8

Heuristics and meta-heuristics for lot sizing and scheduling in the soft drinks industry: a comparison study

D. Ferreira[1], P.M. França[2], A. Kimms[3], R. Morabito[1], S. Rangel[4], and C.F.M. Toledo[5]

[1] Departamento de Engenharia de Produção, Universidade Federal de Sao Carlos, C.P. 676, 13565-905, Sao Carlos, SP, Brazil deise@dep.ufscar.br; morabito@power.ufscar.br

[2] Departamento de Matemática, Estatística e Computação, Universidade Estadual Paulista, C.P. 1234, 19060-400, Presidente Prudente, SP, Brazil paulo.morelato@fct.unesp.br

[3] Dept. of Technology and Operations Management, University of Duisburg-Essen, 47048, Duisburg, Germany alf.kimms@uni-duisburg-essen.de

[4] Departamento de Ciência da Computação e Estatística, Universidade Estadual Paulista, Rua Cristóvão Colombo, 2265, 15054-000, S. J. do Rio Preto, SP, Brazil socorro@ibilce.unesp.br

[5] Departamento de Ciência da Computação, Universidade Federal de Lavras, C.P. 3037, 37200-000, Lavras, MG, Brazil claudio@dcc.ufla.br

Summary. This chapter studies 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 must be solved. The problem can be found in soft drink companies where the production process involves two interdependent levels with decisions concerning raw material storage and soft drink bottling. Models and solution approaches proposed so far are surveyed and conceptually compared. Two different approaches have been selected to perform a series of computational comparisons: an evolutionary technique comprising a genetic algorithm and its memetic version, and a decomposition and relaxation approach.

Key words: Two-level Production Planning, Lot Sizing, Scheduling, Soft Drinks Industry, Genetic Algorithm, Memetic Algorithm.

8.1 Introduction

The motivation behind writing this contribution is to offer the academic and practitioner industrial engineering community dealing with planning and scheduling tasks in the soft drinks industry a text with the most recent

D. Ferreira et al.: *Heuristics and meta-heuristics for lot sizing and scheduling in the soft drinks industry: a comparison study*, Studies in Computational Intelligence (SCI) **128**, 169–210 (2008)
www.springerlink.com

contributions to the field and also a comparative study with some selected approaches. A major concern that inspired the chapter was to review modern techniques especially designed for building production schedules applied to real world settings. The technical literature devoted to planning and scheduling is vast and there are plenty of sophisticated methods. However, the specificities of the soft drinks industry require dedicated models and specific solution methodologies that justify a text like this one. Thus, the objective of this chapter is first to discuss the planning features of a soft drinks plant and then to assess the main suitable mathematical models, as well as to explore and evaluate the quality and computing time of some selected solution methods.

8.1.1 Soft Drinks Plant

The consumption of soft drinks has grown considerably worldwide. In Brazil, where part of the present research has been carried out, there are more than 800 plants supplying a 13-billion liter annual consumer market, which is the third in the world. This figure represents an amount which is twice as large as ten years ago. The diversity of products offered to consumers, the scale of plants and the complexity of modern filling lines require the adoption of optimization-based programs to produce efficient production plans. Indeed, a plenty of specialized commercial packages have been launched over the last years as an effort to overcome the difficulties human schedulers have faced. However, in most cases the complexity of the planning task imposes hard manual adjustments for the production schedules produced by those packages. The biggest contribution of the approaches studied in this chapter is to propose integrated optimization-based models able to encompass both the two interdependent production levels, namely the tank level and the bottling level. Due to its inherent complexity, the needed synchronization between these two levels is disregarded by commercial packages thus often leading to ineffective production schedules.

The production process found in medium to large plants consists of an upper level with capacitated mixing tanks used to prepare and store liquids which are pumped to the lower level constituted by bottling or canning lines disposed in parallel (Fig. 8.1). At the tank level, decisions concerning the amount and the time the raw materials have to be stored in every available tank must be made. Analogously, at the production line level the lot size of each demanded item and its corresponding schedule in each line must also be determined. However, a line is able to meet the weekly demand only if the necessary amount of raw material can be stored in a connected tank. Indeed, a solution which integrates these two lot sizing and scheduling problems has to be determined. Moreover, once the necessary amount of raw material is stored, it can not stay for a long time waiting to be pumped to lines. There is a synchronization problem here because the production in lines and the storage in tanks must be compatible with each other throughout the time horizon. Hence, a lot sizing and scheduling problem has to be solved at each

one of these two-levels taking into account that the corresponding decisions must be synchronized.

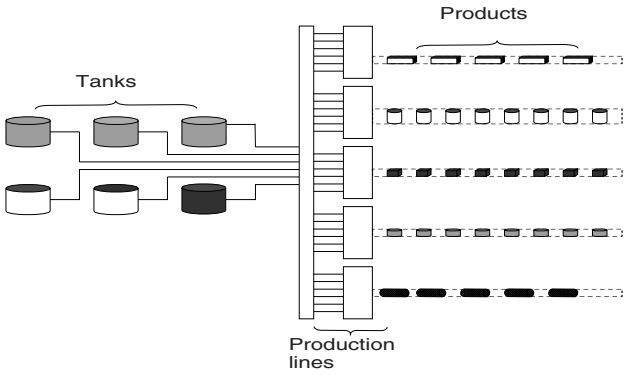

Fig. 8.1: The two-level production process.

Possibly the two-level synchronization is the most challenging aspect of this problem. Due to this fact, this problem is called the Synchronized and Integrated Two-level Lot sizing and Scheduling Problem (SITLSP) [1].

The raw materials are the flavors of the liquids which are bottled in the production lines. For technical reasons, a tank is only filled up when empty and two different raw materials cannot be stored at the same time in the same tank. A sequence-dependent changeover time (setup) of up to several hours occurs to clean and fill up a tank, even if the same soft drink is replaced. A sequence-dependent setup time means that the time required to prepare and fill a tank for the next liquid depends on the liquid previously stored. Indeed, the setup time when a diet drink follows a plain flavor drink is much longer when the sequence is inverted. Nothing can be pumped to a production line from the tank during the setup time. One tank can be connected to several production lines which will share the same raw material. Moreover, the production lines can be connected to any tank. However, it can receive raw material from only one tank at a time. The final product (item) is defined by the flavor of the soft drink and the type of container (glass bottles, plastic bottles or cans) of different sizes. In large plants it is common to find situations where various products can share a common production line and various lines can produce the same product in parallel. The production schedule also has to take into consideration the impact of product changeovers on the effective capacity of the production lines. As in the mixing tanks, these changeover times (setups) are also sequence-dependent and occur whenever a line has two different products switched.

The weekly demands have to be met within a time horizon of a certain number of weeks. Since the forecasts of customer orders are error-prone, there

is little interest in seeking solutions in large horizons. Instead, it is more realistic to work in a rolling-horizon basis with a 3 or 4-week time horizon. The excessive number of final products leads to inventory costs. There are also inventory costs for the storage of raw materials in tanks in various time periods. The sequence-dependent setup costs for products and raw materials are proportional to the sequence-dependent setup times in lines and tanks, respectively.

As synchronization is a key feature to be taken into consideration, a deeper explanation in this respect is now in order. As said before, the lines must wait until the liquids are ready to be pumped to them. On the other hand, the liquids stored in tanks can not be sent to the lines unless they are ready to initiate the bottling process. Fig. 8.2 illustrates the commitment between the two levels.

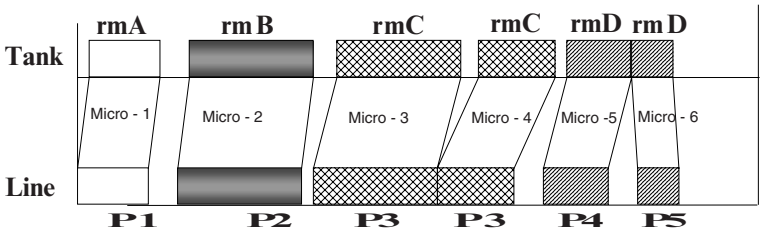

Fig. 8.2: Batches sequenced but not synchronized.

Observe that batches of liquids and items are properly sequenced in the tank and in the line, respectively, but they are not synchronized. The gaps between two batches of liquids (rmA, rmB, rmC, rmD) and items ($P1$, $P2$, $P3$, $P4$, $P5$) represent given changeover times. The planning horizon is divided into 6 micro-periods. Notice that item $P3$ is produced in both micro-periods 3 and 4; it uses the same liquid rmC but needs tank replenishment. Also observe that the same liquid rmD is used in distinct products $P4$ and $P5$ because they make use of, say, different bottle sizes. Due to the discrepancies between tank and line setup times, the product batches have to be delayed by inserting idle times (black rectangles) in the production line in micro-periods 1-4 while the liquid batches must be delayed as well by inserting idle times (empty rectangles) in micro-periods 5 and 6, as shown in Fig. 8.3.

Given that the matter of synchronization has to be treated by postponing batches, it causes impact on the capacities of tanks and production lines and may result in infeasible schedules. Therefore, these actions must be considered together with lot sizing and scheduling decisions. Thinking in terms of mathematical models, they must incorporate decision variables especially designed to deal with the crucial issue of synchronization thus adding substantial complexity to model building and implementation of solution techniques.

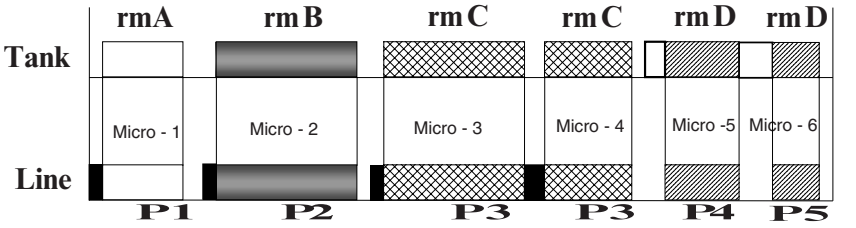

Fig. 8.3: Sequenced and synchronized batches.

8.1.2 Literature Review

The SITLSP is a two-level lot sizing and scheduling problem with some particular considerations - such as the two-level synchronization - which render it much more complex. In this literature review the main published articles addressing the SITLSP itself or similar problems dealing with the planning and scheduling tasks in the soft drinks industry will be summarized and commented on. As a result of this review, the two most promising methods are selected and then described in detail in the next sections, followed by a computational study which compares their performances and application fields.

The SITLSP addresses various issues of classical lot sizing and scheduling problems that have been dealt with in the literature before. We refer to [2] and the references therein for overviews in capacitated lot sizing and scheduling problems. The capacitated lot sizing and scheduling problem is a NP-hard optimization problem [3], but finding a feasible solution is easy (e.g., a lot-for-lot like policy) if no setup times are to be taken into account. If setup times are present, the problem of finding a feasible solution is NP-complete already. A discussion of lot sizing and scheduling with sequence-dependent setup costs or sequence-dependent setup times can be found in e.g. [4]- [8]. Publications addressing multi-level lot sizing (scheduling) problems can be found in e.g. [9]- [13]. Studies regarding these problems with parallel machines are found in e.g. [4], [14]- [18].

However, to the best of our knowledge, the only work that comes close to the SITLSP is the one described in [16], which is a multi-level extension of [8]. In this approach, however, it might be necessary to split up a lot into smaller ones in order not to lose generality. This would not be a practical idea in solving the SITLSP because every new lot for the tanks requires a new setup, which is not desired.

Focusing the attention only on articles which specifically deal with the SITLSP, one can cite [19] where an extensive mixed-integer mathematical model describing the problem is presented. Unfortunately, due to its complexity and size, the model had to be omitted in this chapter. Instead, a brief explanation regarding its formulation is given next. 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). A comparison of the CSLP and GLSP and more details about these problems can be found in [2], [20]- [22].

As shown in [19], the SITLSP model supposes that 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 $t = t_1, t_2, \ldots, t_T$. The limited number of slot assignments is an idea taken from the GLSP. This enables us to determine in the SITLSP for which liquid (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. Fig. 8.4 illustrates the idea. Consider $T = 2$ macro-periods, 5 raw materials (rmA, rmB, rmC, rmD and rmE), 6 products ($P1$, $P2, \ldots, P6$), 3 tanks ($Tk1$, $Tk2$ and $Tk3$) and 3 lines ($L1$, $L2$ and $L3$). Suppose that the raw material rmA produces the 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.

