Hindawi Publishing Corporation
Journal of Applied Mathematics
Volume 2014, Article ID 964120, 14 pages
http://dx.doi.org/10.1155/2014/964120

Research Article

A Scheduling Problem in the Baking Industry

Felipe Augusto Moreira da Silva,[1] Antonio Carlos Moretti,[2] and Anibal Tavares de Azevedo[2]

[1] *Department of Applied Mathematics, Institute of Mathematics, Statistics and Scientific Computing, State University of Campinas, Campinas, SP, Brazil*
[2] *School of Applied Sciences, State University of Campinas, Limeira, SP, Brazil*

Correspondence should be addressed to Felipe Augusto Moreira da Silva; felipe_amsilva@yahoo.com.br

Received 5 December 2013; Revised 29 April 2014; Accepted 29 April 2014; Published 19 June 2014

Academic Editor: Nachamada Blamah

This paper addresses a scheduling problem in an actual industrial environment of a baking industry where production rates have been growing every year and the need for optimized planning becomes increasingly important in order to address all the features presented by the problem. This problem contains relevant aspects of production, such as parallel production, setup time, batch production, and delivery date. We will also consider several aspects pertaining to transportation, such as the transportation capacity with different vehicles and sales production with several customers. This approach studies an atypical problem compared to those that have already been studied in literature. In order to solve the problem, we suggest two approaches: using the greedy heuristic and the genetic algorithm, which will be compared to small problems with the optimal solution solved as an integer linear programming problem, and we will present results for a real example compared with its upper bounds. The work provides us with a new mathematical formulation of scheduling problem that is not based on traveling salesman problem. It considers delivery date and the profit maximization and not the makespan minimization. And it also provides an analysis of the algorithms runtime.

1. Introduction

Production planning problems have been studied extensively since the early twentieth century, and they can be found throughout literature. One of the pioneers in the work on these problems was Henry Gantt in his book "Work, Wages and Profit" [1] in which he demonstrates the need for a job schedule in order to increase production efficiency.

Currently many industries are seeking solutions for the job sequencing problem in order to increase productivity, reduce costs and, consequently, increase profit. Due to the burgeoning consumption of foodstuffs worldwide, optimized planning is necessary. For decades in the industry, planning rules were used in order to prioritize products by taking into account only a few production stages, but, given the increase of complexity and modernization of production, planning as a whole has to be optimized; that is, the entire production chain, from production and stock to shipping and sales, must be taken into account.

The purpose of this paper is to study planning problems applied to baking industry, where, in the current scenario, companies have high product turnover; that is, the goods are highly perishable and companies cannot meet all the demands for their products made by customers; and these decisions are highly correlated to production planning and transportation.

The difficulty of the problem in question is to define a production sequence in each line considering setup and production times, together with jobs that have set dates to be executed, in such a way as to maximize company profit. Thus, the problem has particular production characteristics, such as batch production and a scenario with several parallel production lines. Another relevant point to be considered in this study is that the solution should be found in a timely computational manner for planning purposes, as it is to be executed for the next 24 hours and cannot run for longer than minutes, allowing the implementation of planning in hours following the decision.

There are many variations of scheduling problems, many of them widely studied in literature, but the problem studied here is not found in literature with all of the features proposed herein; we can only find methods to solve part of the problem.

Because of the diversity of production planning problems, the problems can be divided into classes. The ratings found for the more general problem of sequencing are flow-shop and job-shop. For flow-shop each process is identified in one job and job-shop is the problem where the order of each process may not be the same in one job. This problem presented here is a flow-shop problem.

In studies of flow-shop problems, the first theoretical results were presented to minimize the makespan, that is, total production time, with Johnson [2] and Bellman [3] determining the optimal sequence for the cases with two machines and special cases with three machines. More general cases for sequencing with three machines came from Lomnicki [4] and Ignall, and Schrage [5] by applying branch and bound methods introduced by Little [6]. For the problem of flow-shop with more than two machines Garey et al. [7] proved that it is NP-hard; thus we can only solve small problems accurately with algorithms such as branch-bound. Therefore, heuristic methods are proposed for the problem.

