

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

Meyr, H., Simultaneous lotsizing and scheduling on parallel machines. *European Journal of Operational Research*, 2002, **139**, 277-292.

Michalewicz, Z., Genetic Algorithms + data structure = evolution programs.Springer-Verlag, 1996.

Özdamar, L. and Birbil, S. I., Hybrid heuristic for the capacitated lot sizing and loading problem with setup times and overtime decisions. *European Journal of Operational Research*, 1998, **110**, 525--547.

Sikora, R., A genetic algorithm for integrating lot-sizing and sequencing in scheduling a capacitated flow line. *Computers & Industrial Engineering*, 1996, **30** (4), 969-981.

Toledo, C. F. M., Integrated two-stage lot sizing and scheduling problem. Doctoral Thesis, State University of Campinas, 2005.

Toledo, C. F. M., Kimms, A., França, P. M and Morabito, R., A mathematical model for the synchronized and integrated two-level lot sizing and scheduling problem. Submitted to *Journal of Operational Research Society* in May 2006.

Weiner, J., The beak of the finch. Vintage Books, New York, 1995.