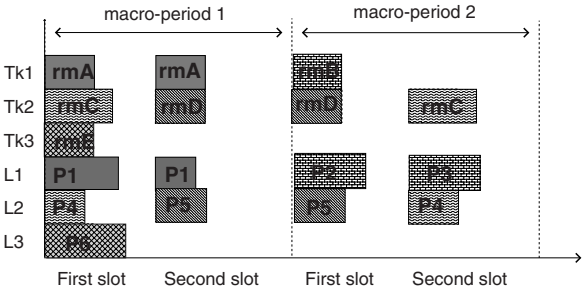

Fig. 8.4: Sequence of slots for tanks and lines.

Fig. 8.4 shows that only one slot is occupied by lots of raw materials and products in tank $Tk3$ and line $L3$, respectively, during the first macro-period. In the second macro-period, there is no slot occupied in $Tk3$ and $L3$. On the other hand, two slots need to be occupied by lots of rmA in the first macro-period of $Tk1$. This raw material is used to produce $P1$ which is assigned to the two possible slots in the first macro-period of $L1$. In this case, the necessary amount of rmA to produce $P1$ filled up $Tk1$ completely in the first slot assignment. Another slot assignment of rmA is necessary to conclude the production of $P1$. Refilling $Tk1$ with rmA leads to an interruption in $L1$, so a second slot assignment of $P1$ in $L1$ is also made. Variables indexed by slots in the SITLSP mathematical model made it possible to write constraints that integrate the line and tank occupation [1, 19].

As well as the assignment of liquids to tanks and products for lines in each macro-period, it is also necessary to synchronize the slots scheduled in

a two-level problem like this. This is done using the micro-period idea found in the CSLP, where each macro-period t is divided into T^m micro-periods with the same length. Furthermore, according to the CSLP assumptions, the capacity of each micro-period can be total or partially occupied and only one product type (item) can be produced per micro-period. These assumptions are used in the SITLSP. Fig. 8.5 illustrates this idea using the assignment shown in Fig. 8.4.

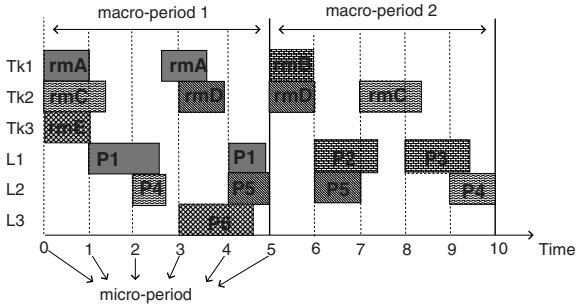

Fig. 8.5: Synchronization between lines and tank slots.

The macro-periods are divided into 5 micro-periods of the same length. The times necessary to prepare each tank (setup time) are represented now by the slots with acronyms rmA, rmB, rmC, rmD and rmE. In the same way, the processing time of each product is represented by the slots with symbols $P1$, $P2$, $P3$, $P4$ and $P5$. For instance, the two slots of $Tk1$ occupied by rmA indicate that the tank is ready to be used at the end of the first micro-period and it is refilled with the same raw material between the third and fourth micro-period. The micro-periods enable us to synchronize the beginning of $P1$ production in the second micro-period of $L1$ after the end of rmA setup time in $Tk1$. Moreover, we can see when the refilling of $Tk1$ occurs in the second slot of $Tk1$. This requires the interruption of $P1$ and the beginning of the second slot occupation of $P1$ in $L1$, after the end of rmA setup time in $Tk1$. The constraints and variables indexed by the micro-period make it possible to describe these situations in the mathematical model of the SITLSP [1, 19].

The objective function is to minimize the total sum of setup costs, inventory holding costs, and production costs for tanks and production lines. Observe that for a given demand a feasible solution may not exist. In order to guarantee a feasible solution the model allows for shortages. This is modelled by allowing every item a certain quantity of units to be "produced" in the first period without using capacity. Naturally, a very high penalty M is attached to such shortages so that, whenever there is a feasible solution that fulfills all demands, one would prefer this one.

In short, one can say that the main contribution of the approach used to model the SITLSP is to integrate and synchronize two lot sizing and scheduling problems. This is done using variables and constraints indexed by slots and micro-periods. The variables and constraints indexed by slots help to integrate a feasible sequence of occupation in the two levels, while variables and constraints indexed by micro-periods help to determine a feasible synchronization in the two levels within the time horizon. The entire set of variables and constraints is able to mathematically describe most of the decisions and constraints present in the industrial problem studied here.

The SITLSP model is coded and solved using the GAMS/CPLEX package that uses a branch-and-cut solution approach to find an optimal solution. For small instances the method is fast and reliable but as problem size grows, the number of distinct integer solutions increases exponentially, causing the search to take too long even for finding the first integer feasible solution. The computational results in a series of instances with $T = 1$, 2 , 3 and 4 macro-periods and $T^m = 5$ micro-periods revealed that for $T > 2$ the method failed in finding optimal solutions within 1 hour of execution time.

A second paper [23] dealing with the SITLSP introduces an evolutionary approach capable of overcoming the limitations faced by the previous method [19] which is not useful to solve real world instances. Being an approximate approach, optimal solutions are not guaranteed but performance comparisons carried out in the paper using a set of small instances with known optimal solutions have shown good behavior in reasonable computing time.

Evolutionary algorithms (EAs) belong to a class of computational methods called bio-inspired systems that simulate biological processes such as crossover, mutation and natural selection. The simulation of these processes follows the method when searching for a solution (individual) of a specific problem. The method begins by analyzing a set of problem solutions (population) and determines new solutions applying genetic operators (selection, crossover and mutation) to the previous set of solutions. Some solutions (new or old ones) are selected and form the new set of solutions for the next iteration. This procedure is repeated until some stop criterion is satisfied. The first work concerning EA was presented by [24] and details on their implementation can be found in [25] and [26]. A memetic algorithm (MA) is a hybrid population-based approach [27] which combines the recognized strength of population methods such as genetic algorithms (GA) with the intensification capability of a local search. In a MA, all agents or individuals evolve solutions until they are local minima of a certain neighborhood, i.e. after steps of recombination and mutation, a local search is applied to the resulting solutions.

The GA presented in [23] proposes a multi-population approach that can be understood as a variant of island models. The method was developed using the NP-Opt that is an object-oriented framework written in JAVA code [28]. This framework has optimization procedures based on evolutionary computation techniques to address NP-hard problems.

The computational experiments carried out with the GA approach on a set of large instances with up to 12 macro-periods, 10 micro-periods and 15 final

products show that the method is able to solve real-world instances. Consequently it was one of the approaches selected to take part in the comparisons using real data.

Recently another interesting method capable of handling the SITLSP and also applied to the planning and scheduling of soft drinks has been proposed. Unlike the approach presented in [1] where a highly general (and complex) model is introduced and solved by a commercial MIP package, the MIP model formulated in [29] is more restricted and amenable to be solved by approximate approaches. Amongst other differences that will be pointed out in detail in Section 8.3, the mathematical formulation is less general than the one presented in [1] as it forces the quantity of tanks to be the same as for the lines. On the other hand it opens up the possibility that a tank can be filled with all liquids needed by its line.

Alternatively from [1], which uses a MIP package for finding optimal solutions, the solution approach adopted in [29] relies on different heuristic methods to obtain approximate production schedules. The details of these heuristics will be explained fully in the next section.

In [30] a mathematical programming model to deal with the planning of a canning line at a drinks manufacturer is introduced. Compared to the models presented in [1] and [29], this one is much more restricted because it disregards any sequencing considerations when designing production plans. Instead, it focuses only on planning issues, i.e., lot sizing problems. The MIP model objective function minimizes inventories and backorder penalties. The model allows for different setup times depending on whether the changeover between canned products involves a change of liquid or not. However, the model does not consider sequence-dependent setup times. Solution strategies consist of a "lazy" default - simply let an industrial strength branch-and-bound MIP solver try to find a good solution within a pre-specified amount of time - to more sophisticated heuristic approaches. One of these that is worth mentioning is a local search-based meta-heuristic which the author called diminishing-neighborhood search (DNS). In this approach one starts the method with the largest possible neighborhood to avoid bad local optima, and then narrow the neighborhood in a continuously diminishing way as the search proceeds towards a good solution, even if it is still a non-global optimum. The methods were tested in real data instances with up to 41 distinct products to be filled from 14 different liquids in a planning horizon made up of 13 consecutive weeks (periods). The final conclusion is that the methods present a classical trade off between quality and CPU time with the best results being obtained by DNS at the expense of 2 hours of computing time. In spite of the interesting solution methods and results reported in [30], the approach will not be included in the comparisons mainly due to the more restrictive character of the production system for which it was developed.

Closing this introduction, one can conclude that the methods for dealing with the SITLSP embedded in a soft drinks production planning scenario, and most importantly with potential to be applied in real world situations

in a broad sense, are the EA approaches as the one proposed in [23] and the heuristic model-based method presented in [29]. Therefore, a computational comparison involving these two approaches are conducted in a series of practical instances obtained from a large soft drinks manufacturer. One of the purposes of the comparison is to suggest practical guidance on which method best suits the different possible situations found in the industry. The other sections are organized as follows. In Section 8.2 an MA approach based on the GA developed in [23] is proposed while the methods introduced in [29] are presented in Section 8.3. Section 8.4 shows the instances used in the computational comparisons and discussions on the test results. Finally, Section 8.5 concludes the chapter and discusses some topics for further research.

8.2 Evolutionary Approaches

In this section a memetic version of the GA presented in [23] is proposed to solve the SITLSP . The term "Memetic Algorithms" [27,31] (MAs) was introduced in the late 80s as a class of meta-heuristics that have the hybridization of different algorithmic approaches as a crucial aspect in their conceptions. The majority of the MA applications are population-based approaches in which a set of cooperating and competing agents are engaged in periods of individual improvement of the solutions while they sporadically interact. The adjective 'memetic' comes from the term 'meme', coined by Dawkins [32] as an analogy to the term 'gene' in the context of cultural evolution. It stands for the unit of information which reproduces itself as people exchange ideas. MAs are also referred in the technical literature as hybrid genetic algorithms and usually they take a less sophisticated conception as a combination of GAs with a local search procedure applied to some of the individuals in the population.

8.2.1 The MA structure

EAs are global search procedures inspired by biological evolution processes [24, 25]. Amongst the EAs, the GAs are the most popular. A GA differs from local search or constructive heuristics because it has an initial set of solutions (individuals) which have been randomly established. Also called chromosomes, these individuals are solution representations for the problem to be solved. At each GA generation, the individual's fitness is measured and genetic operators are executed in the population. These operators are based on genetic behavior such as crossover, mutation and selection. The individuals with better fitness values remain in the population from one generation to another. MAs and GAs have been applied to solve complex and real-world problems (see [33]- [35]).

The evolutionary methods presented in this section are conceived as a multi-population approach with a hierarchical ternary tree structure. The multi-population approach was chosen because populations that evolve separately usually have different characteristics according to the genetic drift

idea [36]. This can lead to a more effective exploration in the solution space of the problem. A better performance of hierarchically structured populations over non-structured population schemes in EAs has been attested in previous experiments concerning different problems (e.g. machine scheduling, asymmetric travelling salesman, capacitor placement). Computational results solving these optimization problems were reported by [18], [28] and [37] using a multi-population GA with a hierarchical ternary tree structure. Furthermore, the authors in [38] reported that the results obtained with the total tardiness single machine scheduling problem using GA with hierarchically structured populations are better than the ones with non-structured populations. These findings have been confirmed by the multi-population MA developed to solve 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 [23].

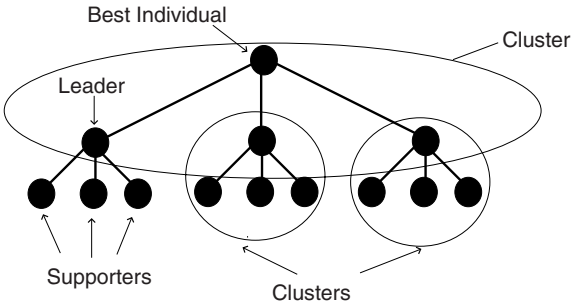

Fig. 8.6: MA clustered population.