In order to study the flow-shop problem with a family setup, Sridhar and Rajendran [8] propose a heuristic to minimize the total time with an algorithm based on simulated annealing. Ziegler [9] proposes a method to minimize the total time weighted by weights in the process. Schaller [10] presents a new approach to the problem of flow-shop setup with families to minimize the makespan. Schaller [11] proposes a new lower bound for the problem and implements a two-stage heuristic algorithm based on branch and bound.

França et al. [12] works on the same problem, but with the Schaller genetic algorithm with local search, "memetic algorithms" (MA), and [13] achieves superior results by using hybrid methods with tabu search and the genetic algorithm.

The problems with delivery dates were studied by Croce et al. [14], who showed a genetic algorithm in which each chromosome consists of m subchromosomes, one for each machine, which identifies each transaction made by the machine. According to the authors, the results were better than those found by Adams et al. [15], but at greater computational cost. Sittisathanchai and Dagli [16] also present a genetic algorithm where the chromosome represents the operational sequence.

Flexible scheduling involves problems where we have the option of executing operations on different machines. This kind of problem is more comprehensive than the problem of traditional scheduling and production in parallel, which ensures that a transaction is made only by a single machine. Arthanari and Ramaswamy [17] were pioneers with exact two-stage methods, using two identical parallel machines in the first stage and one machine in the second stage. Later, Brah and Hunsucker [18] worked on the development of more general branch and bound algorithms, but for problems with more than 8 jobs, 5 stages and 2 or 3 machines, processing time becomes impractical.

For multistage jobs, Sawik [19] proposed a constructive heuristic in which the route of a job is determined at each iteration. Kittichartphayak and Ding [20] developed a heuristic similar to Sawik [19], but for larger orders of tasks and stages, which was extended by Guinet and Solomon [21]. Smutnicki and Nowicki [22] propose the use of tabu search in the work with satisfactory results.

The studied problem is a flow-shop problem with parallel production, setup times, batch production, due date, and transportation capacity, which in large scale justify the use of metaheuristic to solve it.

2. Materials and Methods

2.1. Mathematical Modeling. The scheduling problem can be modeled as a mixed integer linear programming problem.

The problem requires a short-term study in great detail; that is, the study will be forecast to take place over two to seven days from the current day. Given the high level of time detail, it is necessary to deal with accuracies in minutes, as the setup time and production time may occur within minutes. Therefore, a discretization of time can result in inaccuracies because a large number of periods can cause a problem with a large number of variables, thus making it unenforceable, and a small number of time periods may not address the problem with the required accuracy.

Traditional approaches of the scheduling problem are based on the traveling salesman problem, where a variable represents the precedence order of task. But in this case it is necessary to know what time each product will be ready for shipment. Accordingly, a new model is proposed to represent the job sequencing where each variable represents which jobs are executed in each time period (Figure 1).

This problem is defined in a set of period E, where $E = \{0, \ldots, e\max\}$ and $e\max$ is the last period. Production is defined in two stages: in the first one, a preproduct known as mass is produced, and, in the second stage, a mass is transformed into a final product. The set of mass will be represented by M and the set of products determined by the set P. Each product $p \in P$ will be produced by a single mass that can produce more than one product defined by $\text{Prod}(m) \subset P$. Each mass has its production cost defined by $\text{cost}_1(m)$, which defines the cost of producing each product. The production time of each mass is represented by $t(m)$, and the setup time to prepare the mass of two products 1 and 2 is $s(m_1, m_2)$.

Production occurs through two processes. The first process works in a mixer, which mixes all the ingredients of a mass, and the cost of ingredients will determine the profits from the products made with this mass. After the mixer, there is a second process, where the mass will be roasted, sliced, and packaged. In this second case the mass will be transformed into the final product. Products can be grouped into classes due to of the production characteristics, such as cooking time, types of cuts, and packaging. Therefore, within a production line different products with those that have the same production characteristics can be produced.