The MA population structure consists of several clusters, each one having a leader solution and three supporter solutions, as shown in Fig. 8.6. The leader of a cluster is always fitter than its supporters. As a consequence, top clusters tend on average to have fitter individuals than bottom clusters. As new individuals are constantly generated, replacing old ones, periodic adjustments are necessary to keep this structure well ordered. The number of individuals in the population is restricted to the numbers of nodes in a complete ternary tree, i.e. $(3k-1)/2$ where k is the number of levels of the tree. That is, 13 individuals are necessary to construct a ternary tree with 3 levels, 40 to make one with 4 levels and so on. Previous experiments with distinct tree structures (binary, ternary, etc.) and a variable number of levels (two, three, etc.) attested that the best results were obtained by a 3-level ternary tree [6]. Observe that the population is constituted by four distinct clusters, three in the bottom level and one in the upper level, with four leaders, respectively. The upper level leader (best individual) is always the fitter individual in the population.

The MA procedure applied to the structured population is summarized in Algorithm 8.1. The initial population *Pop*() is generated and submitted to a generation loop. Operators *recombinePop*() and *mutatePop*() produce a new individual (or more than one) which is improved by a local search algorithm in *optmizePop*(). In *structurePop*() the population is re-structured to maintain the hierarchy between agents.

Algorithm 8.1: The MA procedure.

```
initializePop();
repeat
    recombinePop();
    mutatePop();
    optimizePop();
    structurePop();
until Termination condition;
```

The operator *recombinePop*() performs a crossover over a cluster (selected at random) and always involves a supporter node (selected at random) and its corresponding leader node. The new individual (Child) is submitted to a mutation procedure *mutatePop*() depending on the mutation probability test result. The operator *optimizePop*() applies a local search to Child and if it is now better than some parents, the Child will replace the parent with the worst fitness value (Fig. 8.7). Otherwise, the new individual is not inserted into this population. After every crossover/mutation/optimization/replacement operation, adjustments carried out by *structurePop*() are necessary to keep the cluster structure well ordered where the best is always the leader (Fig. 8.8).

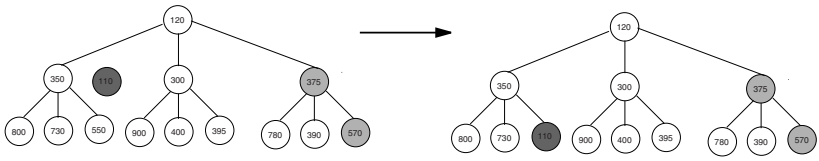

Fig. 8.7: Population before and after Child insertion.

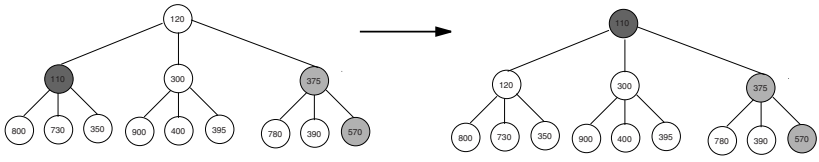

Fig. 8.8: Adjustments in the population.

8.2.2 The multi-population structure

Multi-population MAs are common in implementations in which computational tests are executed on parallel computers. Usually in such case, each processor is responsible for one population. Results obtained on parallel computers are in general much better than the ones obtained on sequential machines. Even though the computational implementation and tests have been carried out on a single-processor computer, the multi-population scheme has been implemented to take advantage of the hierarchical population structure. This decision is also supported by the fact that the SITLSP is a complex combinatorial problem for which simplistic evolutionary solution approaches tend to fail. Fig. 8.9 shows how four populations interact. After a certain number of executions of the genetic operators crossover and mutation performed in each population, followed by a local search step applied to the best individual, one can check if the population convergence occurred, i.e. if no new individuals are inserted in it. If not, the process is repeated until convergence of all the populations. In this case the migration step takes place with a copy of each best individual being inserted into the next population and by replacing some individual randomly selected - except the best one.

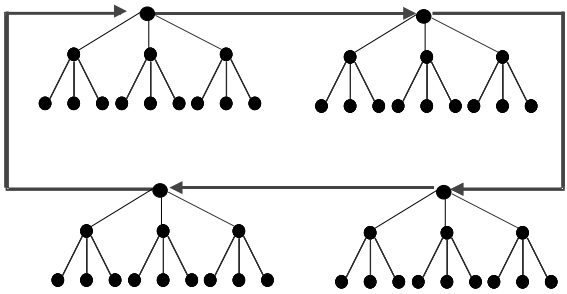

Fig. 8.9: Migration policy.

The pseudo-code shown in Algorithm 8.2 summarizes the whole multi-population MA algorithm. The algorithm executes a fixed number of generations in each population while there is no convergence. This process involves a parent selection (selectParents), a new individual creation by crossover execution (crossover(individualA, individualB)), a possible mutation execution 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 generations given by the parameter $\gamma * PopSize$, where γ is the crossover rate and $PopSize$ is the population size. A migration between populations (executeMigration) is

executed when all populations have converged and the stop criterion has not been satisfied yet. A new initialization of the populations (initializePopulation(pop(i))) will occur, but the best individual and the migrated individuals are kept. The stop criterion is usually a pre-specified computing time.

The MA described in this section was implemented using the NP-Opt [28, 37], an object-oriented framework written in JAVA code which contains procedures based on evolutionary computation techniques to address NP-hard problems.

Algorithm 8.2: Pseudo-code for the multi-population memetic algorithm.

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

8.2.3 Individual representation

A MA approach was developed to solve a lot sizing and scheduling problem with sequence-dependent setup times in [39], where an individual is represented by a string of paired values (type of product and lot size) for each scheduling period. A similar representation of solution as an individual is proposed by [40] for a GA used to solve the capacitated lot sizing and loading problem with setup times, parallel facilities and overtime. In this approach, the individual is also a string of paired values, where the first value is the lot size and the second is the facility. A GA with more elaborated solution representation is proposed by [41], where a binary matrix PxT (P products and T periods) represents an individual for the multi-level lot sizing problem.

Each binary entry $y_{i,t} = 1$, if a setup for product i occurs in t; otherwise $y_{i,t} = 0$. All the GA publications mentioned have specific genetic operators. Their crossover, mutation and selection procedures were designed to deal with those individual representations for lot size and scheduling problems.

A new solution representation is proposed in this work. It is close to one presented in [42] that uses assignment rules in a multi-level proportional lot sizing and scheduling problem with multiple machines. Fig. 8.10 introduces the individual representation proposed for the SITLSP.

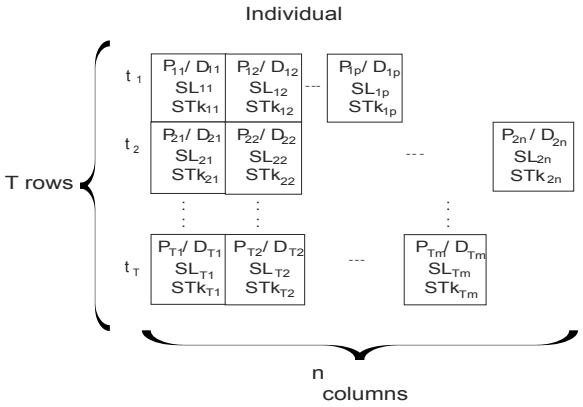

Fig. 8.10: Individual for the SITLSP.

An individual is a two-dimensional matrix with T rows and N columns. The number of rows represents the number of macro-periods t_1, $t_2, \ldots, t_T$. The number of columns represents the number of genes and there can be a different number of genes per macro-period. Each gene corresponds to a cell (m, n), $m \in T$, $n \in N$, in the individual matrix and 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 micro-period t is divided into several lots (D_{mn}) and randomly distributed among the genes in micro-periods t, $t-1$, $t-2, \ldots, 1$. The sequences SL_{mn} and STk_{mn} are randomly generated with length k. Sequence SL_{mn}=$(\alpha_1, \ldots, \alpha_k)$ with $\alpha_i \in \{1, \ldots, L\}$, where α_i is a possible line number and L is the number of lines. The value α_i is taken from L possible values. Sequence STk_{mn}=$(\beta_1, \ldots, \beta_k)$ with $\beta_i \in \{1, 2, \ldots, 2\overline{L}\}$, where β_i defines where and how the raw material will be stored. Parameter $\overline{L}$ is the

number of tanks. The β_i is taken from $2\overline{L}$ possible values, but the real tank number j is obtained from β_i using:

$$j = \begin{cases} \beta_i, & 1 \leq \beta_i \leq \overline{L}; \\ \beta_i - \overline{L}, & \overline{L} < \beta_i \leq 2\overline{L}. \end{cases} \tag{8.1}$$

If $1 \leq \beta_i \leq \overline{L}$, the tank $j = i$ will be occupied after the raw material previously stored has been used. This forces the method to find solutions where there is a partial use of the tank capacity. If $\overline{L} < \beta_i \leq 2\overline{L}$, the tank $j = \beta_i - \overline{L}$ will be immediately occupied. This forces the method to find solutions where the tank capacity is completely used. These conditions have some exceptions. If tank j, selected by one of the previous criterions, stores a raw material different from the raw material of the product P_{mn}, it will be necessarily occupied after the raw material which was previously stored has been used. The same will happen if tank j is completely full. On the other hand, if tank j is empty, it will be immediately occupied. If j is not empty, but the raw material stored is the same as P_{mn} and the minimum tank capacity has not been satisfied, this tank will be also occupied immediately. The individuals in each initial population are generated following the pseudo-code illustrated in Algorithm 8.3. There are T macro-periods, J products, L lines and $\overline{L}$ tanks. 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. At this point, the sequences of lines (SL_{mn}) and tanks (STk_{mn}) are also randomly generated.

Algorithm 8.3: Individual initialization algorithm.

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

The following example clarifies the solution representation of an individual. Suppose two products ($P1$ and $P2$) where each product has a demand of 100 units to be filled in macro-period t_1 and another 200 units to be filled in macro-period t_2. The products use different raw materials ($Rm1$ and $Rm2$, respectively). Moreover, there are two lines available to produce both products and two tanks available to store both raw materials. Fig. 8.11 shows two possible representations, both based on the individual initialization algorithm. The demands are distributed in their respective macro-periods in individual 1. However, the demand of $P1$ in t_1 is split between two genes. The same happens with the demand of $P2$ in t_2. In individual 2, part of the $P1$ demand in t_2 is split between two genes in t_1. Notice that the sequence of lines (SL_{mn}) and tanks (STk_{mn}) can repeat values of $\alpha_i \in \{1, 2\}$ and $\beta_i \in \{1, 2, 3, 4\}$ for $L = \overline{L} = 2$ and $k = 4$ (length).

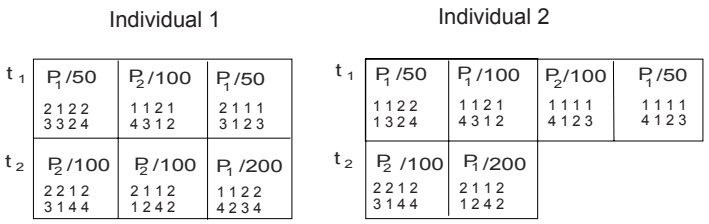

Fig. 8.11: Two possible individuals.

8.2.4 Decoding and evaluation

The decoding procedure is responsible for determining a problem solution from the data encoded in 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 enable 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 the last section and individual 1 shown there. Let's also suppose that for each macro-period there are 5 micro-periods with the same length. Therefore, the time horizon is divided into 10 micro-periods. The process begins by the first gene in the last macro-period (Fig. 8.12). A lot of product $P2$ ($D21 = 100$) has to be produced using the first pair $(\alpha_1, \beta_1) = (2, 3)$ from sequences SL_{21} and STk_{21}. Product $P2$ is produced in line 2 because $\alpha_1 = 2$ and let's assume that its processing time takes two micro-periods. The other processing times used in this example are suppositions as

well. According to equation (1), raw material $Rm2$ of $P2$ has to be stored in tank j=3-2=1 because β_1=3 and $2 < \beta_1 < 4$ with $\overline{L} = 2$. Given the criteria defined in Section 8.2.2, tank j=1 is empty and it must be occupied immediately. A tank should be ready at least one micro-period before the production starts on the lines. Therefore, the setup time of $Rm2$ occurs one micro-period before the $P2$ production starts in $L2$ (see Fig. 8.12).

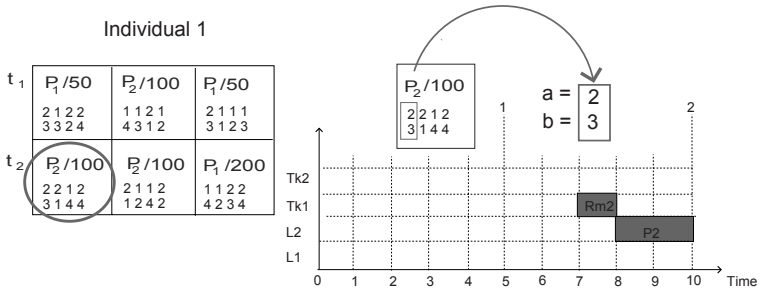

Fig. 8.12: Decoding of the first gene in t_2.

Let us suppose that the setup time of raw materials will take one micro-period in any tank in this example. The demand of this gene has been completely scheduled, so the next gene in t_2 is decoded now (Fig. 8.13). The first pair of rules $(\alpha_1, \beta_1) = (2, 1)$ is selected to schedule 100 units of $P2$ which are produced in $L2$ ($\alpha_1 = 2$). Value β_1=1 means $j = 1$ ($1 \leq \beta_1 \leq 2$) by equation (1) and this tank must be occupied after the raw material which was previously stored has been used. At this point, it is worth noticing that this criterion allows for establishing schedules with a tank partially filled. Fig. 8.14 shows the decoding of the third gene in t_2. A lot from product $P1$ ($D21 = 200$) has to be scheduled in line $L1$ ($\alpha_1 = 1$) and its raw material has to be stored in the empty tank j = 4-2 = 2 ($2 < \beta_1 \leq 4$). However, let's suppose now that tank $j = 2$ has a capacity of storing raw material sufficient to produce only 100 units of $P1$. In this case, the next pair (α_2, β_2)=(1,2) is selected to schedule the remaining lot $D12$ = 200-100 = 100 (Fig. 8.15). The remaining lot is also produced in $L1$ ($\alpha_2 = 1$) and a new setup time of $Rm1$ occurs in tank $j = 2$.

The decoding process continues in the first gene of t_1 (Fig. 8.16). Product $P2$ is produced in $L2$ ($\alpha_1 = 2$). A setup time occurs because $P1$ is produced next in $L1$. Let's suppose that the setup time from $P1$ to $P2$ takes one micro-period. The tank j = 1 (β_1=3 with $2 < \beta_1 \leq 4$) should be immediately occupied, but it already stores $Rm1$. In this case, a new lot assignment to this tank is necessary.

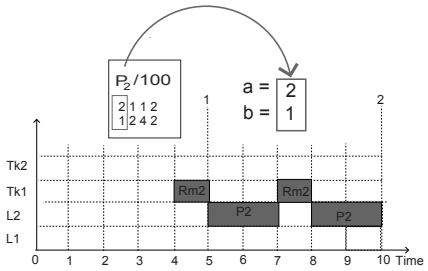

Fig. 8.13: Decoding of the second gene in t_2.

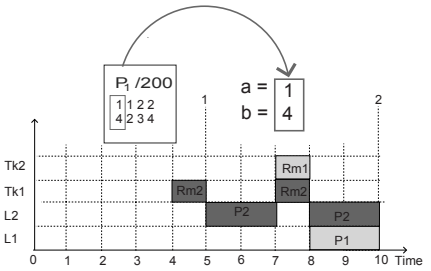

Fig. 8.14: Decoding of the third gene in t_2.

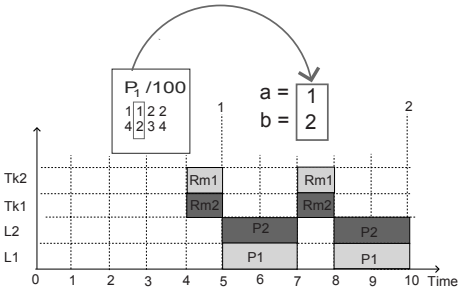

Fig. 8.15: Decoding of the remaining demand of the third gene in t_2.

The next gene decoding is shown in Fig. 8.17. Line $L1$ ($\alpha_1 = 1$) is selected to produce $D12 = 100$ units of $P2$. Let's suppose that the setup time from $P2$ to $P1$ takes two micro-periods. Raw material $Rm2$ is assigned to $j = 4\text{-}2 = 2$ ($\beta_1 = 4$) and it must be ready one micro-period before $L1$ produces $P2$. Fig. 8.18 has the last gene decoding. Product $P1$ is scheduled in $L2$ ($\alpha_1 = 2$) and there is no setup time because $P1$ is also produced next. $Rm1$ will integrate the lot previously stored in tank $j = 3\text{-}2 = 1$ because $\beta_1 = 3$. For this reason, the setup time of tank $j = 1$ is anticipated to the second micro-period.

8.2.5 Crossover and mutation

Previous computational experiments with various crossover operators revealed that the best behavior was attained by the uniform crossover. In this recombination operator, genes from two different parents that occupy the same position in the individuals have some probability of being inherited by the child. Individuals 1 and 2 (Fig. 8.11) are used to show how the uniform crossover

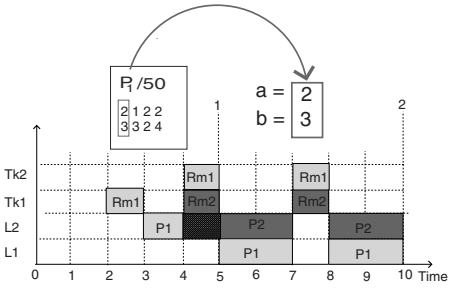

Fig. 8.16: Decoding of the second gene in t_2.

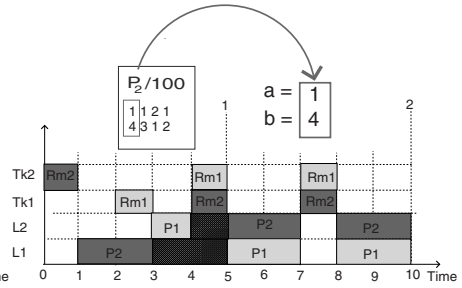

Fig. 8.17: Decoding of the third gene in t_2.

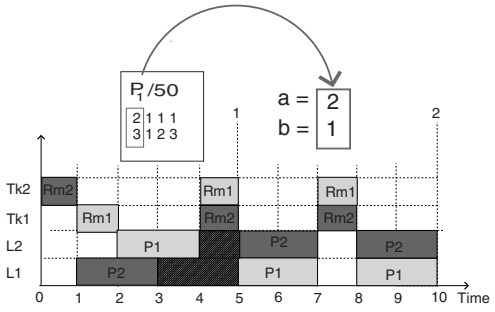

Fig. 8.18: Decoding of the remaining demand of the third gene in t_2.

operator works. Sequences SL_{mn} and STk_{mn} are not relevant because the new individual (Child) will inherit these sequences without changes. Fig. 8.19 illustrates the crossover of Ind1 and Ind2.

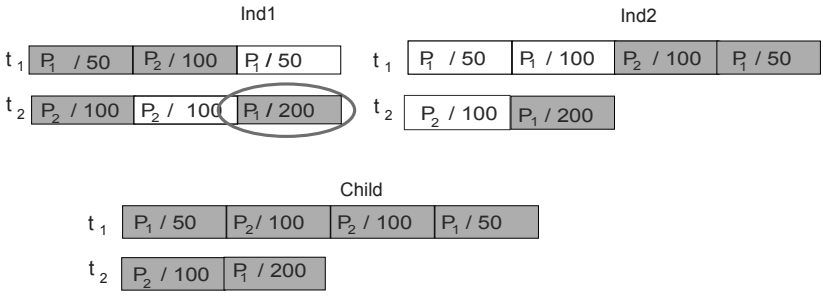

Fig. 8.19: Uniform crossover example.

For each gene in the same position in Ind1 and Ind2, a random value $\lambda \in [0, 1]$ is generated. If $\lambda < 0.5$, the Child inherits the gene from Ind1; otherwise, the Child inherits the gene from Ind2. The genes selected following this procedure are shaded in Fig. 8.19. Notice that there are more genes in macro-periods t_1 of Ind2 than in the same macro-period 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 from Ind1 marked by a circle in Fig. 8.19 is not inherited because it would exceed the total demand of $P1$ in the Child. If there is a gene in the same position in Ind2, this gene must be inherited by the Child if the same problem does not occur. On the other hand, a lack of demand can take place at the end of the crossover. For this reason, a repair procedure is necessary and the demand deficits in some macro-period are inserted.

Mutation aims to keep diversity in a population avoiding premature convergence. A mutation rate determines the number of individuals that are changed. The mutations adopted basically swap gene positions (Fig. 8.20).

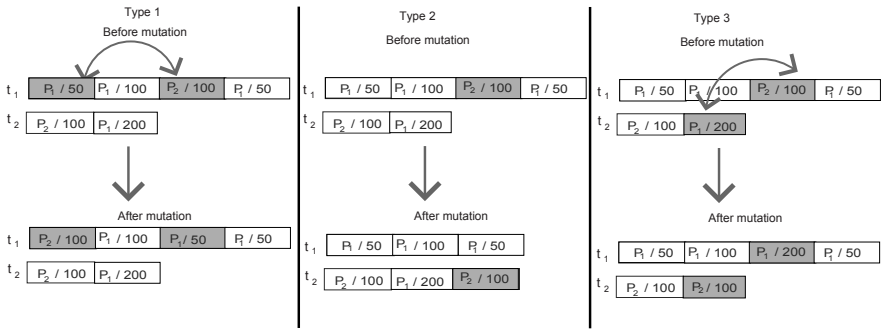

Fig. 8.20: Mutation types.

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.

8.2.6 Local search algorithm

A substantial part of the computational effort related to MA implementations is due to the local search. Bearing this in mind, and also that SITLSP is a

complex combinatorial problem to be solved in a real-world context, a simple local search method is a natural choice. A threshold accepting (TA) procedure was elected as the local search built-in method to be used in this MA implementation [43]. The pseudo-code of the TA is shown in Algorithm 8.4.

The neighborhood movements can lead to worse individuals and the TA accepts them, since their fitness value remain inside a threshold. Doing this the method can skip from local minima. Threshold reductions lead the method to convergence. Two neighborhood movements were carried out to modify the best individual making changes in the gene positions and changes in the lot size. The changes in the gene positions follow the same behavior of the three types of mutations described in the previous section. Regarding the mutation types, the allowed movements are:

- Swap positions of two genes in the same macro-period.
- Remove one gene and insert it into another position.
- Swap positions of two genes in different macro-periods.

The changes in the lot sizes consider two cases:

- Split lots of one gene in two pieces. One piece stays in the gene and the other piece is inserted into another position of the individual.
- Merge lots of the same product. The lot of one gene is removed from its original position and inserted into another gene with the same product.

Problems with demand satisfaction are taking into account, so neighborhood movements that lead to infeasibility are forbidden. After the fixed number of generations has been completed, the MA takes the best individual of the population and executes the TA local search. First, one of the three possible changes in the gene positions are randomly chosen and executed. After a maximum number of iterations, the TA stops and restarts executing now changes in the lot sizes. Moreover, the two possible changes in the lot sizes are randomly chosen and executed over randomly selected genes. In the end, if a better individual is obtained, it is inserted as the new best individual of this population.

Algorithm 8.4: Local search procedure.

```
individual = bestIndividual;
repeat
    newIndividual = moveExecution(individual);
    Δf = fitness(Individual)-fitness(newIndividual);
    if Δf > −Th∗fitness(individual) then
        individual = newIndividual;
    else
        reduce(Th);
    end
until maxNumberOfIteration;
```

The MA as conceived here transforms itself into a GA simply by the deleting the local search procedure. Although our experience with previous research has demonstrated that in most cases the MA versions outperform their GA versions, a comparison within a fixed amount of time will be carried out in Section 8.4. The underlying issue under consideration is whether the time spent by the local search executions in the MA is favorably used to reach better solutions or if it could be used better by the GA version which will spend this time executing a higher number of generations.