The production lines will be represented according to the set L, and mass produced on each line will be determined by set Lin where for each l we have $\text{Lin}(l) \subset M$, where

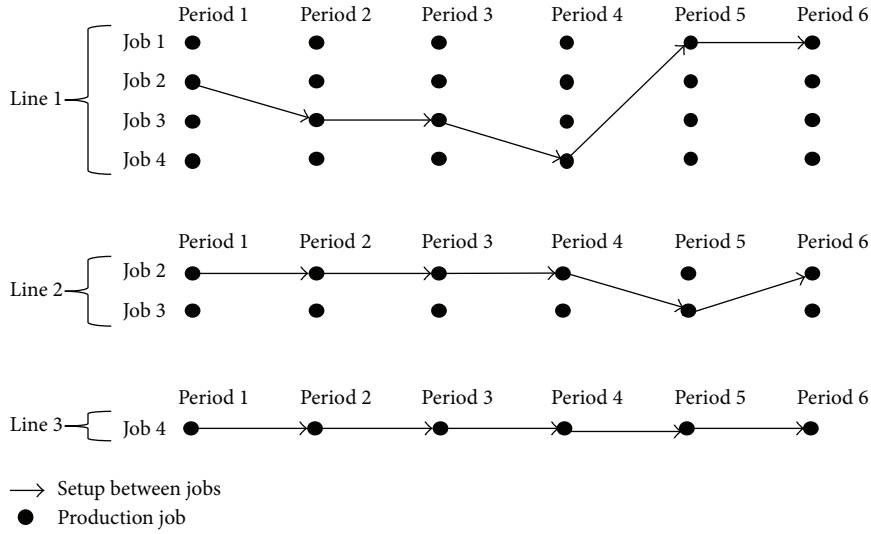

Figure 1: Job sequencing representation.

an input can be present in more than one production line. We emphasize that the production of masses will occur in production batches, and each mass has a certain number of units produced per batch represented by batch(m).

One of the objectives of the problem is to fulfill the demand. Demand is divided by markets such that each market requests a daily mix of products. These requests can be fully addressed, or partially addressed, or not addressed at all. If the market is not attained there is no penalty, only the profit made by that market will not be obtained.

The amount of products to be manufactured is a function of demand for each market. The customers will be represented by the set C, the demand in each customer for a product in a period will be $d(c, p, e)$, and the profit for each sale will be profit(p, c).

In order to transport products to markets, there are options of trucks with different capacities that can be used; all types of trucks will be represented by T, with the cost of each truck given by $\text{cost}_2(c, t)$ and capacity by cap(t). The service to the customer should take place at a specific time because the trucks have schedules for loading in factories and unloading at customers' premises; accordingly, the demands of each market should be seen at a predefined period.

A summary of model features and variables is given in Figure 2.

To describe the model, consider the following model parameters.

2.1.1. Sets

E: set of periods.

M: set of mass.

P: set of products.

Prod(m): set of products produced by mass m.

L: set of production line.

Lin(l): set of mass produced in production line l.

C: set of customer.

T: set of trucks.

2.1.2. Data

$\text{cost}_1(m)$: production cost of mass m.

$t(m)$: production time of mass m.

$s(m_1, m_2)$: setup time from mass m_1 to mass m_2.

batch(m): production batch of mass m.

$d(c, p, e)$: demand of customer c, product p in period e.

profit(c, p): sales profit of product p in customer c.

$\text{cost}_2(c, t)$: transportation cost by truck t to customer c.

cap(t): transportation capacity of truck t.

Define the following set of variables.

2.1.3. Variables

$z(l, m, e)$ = {1, if mass m is produced by production line l in period e; 0, otherwise}.

$y(c, t, e)$: integer amount of truck t sender off to customer c in period e.

stk(p, e): stock quantity of product p in period e.

$x(l, p, e)$: quantity of product p produced in production line l and period e.

$v(c, p, e)$: quantity of product p sold at customer c and period e.