8.3 The Decomposition and Relaxation Approach

In this section the mathematical model to represent the SITLSP and the solution approaches proposed in [29] are described. The model considers the synchrony between the production levels and integrates the lot sizing and scheduling decisions as well as the model in [19]. However, as pointed out in Section 8.1.2, it differs from the latter in many aspects. A simplification of the problem is considered supposing that each filling line, thereafter called machine, has a dedicated tank. Each tank can be filled, in turn, with all the liquids needed by the associated machine. The planning horizon is divided into T macro-periods. It is a big bucket model and to obtain the order at which the items will be produced each macro-period is divided into a number of micro-periods. The total number of micro-periods is defined by the user, but should be set as the maximum number of setups in each macro-period. The micro-period size is flexible and is defined by the model since it depends on the item's lot size. The total number of micro-periods in both levels is the same and only one item (liquid flavor) can be produced in each micro-period.

8.3.1 Model development

To describe the Two-Level Multi-Machine Lot Scheduling Model (P2LMM) given in [29, 44], let the following parameters define the problem size:

J = number of soft-drinks (items);
M = number of machines (and tanks);
F =number of liquid flavors;
T = number of macro-periods;
N = total number of micro-periods (i.e. total number of setups);

and let (i, j, m, k, l, t, s) be the index set defined as:

$$i, j \in \{1 \dots J\}; t \in \{1 \dots T\}; k, l \in \{1 \dots F\}; s \in \{1 \dots N\}; m \in \{1 \dots M\}.$$

Consider also, that the following sets and data are known:

Sets:

S_t = set of micro-periods in each macro-period t;
P_t = first micro-period of period t;
ρ_j = set of machines that can produce item j;
δ_m = set of items that can be produced on machine m;
θ_m = set of liquid flavors that can be produced on tank m;
ω_{ml} = set of items that can be produced on machine m and need flavor l.

The data and variables described below with superscript I relate to Level I (tank) and with superscript II relate to Level II (bottling):

Data:

d_{jt} = demand for item j in macro-period t;
h_j = non-negative inventory cost for item j;
g_j = non-negative backorder cost for item j;
s^I_{kl} = changeover cost from liquid flavor k to l;
s^{II}_{ij} = changeover cost from item i to j;
b^I_{kl} = changeover time from liquid flavor k to l;
b^{II}_{ij} = changeover time from item i to j;
a^{II}_{mj} = production time in machine m of item j;
K^I_m = total capacity of tank m, in liters of liquid;
K^{II}_{mt} = total time capacity in machine m in period t;
r_{jl} = quantity of liquid flavor l necessary for the production of one lot of item j;
q^I_{ls} = minimum quantity to produce liquid flavor l in micro-period s;
I^+_{j0} = Initial inventory for item j.

Variables:

I^+_{jt} = inventory for item j at the end of macro-period t;
I^-_{jt} = backorder for item j at the end of macro-period t;
x^{II}_{mjs} = production quantity in machine m of item j in micro-period s;

$$y^I_{mls} = \begin{cases} 1, \text{ if the tank } m \text{ is setup for syrup } l \text{ in micro-period } s; \\ 0, \text{ otherwise} \end{cases}$$

$$y^{II}_{mjs} = \begin{cases} 1, \text{ if the machine } m \text{ is setup for item } j \text{ in micro-period } s \\ 0, \text{ otherwise} \end{cases}$$

$$z^I_{mkls} = \begin{cases} 1, \text{ if there is changeover in tank } m \text{ from syrup } k \text{ to } l \text{ in } s; \\ 0, \text{ otherwise} \end{cases}$$

$$z^{II}_{mijs} = \begin{cases} 1, \text{ if there is changeover in machine } m \text{ from item } i \text{ to } j \text{ in } s; \\ 0, \text{ otherwise} \end{cases}$$

To include the synchrony between the two production levels in the P2LMM model another set of variables is necessary. As discussed in Section 8.1, a machine must wait until the liquid is ready in the tank. The set of continuous variables, $v_{ms}^{II} \geq 0$, computes this waiting time for each machine m, in each micro-period s. The waiting time is equal to the difference between the tank changeover time and the machine changeover time. That is:

$$v_{ms}^{II} \geq \sum_{k\in\delta_m}\sum_{l\in\theta_m} b_{kl}^{I} z_{mkls}^{I} - \sum_{i\in\delta_m}\sum_{j\in\delta_m} b_{ij}^{II} z_{mijs}^{II} \qquad m=1,\ldots,M, s=1,\ldots,N$$

If the machine changeover time from item i to j is greater than the tank changeover time from liquid flavor k to l, the waiting variable is zero and only the machine changeover time is considered in the associated capacity constraint. Otherwise, the total waiting time of the macro-period is taken into account.

The P2LMM model is then:

$$Min \;\; Z = \sum_{j=1}^{J}\sum_{t=1}^{T}(h_j\, I_{jt}^{+} + g_j\, I_{jt}^{-}) + \sum_{m=1}^{M}\sum_{s=1}^{N}\sum_{k\in\theta_m}\sum_{l\in\theta_m} s_{kl}^{I}\, z_{mkls}^{I} + \sum_{m=1}^{M}\sum_{s=1}^{N}\sum_{i\in\delta_m}\sum_{j\in\delta_m} s_{ij}^{II}\, z_{mijs}^{II} \tag{8.2}$$

Subject To

Level I (Tank)

$$\textstyle\sum_{j\in\omega_{ml}} r_{lj} x_{mjs}^{II} \leq K_m^{I} y_{mls}^{I}, \qquad m=1,\ldots,M, l\in\theta_m, s=1,\ldots,N; \tag{8.3}$$

$$\textstyle\sum_{j\in\omega_{ml}} r_{lj} x_{mjs}^{II} \geq q_{ls}^{I} y_{mls}^{I}, \qquad m=1,\ldots,M, l\in\theta_m, s=1,\ldots,N; \tag{8.4}$$

$$\sum_{l\in\theta_m} y_{ml(s-1)}^{I} \geq \sum_{l\in\theta_m} y_{mls}^{I}, \qquad m=1,\ldots,M, t=1,\ldots,T, s\in S_t-\{P_t\}; \tag{8.5}$$

$$z_{mkls}^{I} \geq y_{mk(s-1)}^{I} + y_{mls}^{I} - 1, \qquad m=1,\ldots,M, k,l\in\theta_m, s=2,\ldots,N; \tag{8.6}$$

$$z_{mkls}^{I} \geq \sum_{j\in\omega_{ml}} y_{mj(s-1)}^{II} + y_{mls}^{I} - 1, \qquad \begin{array}{l} m=1,\ldots,M,\; k,l\in\theta_m, \\ t=2,\ldots,(T-1), s=P_t; \end{array} \tag{8.7}$$

$$\sum_{k\in\theta_m}\sum_{l\in\theta_m} z^I_{mkls} \le 1, \qquad m = 1,,\ldots,M, s = 1,\ldots,N; \tag{8.8}$$

$$\sum_{k\in\theta_m} z^I_{mkl1} \ge y^I_{ml1}, \qquad m = 1,\ldots,M, l \in \theta_m\ ; \tag{8.9}$$

Level II (bottling)

$$I^+_{j(t-1)} + I^-_{jt} + \sum_{m\in\rho_j}\sum_{s\in S_t} x^{II}_{mjs} = I^+_{jt} + I^-_{j(t-1)} + d_{jt}, \qquad \begin{array}{l} j = 1,\ldots,J, \\ t = 1,\ldots,T \end{array} \tag{8.10}$$

$$\sum_{j\in\delta_m}\sum_{\mathrm{s}\in\mathrm{S_t}} a^{II}_j x^{II}_{mjs} + \sum_{i\in\delta_m}\sum_{j\in\delta_m}\sum_{\mathrm{s}\in\mathrm{S_t}} b^{II}_{ij} z^{II}_{mijs} + \sum_{s\in S_t} v^{II}_{ms} \le K^{II}_{mt}, \qquad m = 1,\ldots,M, t = 1,\ldots,T \tag{8.11}$$

$$v^{II}_{ms} \ge \sum_{k\in\theta_m}\sum_{l\in\theta_m} b^I_{kl} z^I_{mkls} - \sum_{i\in\delta_m}\sum_{j\in\delta_m} b^{II}_{ij} z^{II}_{mijs}, \qquad m = 1,\ldots,M, s = 1,\ldots,N; \tag{8.12}$$

$$x^{II}_{mjs} \le \tfrac{K^{II}_{mt}}{a^{II}_{mj}} y^{II}_{mjs}, \qquad \begin{array}{l} m = 1,\ldots,M, \\ j \in \delta_m, t = 1,\ldots,T, s \in S_t; \end{array} \tag{8.13}$$

$$\sum_{j\in\delta_m} y^{II}_{mjs} = 1, \qquad m = 1,\ldots,M, s = 1,\ldots,N; \tag{8.14}$$

$$z^{II}_{mijs} \ge y^{II}_{mi(s-1)} + y^{II}_{mjs} - 1, \qquad \begin{array}{l} m = 1,\ldots,M, i, j \in \delta_m, \\ s = 2,\ldots,N \end{array} \tag{8.15}$$

$$\sum_{i\in\delta_m}\sum_{j\in\delta_m} z^{II}_{mijs} \le 1, \qquad m = 1,\ldots,M, s = 1,\ldots,N; \tag{8.16}$$

$$\sum_{i\in\delta_m} z^{II}_{mij1} \ge y^{II}_{mj1}, \qquad m = 1,\ldots,M, j \in \delta_m; \tag{8.17}$$

$$\begin{array}{l} I^+_{jt}, I^-_{jt} \ge 0, \qquad j = 1,\ldots,J, t = 1,\ldots,T, \\ z^I_{mkls}, v_{ms}, x^{II}_{mjs}, z^{II}_{mijs} \ge 0, \quad y^I_{mls}, y^{II}_{mjs} \in \{0,1\} \\ m = 1,\ldots,M, \ \ i, j \in \delta_m, \ \ k, l \in \theta_m, \quad s = 1,\ldots,N. \end{array} \tag{8.18}$$

The objective function 8.2 is to minimize the total sum of inventory costs, backorder costs, machine and tank changeover costs. In Level I, the demand for liquid flavor l is computed in terms of the production variables. That is, the demand for liquid l in each tank m in each micro-period s is given by and $\sum_{j \in \omega_{ml}} r_{lj} x^{II}_{mjs}$. The constraints (8.3), similar to constraints (8.13) in Level II, together with constraints (8.4) guarantees that if tank m is setup for syrup l in micro-period s ($y^{I}_{mls} = 1$) there will be production of liquid flavor l (between the minimum quantity necessary for liquid homogeny and the tank maximum capacity). The constraints (8.5) force the idle micro-periods to happen at the end of the associated macro-period. Constraints (8.6), similar to constraints (8.15) in Level II, control the liquid flavor changeover. Note that the tank setup does not hold from one macro period to another if the setup variable in the last micro-period is zero. Therefore, constraints (8.7) are needed to count the changeover between macro-periods. Note also that the setup variables in level II indicate which liquid flavor was prepared in the last non-idle micro-period of each macro-period. Constraints (8.8), similar to constraints (8.16) in Level II, count the first changeover of each tank. Constraints (8.9), similar to constraints (8.17) in Level II, guarantee that there is at most one changeover in each tank m in each micro-period s.

In Level II, constraints (8.10) represent the flow conservation constraints for each item in each macro-period. Since the production variable is defined for each micro-period, to obtain the total production of item j in a given macro-period t it is necessary to sum the associated production variables over all machines where it can be produced ($m \in \delta_j$) and micro-periods ($s \in S_t$) of macro-period t. Constraints (8.11) represent the machine capacity in each macro-period. Note here the inclusion of the waiting variable, v^{II}_{ms}, to ensure that the lot schedule will be feasible. The waiting time in machine m in each micro-period s is computed by constraints (8.12) as explained above. Constraints (8.13) guarantee that there is a production of item j only if the associated setup variable is set to one, and constraints (8.14) and (8.15) count the changeover in each machine m in each micro-period s. Constraints (8.14) refer to a single mode production in each micro-period s. Note that production may not occur although the machine is always ready to produce an item.

Finally, constraints (8.18) define the non-negativity and integrality restrictions. Note that the changeover variables z^{I}_{mkls} and z^{II}_{mijs} are continuous. Constraints (8.5), (8.6), (8.14), (8.15), and the optimization sense (minimization) ensure that these variables will take only 0 or 1 values.

As happened to the model presented in [19], the solution of practical instances of model P2LMM using the exact methods included in standard software such as CPLEX [45] was not satisfactory. This indicated the need to develop specific solution strategies.

8.3.2 Relax and fix strategies

The relax and fix heuristic has been largely used as a method to obtain good primal bounds (feasible solutions) for hard mixed-integer programs either on its own or in hybrid algorithms e.g. [47]- [50]. In this approach, first the integer variable set is partitioned into P disjunctive sets Q_i, $i = 1, \ldots, P$. At iteration n, the variables of Q_n are defined as integers while all others are relaxed. The resulting problem is then solved. If it is infeasible, the procedure finishes since it is not possible to find a feasible solution with the variables in Q_i, $i = 1, \ldots, n-1$ fixed at their actual values. Otherwise the variables of Q_n are fixed at their current values and the process is repeated for all the P sets. Besides the variable set partition, criteria to fix the variables in set Q_n must also be defined before applying the procedure. The main feature of this heuristic is the solution of submodels that are smaller, and possibly easier, than the original one. The partition of variables and the criteria used to fix the variables have a strong connection with the degree of the submodel difficulty.

In the usual relax and fix strategy the variables are grouped by periods (macro-periods) and only the integer variables are fixed at each iteration. In [47] these criteria were used in the solution of a multi machine multi items lot sizing model. The heuristics iterations number is thus the number of periods. In [48] this heuristic is applied to a class of project scheduling problems and explores various strategies to partition the set of binary variables. The relax and fix heuristic has also been used in combination with meta-heuristics such as Tabu Search. In [49] a hybrid tabu search procedure in which the relax and fix heuristic is used either to initialize a solution or to complete partial solutions is presented. The hybrid approach is applied to solve a big bucket lot sizing problem with setup and backlog costs. At each iteration of the relax and fix heuristic only the variables of a given period that concern a single product is fixed. The strategy is called relax-and-fix-one-product. The main advantage of this strategy is to solve smaller submodels since some mono-period, mono-machine multi-items lot sizing problems are hard to solve. In [1] a relax and fix heuristic to solve the SITLSP model formulated in [19] is proposed. The criterion used is to first fix the binary variables in Level I, then the ones in Level II, in a backward fashion. That is from the last period to the first. Other relax and fix strategies also fix continuous variables. In [50] the relax and fix heuristic is classified as a particular case of a progressive interval heuristics. They also mention that fixing continuous variables reduce the flexibility of the heuristic and propose various strategies varying the number of continuous variables fixed.

The P2LMM model presents various possibilities to build sets Q_i, $i = 1, \ldots, P$ [29]. The setup and changeover variables are indexed by levels, machines, items and periods. These indexes are explored when defining various variable partition strategies. Different criteria were proposed to fix variables. For example, after solving a submodel in a given iteration, it is possible to only

fix the binary variables associated to non-zero production variables. Table 8.1 shows 12 relax and fix strategies, divided into two groups. The Group 1 has five strategies (G1.1 - G1.5) and the Group 2 has seven (G2.1 - G2.7). The first column shows the strategy name (Strat.), the second and third columns show the criteria used for the partition (Part.) and fixing (Fix) the variables, respectively. The variables presented in Table 8.1 are the same ones used to describe the model P2LMM, however some of its indexes were omitted.

Table 8.1: Relax and Fix Strategies.

Strat.	Part.	Fix
G1.1	Period	y^I , y^{II}
G1.2	Period	y^I, z^I, y^{II},z^{II}
G1.3	Period	y^I, z^I, y^{II}, z^{II}, x^{II}
G1.4	Period	y^I, z^I, y^{II}, z^{II} i.th.p.*
G1.5	Period	y^I, z^I, y^{II}, z^{II} i.th.p. with reevaluation
G2.1	Machine/Period	y^I, y^{II}
G2.2	Level I then Level II	y^I, y^{II}
G2.3	Level II then Level I	y^I, y^{II}
G2.4	Period/ Level I then Level II	y^I, y^{II}
G2.5	Period/ Level II then Level I	y^I, y^{II}
G2.6	Machine/Period/ Level II and Machine/Period/ Level I	y^I, y^{II}
G2.7	Machine/Period/ Level II and Machine/Period/ Level I	y^I, z^I, y^{II}, z^{II} i.th.p. with reevaluation

The first five strategies (G1.1 - G1.5) use the usual criteria of partitioning the variables according to periods. They differ from each other by the criteria used to fix the variables in a given iteration. These criteria are based on the idea that the submodels should have a dimension that favors the decision process. The objective was to evaluate the influence of the variables in the submodels solution.

The Group 2 strategies explore the multi-machine, two level structure of the model to partition the set of variables. The objective was to evaluate the influence of each machine (level) in the solution of the submodels. Note that the criterion used to fix the variables in this group is the same one used in the strategy G1.1, except for the strategy G2.7. In this criterion, when the variable y^{II} is fixed to one, it only assures that the machine will be prepared, it does not say if the item will be produced or not. The variable y^I besides defining that the tank will be prepared, also states that there will be production of an item, between the tank capacities (constraints (2) and (3) in the P2LMM model). In the strategy G2.7, besides the binary variables, the continuous

variables (z^I, z^{II}) are also considered to be fixed when there is production ($x^{II} > 0$). In this strategy the idle micro-periods in previous iterations which did not have any variables fixed are also reevaluated for further variable fixing.

At each iteration of the relax and fix heuristic an instance of a mixed integer optimization submodel has to be solved. In general, they are solved by exact methods included in standard software (e.g. the branch and cut method in CPLEX). Although the submodels in each iteration are smaller than the original model, they are still difficult. If the optimal solution of the submodel solved at each iteration is not achieved in a pre-defined amount of time, the branch and cut execution is halted and the best solution is used to fix the variables.

8.3.3 The relaxation approach

In some industries the liquid preparation in Level I does not represent a bottleneck for the production process. That is, the tank capacities are large enough to ensure that whenever a machine needs a liquid of a given flavor it will be ready to be released. Therefore, there is no need to control either the changeover in the tanks or the synchrony between the two production levels, only the minimum tank capacity constraints to ensure the liquid homogeny is necessary in Level I. This situation was explored as a solution approach to model P2LMM. The Relaxation Approach (RA) is based on the idea that once the production decision is taken in Level II, the decision for Level I is easily taken.

In [44] a one level model to the production planning of a small soft drink manufacturer that has only one machine and several tanks is presented. To extend their model to the multi-machines case, only constraints (8.3), (8.4), (8.10), (8.12)-(8.18) has to be considered, dropping the changeover variables in Level I, z^I_{mkls}, and adding the constraint:

$$\sum_{j\in\delta_m}\sum_{\mathrm{s}\in\mathrm{S_t}} a^{II}_j x^{II}_{mjs} + \sum_{i\in\delta_m}\sum_{j\in\delta_m}\sum_{\mathrm{s}\in\mathrm{S_t}} b^{II}_{ij} z^{II}_{mijs} \leq K^{II}_{mt}, \quad m = 1,\ldots,M, t = 1,\ldots,T \tag{8.19}$$

The objective function is:

$$Min \sum_{j=1}^{J}\sum_{t=1}^{T}(h_j\, I^+_{jt} + g_j\, I^-_{jt}) + \sum_{m=1}^{M}\sum_{s=1}^{N}\sum_{i\in\delta_m}\sum_{j\in\delta_m} s^{II}_{ij}\, z^{II}_{mijs} \tag{8.20}$$

These sets of constraints and the objective function 8.20 define a one level multi-machines lot scheduling (P1LMM) model. It can be used in the first phase of the RA algorithm (described below) to define the lot sizing and scheduling of items in Level II. An adjustment of the solution of this model

might be necessary to take into account the synchrony between the two levels. This can be obtained by model P2LMM with the setup variables fixed according to the solution of model P1LMM. That is, if item j is produced then the tank and the machine must be setup. This procedure is outlined in Algorithm 8.5, where σ_j is the liquid flavor necessary to produce item j.

Algorithm 8.5: The RA Algorithm.

Step 1 Solve the P1LMM model;
Step 2 **if** *P1LMM is feasible* **then**
  for *m=1* **to** *M;* $j \in \delta_m$,$s = 1, \ldots, N$ **do**
    if $x^{II}_{mjs} > 0$ **then**
      Fix the setup variables of P2LMM model according to;
      for $j \in \delta_m$, $l \in \sigma_j$ **do**
        $y^{II}_{mjs} = 1$ and $y^{I}_{mls} = 1$
      end
    end
  end
end
Step 3 Solve the P2LMM model obtained in Step 2.

The relax and fix strategies presented in Section 8.3.2 can also be used to solve the models in Steps 1 and 3 of Algorithm RA. Note however that in model P1LMM in Level I only the capacities constraints are considered and there are no changeover variables associated to this level. Therefore the relax and fix strategies described in Table 8.1 have to be modified accordingly. If the models in Algorithm RA are solved by standard software and their optimal solution are not achieved in a pre-defined amount of time, the branch and cut execution is also halted and the best solution is considered. Other solution approaches for SITLSP are presented and tested in [44] and [51].

8.4 Computational Tests

In this section we present and analyze the computational results of the evolutionary algorithms (MA and GA) described in Section 8.2, and the decomposition and relaxation approaches described in Section 8.3, when applied to solve industrial instances of SITLSP. Among all approaches reported in Section 8.3, we present the results of only P2LMM_G2.7 (relax-and fix strategy G2.7 applied to model P2LMM) and RA_G2.1 (relax-and-fix strategy G2.1 applied to the model P1LMM of Algorithm RA), since they were the ones that produced the best solutions for this set of instances tested. Other computational tests with randomly generated examples of small-to-moderate and moderate-to-large sizes and other industrial instances were performed with the MA and the GA approaches presented in Section 8.2. The P2EMM_G2.7, RA_G2.1 and the other approaches of Section 8.3 are as well tested in other examples. Their results are reported in [23] and [29], respectively.

8.4.1 Generation of instances

The solution methods presented in Sections 8.2 and 8.3 can be easily adapted to represent the particularities of different soft drinks companies. To follow, we describe next the necessary adjustments to represent the situation encountered at a Brazilian soft drink manufacturer, Plant A, as well as the data used to generate the instances used in the computational tests described in Section 8.4.2.

Plant A produces many types of soft drinks (items) characterized by the liquid flavor and bottle type. It has various tanks and filling lines (machines) with different capacities. Some tanks (machines) are dedicated to produce only a given subset of flavors (items), whereas others can produce any one. There is a single liquid flavor ($l = 4$) whose demand is by far superior to the others. While most of the liquid flavors have demands around 20,000 units per period, flavor 4 has demand around 150,000 units. It also has a high setup time and cost. Therefore, in Plant A there is a tank which is fully dedicated to continuously produce flavor 4. That is, whenever a machine is ready to produce an item that uses this flavor, the tank is also ready to release it.

To represent this situation in model P2LMM of Section 8.3, the changeover time in Level I from any flavor k to flavor 4 and from flavor 4 to any other is set to zero ($b^I_{k4} = b^I_{4k} = 0$). Similar adjustments are made in the evolutionary algorithms of Section 8.2. Although there is no need to setup the tanks for flavor 4, there is still a need to setup the machines when items that need this flavor are produced. However, the tank setup variable, y^I_{m4s} cannot be set to zero, since when this is done, constraints (8.3) together with (8.4) impose that the production of any item that uses this flavor is zero (if $y^I_{m4s} = 0$ then $x^{II}_{mjs} = 0$, for $j \in \omega_{m4}$). Therefore the tank capacity (constraints (8.3) and (8.4)) for $l = 4$ is dropped.