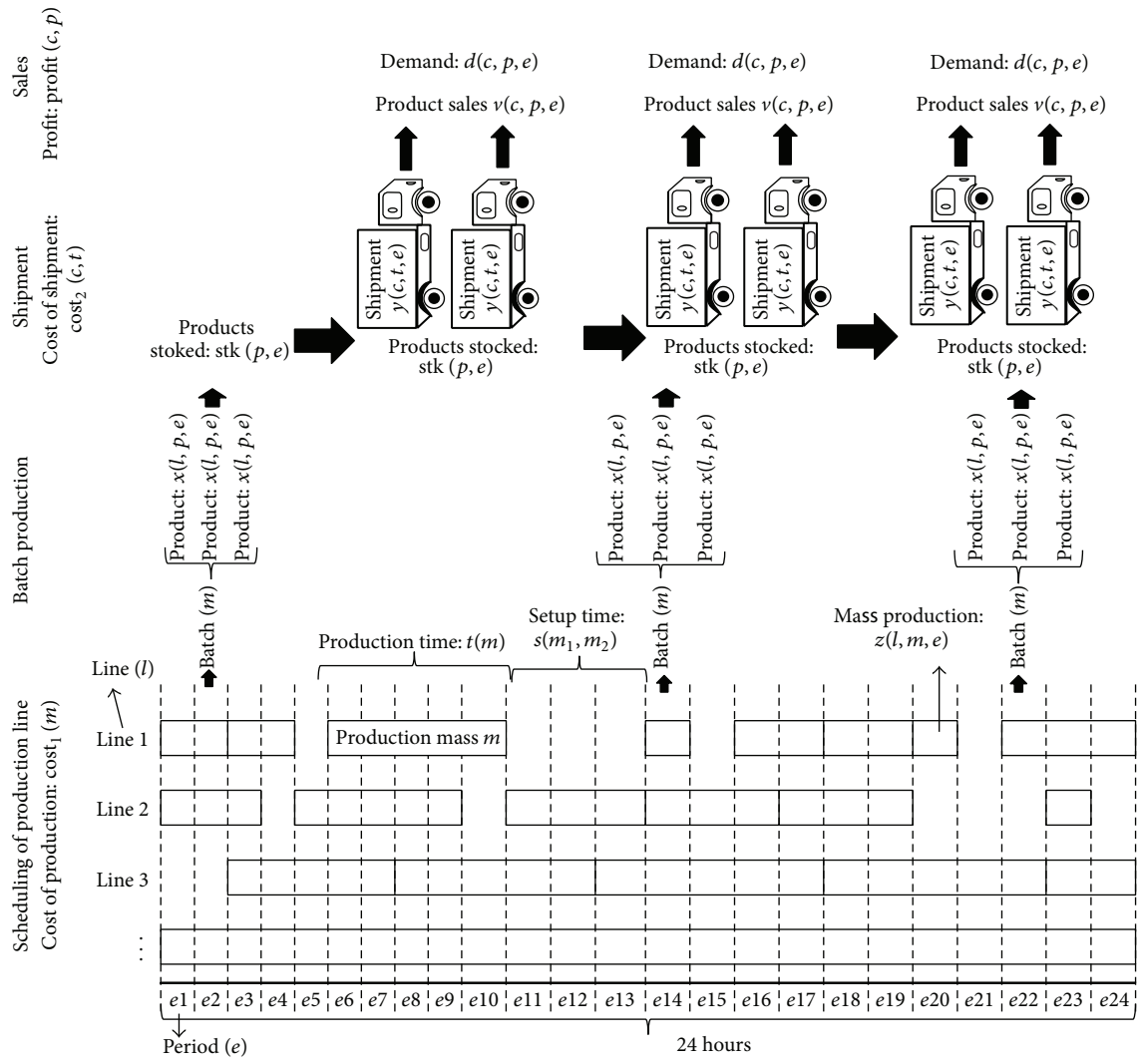

Figure 2: Map of production, shipment, and sales.

2.1.4. Constraints

(1) Unique constraint on the use of the line at each period: in each line, only one product can be produced per period because it is not possible for a machine to run two products simultaneously

$$\forall e, l \quad \sum_{m \in M} z(l, m, e) \leq 1, \tag{1}$$

where $z(l, m, e)$ is a binary variable that represents the production of mass m in period e and line l.

(2) Constraints of stock training: at each period, the stock is formed by the stock from the previous period plus the sum of what was produced, minus the sale for a given product

$$\forall e, p \quad \text{stk}(p, e) = \text{stk}(p, e-1) + \sum_{l \in L} x(l, p, e - t(m)) - \sum_{c \in C} v(c, p, e), \tag{2}$$

where $\text{stk}(p, e)$ is the stock of product p in the period e, $t(m)$ is time of production for mass m that produces p, $x(l, p, e)$ is the amount of production of product p in the period e and line l, and $v(c, p, e)$ is the sales of product p in the customer c and the period e.