When defining the lot size and schedule, Plant A also considers that the inventory in a given period must be enough to cover the demand in the next period. To have a fair comparison between the model and the company solutions, a new set of constraints should be included in the model:

$$I^+_{jt} = d_{j(t+1)} \quad j = 1, \ldots, J; t = 2, \ldots, T+1; \tag{8.21}$$

Two minor modifications were made in the MA and GA to ensure that their solutions are comparable to the solutions of P2LMM_G2.7 and RA_G2.1, as well as the solutions used by Plant A. The first change refers to the assumption that each filling line has a dedicated tank, that is, there is only one tank allocated to only one line. Therefore, the criterion of the MA/GA for selecting a tank, instead of determined by (Section 8.2):

$$j = \begin{cases} \beta_i, & 1 \leq \beta_i \leq \overline{L}; \\ \beta_i - \overline{L}, & \overline{L} < \beta_i \leq 2\overline{L}. \end{cases} \tag{8.22}$$

It was re-defined as: $j = \beta_i$, where $\beta_i = \alpha_i$. In addition, the fitness function of the MA/GA was modified to include only the product inventory, product changeover and demand shortage costs. As the company primarily favors production schedules with no demand shortages, the shortage unit cost was considered as a sufficiently large penalization so as to avoid shortages in the solutions. In cases whereby the MA and GA were unable to find a schedule without shortages, their solution was simply considered infeasible. The same was considered with the solutions of approaches P2LMM_G2.7 and RA_G2.1. We refer to these modified versions of MA and GA as MA1 and GA1, respectively.

The second change in the MA/GA is the consideration of penalization costs over the liquid flavor inventories maintained in the tanks from one period to another. In other words, besides the costs taken into account in the fitness function of the MA1 and GA1, we also considered penalization costs to avoid holding liquid flavor inventories between consecutive periods. This is because the company prefers production schedules that do not hold liquid inventories from one period to other, because they are perishable. Recall from Section 8.3 that approaches P2LMM_G2.7 and RA_G2.1 do not hold liquid inventories. We refer to these modified versions of the MA and GA as MA2 and GA2, respectively.

Several visits to Plant A were made in order to understand its production processes and to collect the data necessary to simulate their SITLSP. Data associated to demands, changeover times in both levels, tank and machine capacities, etc., were obtained during these visits. The details of the collected data are presented in [29, 51]. The data was used to generate 15 instances of the SITLSP.

The first instance (P1) was generated based on data related to two machines that can produce items in common. The first one (machine 1) can produce 23 items and the second one (machine 2) only 10 out of these. That is, there are 13 items that can be produced on any one of these two machines. Eighteen different flavors are necessary to produce this set of items.

Three weeks were considered in the planning horizon. Machine 1 was available for four working days per week (total of 5,760 minutes per week) and machine 2 six working days (total of 8,640 minutes per week). It was estimated that the tank could have up to five changeovers per day. Taking the average of the number machine working days (5 days), it is possible to have up to 25 changeovers per week. Therefore, the P1 instance has three macro-periods (3 weeks) with a total of 75 micro-periods (25 per macro-period). The production scheduling for this instance was provided by Plant A, making the comparison with the strategies proposed in Sections 8.2 and 8.3 possible.

To simulate different scenarios, four other instances (P2-P5) were generated by modifying part of the data used in instance P1. The inventory costs were doubled (P2), the changeover costs halved (P3), the total demand was redistributed among the periods (P4), and the machines capacities were reduced (P5). These modifications are detailed in Table 8.2.

Demand data related to a period of 30 weeks was also available. This data allowed the generation of another group of 10 instances (P6-P15). Each one of these instances is associated to three consecutive weeks. Instances P6, P7, P14 and P15 are associated to periods of higher demands when compared to P8-P13. Except for the demands, all the other parameters used to generate these instances were the same ones used to generate instance P1. More details of the procedure used to generate the instances can be found in [29, 51].

Table 8.2: Modifications in instance P1 to generate P2-P5.

Instances	Modification
P1	Plant A data
P2	Inventory costs of P1 doubled
P3	Inventory costs of P1 doubled and the changeover costs of P1 halved.
P4	The total demand of P1 was randomly redistributed among the periods.
P5	The machines capacities were reduced

8.4.2 Computational results

The experiments described in this section ran on a microcomputer Pentium IV with 1 GB Ram and 2.8 GHz. The approaches P2LMM_G2.7 and RA_G2.1 were implemented using the modelling language AMPL [52] and the optimization solver CPLEX 10.0 [45] with default parameters, while the MA and the GA were implemented using the NP-Opt [28,37], an object-oriented framework written in JAVA code which contains procedures based on evolutionary computation techniques to address NP-hard problems. An execution time limit of 4 hours was established to solve each example by each method. The MA and the GA were applied three times to each example and the best solution obtained was chosen. In order to satisfy the total time limit of 4 hours, a limit of 4800 seconds was also imposed to each MA or GA run. It should be mentioned that the 4-hour limit is acceptable to support the decisions involved in the production scheduling of the company.

The MA and GA have been adjusted with 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 leading the methods to execute 19 crossovers over each population. The mutation rate was fixed at 0.7 where a γ value is randomly chosen in $[0, 1]$ with uniform distribution. The mutation operator is executed on the new individual, if $\gamma < 0.7$. All these values are based on previous tests which are reported in [1].

Table 8.3 presents the total cost values (in thousands of monetary units) of the solutions obtained by the MA1, GA1, MA2 and GA2, in comparison with the solutions of P2LMM_G2.7 and RA_G2.1, for the instances P1-P15. As

mentioned, for each example, the values of the memetic and genetic algorithms correspond to the lowest cost solution found among the three executions of the algorithm. Note that the values of the table correspond only to inventory and changeover costs, since there are no demand shortages. The symbol "inf." in the table indicates that the method was unable to find a feasible solution (i.e., a solution without demand shortages) within the time limit. The best solution value of each example is highlighted in bold.

Table 8.3: Solution values of methods P2LMM_G2.7, RA_G2.1, MA1, GA1, MA2 and GA2 for instances P1-P15.

Ex.	P2LMM_G2.7	RA_G2.1	MA1	GA1	MA2	GA2
P1	399.3	322.3	256.8	285.6	**244.6**	271.4
P2	509.6	325.9	306.7	280.3	287.5	**268.4**
P3	270.3	212.9	146.6	148	**141.8**	158.8
P4	480,3	330.5	217.6	221.5	214.4	**208.1**
P5	inf.	inf.	inf.	inf.	inf.	inf.
P6	inf.	inf.	inf.	inf.	inf.	inf.
P7	inf.	inf.	inf.	inf.	inf.	inf.
P8	inf.	**529**	inf.	inf.	inf.	inf.
P9	560.9	**261.7**	inf.	372.5	348.7	391.6
P10	744.4	**266.2**	303.9	317.1	358.3	375.4
P11	461.6	**294**	inf.	inf.	inf.	inf.
P12	inf.	**344.8**	inf.	inf.	inf.	inf.
P13	inf.	**358.5**	405.7	422.4	393	483.6
P14	inf.	inf.	inf.	inf.	inf.	inf.
P15	inf.	inf.	inf.	inf.	inf.	inf.

"inf." - No feasible solution found within the time limit

As discussed in Section 8.4.1, instance P1 is the only one of the problem set for which we are aware of the corresponding production schedule used by Plant A. This company solution meets all product demands with no delays, yielding a total cost of 422.7 in thousands of monetary units. Comparing this solution to the ones presented in Table 8.3, we notice that all methods P2LMM_G.7, RA_G2.1, MA1, GA1, MA2 and GA2 were able to find better solutions than the company, with relative cost reductions of 5.5%, 23.8%, 39.2%, 32.4%, 42.1% and 35.8%, respectively. This indicates that these approaches have a potential to generate competitive solutions - note that the cost reductions can be significant. Moreover, it can be observed that the memetic versions outperformed the genetic ones.

In instances P1 and P2-P4 (which are based on instance P1), the solutions obtained by MA1, GA1, MA2 and GA2 are better than the ones obtained by P2LMM_G.7 and RA_G2.1 (Table 8.3). For these instances, the P2LMM model involved in approach P2LMM_G.7 has 86,359 variables (4,575

binary variables) and 86,140 constraints, while the P1LMM model in approach RA_G2.1 has 54,559 variables (4,575 binary variables) and 49,544 constraints. Some performance variations between the MA1 and MA2 (and between the GA1 and GA2) are due to the randomly generated initial populations of the algorithms. In particular, we do not know if instance P5 is feasible (from the point of view of demand shortages), since none of the methods found a solution without shortages. This instance was generated using the same data as P1, except for the reduction in the machine capacities (Table 8.2).

The remaining instances P6-P15 refer to collected data of the product demands for different months, provided by Plant A. Since we do not have the production schedules used by the company for these examples, we are not aware if they are feasible from the point of view of the demand shortages. In these examples, approach RA_G2.1 produced better solutions than MA1, GA1, MA2 and GA2. Moreover, for all instances P1-P15, the RA_G2.1 solutions dominate the ones of P2LMM_G.7 (Table 8.3). As well as for instance P5, we do not know if instances P6, P7, P14 and P15 are feasible, since none of the methods found a solution without shortages. The minimum capacity necessary to produce all the items in these instances is higher than in the others. However, it is worth mentioning that by using realistic backlogging or lost sales unit costs (instead of large penalties) in these methods, they can be employed to generate effective schedules balancing the trade-offs between the inventory, changeover and shortage costs.

Table 8.4 resumes the relative deviations of the solution values of MA1, GA1, MA2 and GA2 regarding approach RA_G2.1. These deviations were calculated using the expression: $Dev(\%) = 100(z-\overline{z})/\overline{z}$, where z is the solution value of MA1 (or GA1, MA2, GA2) and $\overline{z}$ is the value found by approach RA_G2.1. Note that the deviations of the evolutionary algorithms with respect to RA_G2.1 are relatively large, varying from -37.0% to 49.6%. The superiority of the memetic approach over its genetic version is also corroborated by these results.

Given that the methods were unable to find a solution with no demand shortages for instances P5, P6, P7, P14 and P15, we roughly approximate the backlogging unit costs as the profit contributions of the products and we applied the methods again to solve the examples using these parameters as the shortage unit costs. Table 8.5 presents the modified total cost values (in thousands of monetary units) of these experiments - note that, unlike Table 8.3, these values correspond to inventory, changeover and shortage costs. The best solution found for each example alternates between the methods RA_G2.1, MA1, GA1 and MA2, and these methods do not dominate each other. Table 8.6 depicts the relative deviations of the evolutionary approaches with respect to method RA_G2.1 for the examples with demand shortages.

Considering the results in Tables 8.3 and 8.5, we note that MA1 (or MA2) outperforms GA1 (or GA2) in 8 out of the 12 examples for which

Table 8.4: Relative deviations of the solution values of MA1, GA1, MA2 and GA2 with respect to approach RA_G2.1.

Ex.	MA1	GA1	MA2	GA2
P1	-20.3	-11.4	-24.1	-15.8
P2	-5.9	-14	-11.8	-17.6
P3	-31.1	-30.5	-33.4	-25.4
P4	-34.1	-33	-35.1	-37
P8	***	***	***	***
P9	***	42.3	33.2	49.6
P10	14.2	19.1	34.6	41
P11	***	***	***	***
P12	***	***	***	***
P13	13.1	17.8	9.6	34.9

*** The deviation was not computed because the approaches MA1, GA1, MA2 or GA2 did not find a feasible solution within the time limit

Table 8.5: Modified solution values of methods P2LMM_G2.7, RA_G2.1, MA1, GA1, MA2 and GA2 with demand shortages.

Ex.	P2LMM_G2.7	RA_G2.1	MA1	GA1	MA2	GA2
P5	603.7	379.5	**371.7**	422.2	485.7	393.7
P6	663.9	526.5	565.4	**492.8**	527	567.1
P7	591.5	509.5	**370.4**	398.1	372.8	413.9
P14	588.5	449.5	357.6	343.3	**310.4**	378.5
P15	671.3	**446.2**	476.6	541.1	555.4	491.2

Table 8.6: Relative deviations of the solution values of MA1, GA1, MA2 and GA2 with respect to approach RA_G2.1 (infeasible examples).

Ex.	MA1	GA1	MA2	GA2
P5	**-2**	11.2	27.9	3.7
P6	7.3	**-6.4**	0.1	7.7
P7	**-27.3**	-21.9	-26.8	-18.8
P14	-20.4	-23.6	**-30.9**	-15.8
P15	6.8	21.3	24.5	10.1

these algorithms found a feasible solution, indicating a better performance of MA over GA in these instances. A similar result was observed in the other instances randomly generated and tested in [1]. Moreover, comparing MA2 and RA_G2.1, we note that MA2 outperforms RA_G2.1 in 6 examples and RA_G2.1 outperforms MA2 in 9 examples, showing that these methods are

competitive. The GA2 and MA2 approaches provide a better solution in scenarios where the total capacity is loose and the RA_G2.1 in scenarios where the total capacity is tight.