(3) Demand constraints: at each customer, the sale of a product cannot exceed the requested demand

$$\forall c, p, e \quad v(c, p, e) \leq d(c, p, e). \tag{3}$$

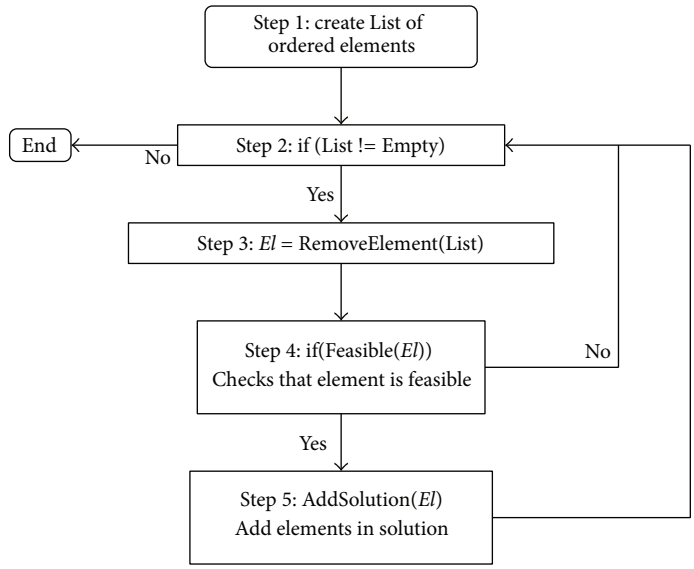

Figure 3: Representation of the greedy heuristic.

Priority list

ID	Customer	Product	Profit	Period
1	1	1	20	5
2	1	3	19	5
3	2	1	19	3
4	1	4	18	5
5	1	5	17	5
6	2	3	10	3
7	1	2	5	5
8	1	4	4	5
9	2	4	2	3
⋮	⋮	⋮	⋮	⋮

Scheduling of production line

Period	1	2	3	4	5
Line 1	...	(ID)-Product X	(3)-Product 1	...	(1)-Product 1
Line 2	...	...	(7)-Product 2	(5)-Product 5	(2)-Product 6
Line 3	...	(9)-Product 4	(6)-Product 3	(8)-Product 4	(4)-Product 4

Shipment

Customer 2
Product 1
Product 3
Product 4

Shipment

Customer 1
Product 1
Product 2
Product 3
Product 4
Product 5

Figure 4: Matrix representation of solution to the greedy heuristic.

(4) Constraints of setup time (cleaning and change of mass in the production line): at each exchange of mass the machine should be stopped by a number of periods defined by $s(m, m_1)$. Therefore, if the machine is used for a mass m, the other mass m_1 cannot use the machine for the next $s(m, m_1)$ periods

$$\forall e, l, m \quad P * (1 - z(l, m, e)) \geq \sum_{m_1 \in \text{Lin}(l)} \sum_{i=1}^{s(m,m_1)} z(l, m_1, e + i), \tag{4}$$

where P is a large enough number compared to the variables and problem constants.

(5) Production batch constraints: each mass production creates several units of products; then if a mass is produced, the amount of batch(m) units of products is produced

$$\forall l, m, e \sum_{i \in \text{Prod}(m)} x(l, i, e) = \text{batch}(m) * z(l, m, e). \tag{5}$$

(6) Constraints of transport capacity: there is a limit to the amount of goods that each truck can transport; therefore, if a truck is used, a maximum of cap(t) units of products will be sent by truck t, where the truck is filled with baskets of the equal dimensions

$$\forall c, e \quad \sum_{p \in P} v(c, p, e) \leq \sum_{t \in T} \text{cap}(t) * y(c, t, e), \tag{6}$$

where $y(c, t, e)$ is the amount of trucks used for customer c over period e and type of trucks t.

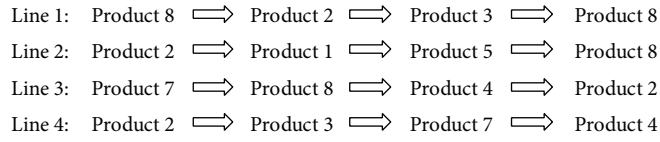

Figure 5: Production scheduling.

8	2	3	8
2	1	5	8
7	9	4	2
2	3	7	4

Figure 6: Matrix representation of solution to the genetic algorithm.

2.1.5. Objective Function. The objective of the problem is to maximize profit, that is, maximize the difference between the sale price of each product in each market and production costs of each product combined with transportation costs. The problem presented is characterized as a mixed integer linear programming problem

$$\begin{aligned} f(v,y) = {} & \sum_{c\in C}\sum_{p\in P}\sum_{e\in E}\text{profit}(c,p) * v(c,p,e) \\ & - \sum_{c\in C}\sum_{t\in T}\sum_{e\in E}\text{cost}_2(c,t) * y(c,t,e) \\ & - \sum_{l\in L}\sum_{m\in M}\sum_{e\in E}\text{cost}_1(m) * z(l,m,e). \end{aligned} \tag{7}$$

The mixed integer linear programming problem consists of

$$\begin{aligned} &\text{Max} \quad f(v,y) \\ &\text{s.t.} \quad (1),(2),(3),(4),(5),(6) \\ &\qquad \text{where } x, v, \text{stk} \in R_+, \\ &\qquad z, y \in Z_+. \end{aligned} \tag{8}$$

2.2. Solution Methods and Implementation Details. Several methods are proposed to solve the problem. This problem can be solved by an exact method for solving integer linear programming problems, such as branch-bound. The solution will be presented through the Xpress solver using the interior point method and branch-bound in the default solver configuration. Other solution methods presented will be a method based on a greedy heuristic (Figure 4) and also metaheuristics of the genetic algorithm type to solve the problem.

2.2.1. Greedy Heuristic. The greedy algorithm can be found in Introduction to Algorithms by Cormen et al. [23] or Bendall and Margot [24]. In the context of scheduling delivery date problems, the problem is solved exactly for one machine in ([25], page 207).

The greedy heuristic can be divided into two phases:

(1) start-up and creation of the priority list ordered,

(2) production sequencing.

A list of priorities should be created to be used in the algorithm; accordingly, the following criteria will be used:

$$\begin{aligned} &\text{Priority}(c,p) \\ &= (\text{(Sales Price}(c,p) - \text{Production Cost}(m) \\ &\qquad - \text{Average Cost of Shipping}(p))) \\ &\qquad \times (\text{(Production Time}(m) - \text{Average Setup}(m)))^{-1}, \end{aligned} \tag{9}$$

where each product is p, m is mass that produces the product p, c is a customer, and

$$\begin{aligned} &\text{Average Cost of Shipping}(p) \\ &= \frac{\sum_{c\in C|d(c,e,p)\neq 0}\sum_{t\in T}\text{cost}_2(c,t)}{\sum_{t\in T}\text{cap}(t)}, \\ &\text{Average Setup}(m) \\ &= \frac{\sum_{m_1\in M|s(m,m_1)\neq 0} s(m,m_1)}{\left|\{m_1 \in M \mid s(m,m_1) \neq 0\}\right|}. \end{aligned} \tag{10}$$

For each product in each market we can prioritize the one with the biggest impact in the objective function and it will have the highest execution priority.

In the production sequencing for each element of the list *Priority*, we checked whether it would be possible to run it on a production line so that the solution does not become infeasible; that is, for each element determined by a product, market, and delivery time, we have to run it before the delivery time. Thus, if space is available (idle line space before the delivery time longer than production and setup time of the product in question) in a line that produces this element, this product will run within this space. If space is not available, the element is discarded. We note that the task should be executed as late as possible, in order to ensure that products with earlier delivery times and lower priorities can occupy earlier positions in the sequence.