It is worth remarking that, for all instances P1-P15, the gap between the best feasible solutions of model P2LMM (found within the time limit) and its linear relaxation solution is over 90%, which does not provide much information about the optimality gap of the solutions in Tables 8.3 and 8.5.

8.5 Final Remarks and Conclusions

In this chapter we study a two-level production planning problem involving, on each level, a lot sizing and scheduling problem with parallel machines, capacity constraints and sequence-dependent setup costs and times. The problem is referred to as the Synchronized and Integrated Two-Level Lot sizing and Scheduling Problem (SITLSP) and it can be found in some industrial settings, as, for example, in soft drink companies where the production process involves two interdependent levels with decisions concerning raw material storage and soft drink bottling.

A mixed integer programming model for the SITLSP was introduced in [19], integrating and synchronizing the two lot sizing and scheduling problems involved. This model can be useful when dealing with small-to-moderate size instances, however, as the problem size grows, the number of distinct integer solutions increases exponentially, causing the branch-and-bound search to take too long, even for finding the first integer feasible solution.

In order to overcome these limitations, and deal with larger and more realistic problem instances, a genetic algorithm with a particular representation of solutions for individuals and a hierarchically structured multi-population is proposed in [23]. In Section 8.2 the genetic approach is extended by a memetic algorithm version. A tailor-made decoding procedure is used to evaluate the solution encoded in the gene of each individual. Moreover, a tailor-made recombination over population clusters takes place and migrations among different populations are allowed. The memetic approach is capable of generating competitive solutions if compared with the ones utilized in practice, as shown in Section 8.4.

As an alternative for these meta-heuristic approaches, another solution method capable of dealing with realistic problem instances of the SITLSP is presented (Section 8.3). Unlike the highly general and complex MIP model in [19], a simplified MIP formulation, in the sense that it forces the number of tanks to be the same as the filling lines so that each line has a dedicated tank, is described. The solution approach of the simplified model relies on different relax-and-fix heuristics and model decomposition strategies.

The performances of the memetic algorithm-MA (as well as its genetic version- GA) and the decomposition/relaxation approaches (P2LMM_G2.7 and RA_G2.1) are evaluated solving a set of instances based on actual data

provided by a Brazilian soft drink company. All approaches are capable of producing competitive solutions when compared to the production schedule used by the company, requiring affordable computational runtime. In some cases the cost savings of the solutions are substantial (Section 8.4). Comparing the relative performances of the approaches when solving this problem set, it is observed that the MA outperforms the GA and that the MA and the RA_G2.1 do not dominate each other. In particular, the MA provides a better solution in scenarios where the total capacity is loose and the RA_G2.1 in scenarios where the total capacity is tight.

Topics for future research include new experiments with the methods studied in this chapter involving real instances to be obtained from other soft drinks plants. The evolutionary approaches can be improved by introducing new genetic operators, especially other crossovers not tested yet and other more powerful local search-based algorithms such as Simulated Annealing or Tabu Search. The introduction of valid inequalities is a research topic to be attempted to improve the decomposition/relaxation approaches.

As point out in the Introduction, the main purpose of this chapter was bring to the attention of the academic community and also to the industrial engineering practitioners the state-of-the art of computer-aided tools used to generate effective production schedules in the soft drink industry. Focusing the SITLSP, a hard combinatorial optimization problem, we believe that the most important conclusions that can be extracted from the studied methods as well as from the computational assessment performed in this chapter are two-fold: first, mathematical modelling provides deeper insight into a very complex industrial planning problem, and, second, tailor-made optimization methods can yield less costly and more effective production schedules in reasonable computing times when compared with the ones provided by general purpose commercial packages commonly used by the industry.

Acknowledgements

We are indebted to Flávio Veronese and Aldo Fernandes Júnior from Companhia de Bebidas Ipiranga, Brazil for their kind collaboration. This research was partially supported by grants from FAPESP and CNPq.

References

1. Toledo C. F. M. (2005) The integrated two-stage lot sizing and scheduling problem, Doctoral Thesis (in Portuguese), State University of Campinas, Brazil
2. Drexl A., Kimms A. (1997) Lot sizing and scheduling - survey and extensions, European Journal of Operational Research 99: 221-235.
3. Bitran G. R., Yanasse H.H. (1982) Computational complexity of the capacited lot size problem, Management Science 28(10): 1174-1186.

4. Clark A. R., Clark S. J. (2000) Rolling-horizon lot sizing when set-up times are sequence-dependent, International Journal of Production Research 38(10): 2287-2307.
5. Fleischmann B. (1994) The discrete lot sizing and scheduling problem with sequence-dependent setup costs, European Journal of Operational Research, 75: 395-404.
6. Gupta D., Magnusson T. (2005) The capacitated lot sizing and scheduling problem with sequence-dependent setup costs and setup times, Computers & Operations Research 32: 727-747.
7. Haase K., Kimms A. (2000) Lot sizing and scheduling with sequence dependent setup costs and times and efficient rescheduling opportunities, International Journal of Production Economics 66: 159-169.
8. Meyr H. (2000) Simultaneous lot sizing and scheduling by combining local search with dual reoptimization, European Journal of Operational Research120: 311-326.
9. Berreta R. E., França P. M., Armentano V. (2005) Meta-heuristic approaches for the multilevel resource-constrained lot sizing problem with setup and lead times, Asia-Pacific Journal of Operational Research 22(2): 261-286.
10. França P. M., Armentano V., Berretta R. E., Clark, A. R. (1997) A heuristic method for lot sizing in multi-stage systems, Computers & Operations Research 24 (9): 861-874.
11. Kimms A. (1997) Demand shuffle - A method for multi-level proportional lot sizing and scheduling, Naval Research Logistics 44: 319-340.
12. Kimms A. (1997) Multi-level lot sizing and scheduling - Methods for capacitated, dynamic, and deterministic models, Physica, Heidelberg.
13. Özdamar L., Barbarosoglu G. (2000) An integrated lagragean relaxation-simulated annealing approach to the multi-level multi-item capacitated lot sizing problem, International Journal of Production Economics 68: 319-331.
14. Kang S., Malik K., Thomas L. J. (1999) Lot sizing and scheduling on parallel machines with sequence-dependent setup costs, Management Science 45: 273-289.
15. Kuhn H., Quadt, D. (2002) Lot sizing and scheduling in semiconductor assembly - A hierarchical planning approach. In: Mackulak G. T., Fowler J. W., Schömig A. (eds), Proceedings of the International Conference on Modeling and Analysis of Semiconductor Manufacturing, Tempe, USA: 211-216.
16. Meyr H. (2002) Simultaneous lot sizing and scheduling on parallel machines, European Journal of Operational Research 139: 277-292.
17. Quadt D., Kuhn H. (2003) Production planning in semiconductor assembly, Working Paper, Catholic University of Eichstätt-Ingolstadt.
18. Stadtler H. (2003) Multilevel lot sizing with setup times and multiple constrained resources: internally rolling schedules with lot sizing windows 51(3): 487-502.
19. Toledo C.F.M., Kimms A., França P.M, Morabito R.(2006) A mathematical model for the synchronized and integrated two-level lot sizing and scheduling problem, Journal of Operational Research Society: under review.
20. Bitran G. R., Matsuo H. (1986) Approximation formulations for the single product capacitated lot size problem, Operations Research 34: 63-74.
21. Fleischmann B., Meyr, H. (1997) The general lot sizing and scheduling problem, OR Spektrum 19: 11-21.

22. Drexl A., Haase K. (1997) Proportional lotsizing and scheduling, International Journal of Production Economics 40: 73-87.
23. Toledo C. F. M., França P. M., Morabito R., Kimms A. (2007) A multi-population genetic algorithm to solve the synchronized and integrated two-level lot sizing and scheduling problem, International Journal of Production Research: in press.
24. Holland J. H. (1975) Adaptation in natural and artificial systems, The University of Michigan Press.
25. Goldberg D. E. (1989) Genetic algorithms in search, optimization, and machine learning, Addison Wesley.
26. Michalewicz Z. (1996) Genetic Algorithms + data structure = evolution programs, Springer-Verlag.
27. Moscato P. (1989) On evolution, search, optimization, genetic algorithms, and martial arts: towards memetic algorithms, Technical Report, Caltech Concurrent Computation Program, C3P Report 826.
28. Mendes A. S. (2003) The framework NP-Opt and its applications to optimization problems, Doctoral Thesis (in Portuguese), State University of Campinas - Brazil.
29. Ferreira D., Rangel S., Morabito R. (2007) Solution approaches for the soft drink integrated lot sizing and scheduling problem, European Journal of Operational Research (under review).
30. Clark A. R. (2003) Hybrid heuristics for planning lot setups and sizes, Computers & Industrial Engineering 45: 545-562.
31. Moscato P. (1989) On genetic crossover operators for relative order preservation. C3P Report 778, California Institute of Technology, Pasadena, CA 91125.
32. Dawkins R. (1976) The selfish gene. Oxford University Press Oxford.
33. Gen M., Cheng R. (1997) Genetic algorithms & engineering design. John Wiley & Sons New York NY.
34. Goldberg D.E. (2002) The design of innovation: lessons from and for competent genetic algorithms. Addison-Wesley Reading, MA.
35. Hart W. E., Krasnogor N., Smith J.E. (Eds.) (2005) Recent advances in memetic algorithms series: studies in fuzziness and soft computing 166.
36. Weiner J. (1995) The beak of the finch. Vintage Books New York.
37. Mendes A. S., França P. M., Moscato P. (2001) NP-Opt: an optimization framework for NP problems. Proceedings of POM2001: 82-89 Guarujá, Brazil.
38. França P. M., Mendes A. S., Moscato P. (2001) A memetic algorithm for the total tardiness single machine scheduling problem. European Journal of Operational Research 1(132): 224-242.
39. Sikora R. (1996) A genetic algorithm for integrating lot sizing and sequencing in scheduling a capacitated flow line. Computers & Industrial Engineering 30(4): 969-981.
40. Özdamar L., Birbil S. I. (1998) Hybrid heuristic for the capacitated lot sizing and loading problem with setup times and overtime decisions. European Journal of Operational Research 110: 525-547.
41. Dellaert N., Jeunet J., Jonard N. (2000) A genetic algorithm to solve the general multi-level lot sizing problem with time-varying costs. International Journal of Production Economics 68: 241-257.
42. Kimms A. (1999) A genetic algorithm for multi-level, multi-machine lot sizing and scheduling. Computers & Operations Research 26: 829-848.

43. Dueck G., Scheuer T. (1990) Threshold accepting: A general purpose optimization algorithm appearing superior to simulated annealing. Journal of Computational Physics 90: 161-175.
44. Ferreira D., Morabito R., Rangel S. (2007) A MIP model and relax and fix heuristics for production planning and scheduling in a small soft drink plant (in Portuguese), Produção (São Paulo): in press.
45. ILOG (2001) Using the CPLEX Callable Library, Copyright, ILOG.
46. Wolsey L. A. (1998) Integer Programming, John Wiley & Sons.
47. Dillemberger C., Escudero L. F., Wu Zhang A. W. (1994) On practical resource allocation for production planning and scheduling with period overlapping setups, European Journal of Operational Research 75: 275-286.
48. Escudero L. F., Salmeron J. (2005) On a fix-and-relax framework for a class of project scheduling problems, Annals of Operations Research 140: 163-188.
49. Pedroso J. P., Kubo M. (2005) Hybrid tabu search for lot sizing problems, in Hybrid Meta-heuristics, Lecture Notes in Computer Science 3636: 66-77, Springer, Berlin.
50. Federgruen A., Meissner J., Tzur M. (2007) Progressive interval heuristics for multi-item capacitated lot sizing problems, Operations Research, 55 (3): 490-502.
51. Ferreira D. (2006) Approaches to the integrated problem of the lot sizing and scheduling of the soft drink production, Doctoral Thesis (in Portuguese), Federal University of São Carlos, Brazil.
52. Fourer R., Gay M. D., Kernighan B. W. (1993) AMPL - A modeling language for mathematical programming, The Scientific Press, Danvers, Massachusetts.