At the production adjustment stage, the algorithm will only place idle time between the execution of two tasks at the end of the stage, because idle time between two tasks is

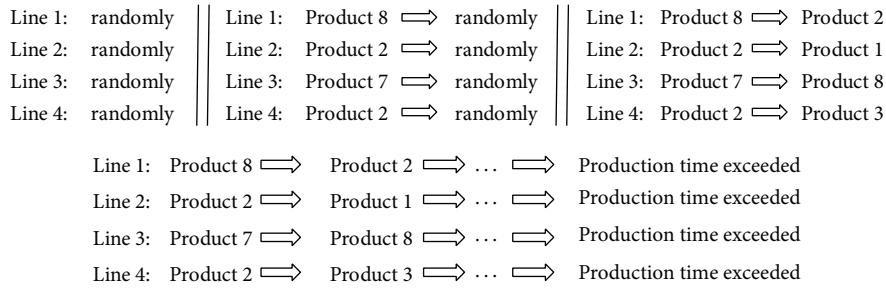

Figure 7: Creation of initial population to genetic algorithm.

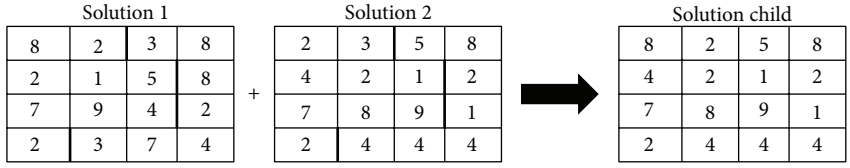

Figure 8: Crossover to genetic algorithm.

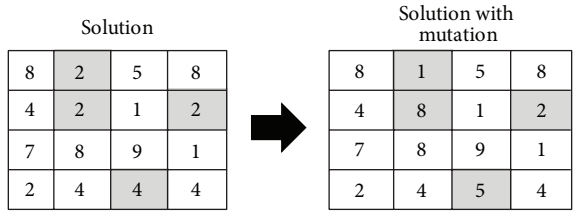

Figure 9: Mutation to genetic algorithm.

impractical; therefore it can be adjusted without any loss in objective function.

We can represent the heuristic by means of Figure 3.

El is one element of *priority* list that represents a sale of product p in customer c.

2.2.2. Genetic Algorithm. The application of methods for using genetic algorithm in scheduling problems can be found at Allahverdi et al. [26], which shows several authors working with genetic algorithms in different scheduling problems.

The genetic algorithm is based on building an initial population where each individual represents a possible solution, and, through this population, building new populations through an evolutionary process to find better solutions.

We can divide the algorithm into a few steps: initial population, crossover, mutation, and selection.

To represent an element of the population on the genetic algorithm will be used one matrix; this representation is unusual to the genetic algorithm in implementations already known. Each row represents a production line and each column a product that has been produced. This matrix has dimension $\mathbb{Z}^{n\times m}$, with n being quantity of lines and m maximum quantity of products produced.

For each production line we will have a set of genes where each gene represents the product to be produced. Therefore, the solution is represented by a set of schedules of lines, which are sets of chromosomes, where each chromosome is a set of genes as shown in Figure 5, which represents in each line what will be produced by the genetic algorithm represented in Figure 6.

(1) Creation of Initial Population. The initial population is created randomly by respecting the feasibility of the solution. For each element of the population vectors are created, and each vector represents a production line and each matrix element represents the product that will be produced according to Figure 7, in order that the sum of production time and setup of a line does not exceed the maximum time that the line can operate. Thus, we can create all the elements of the population.

(2) Crossover. The crossover phase will start the process of building the next population. At this stage, two elements are chosen at a time, and these two elements exchange components from their solutions with each other. This exchange of components is known as crossover; it is executed by choosing two random points of each parent chromosome, and these components are inherited to the new solution that will be created. After a certain number of children are formed, the crossover phase is over (Figure 8).

(3) Mutation. In this phase, each child element created in the crossover phase is mutated, in numbers determined by

```
begin
  /*Upper Bound/;
  at = 0;
  v = 0;
  foreach p ∈ P do
    foreach i ∈ {n, ..., 1}|e_i ∈ E do
      Qtprov(p, e_j) = Qt(p, e_j);
      foreach j ∈ D_{p,i} do
        v(p, c_j) = min(d(p, c_j) − at(p, c) j, Qtprov(p, e_j));
        Qtprov(p, e_j) = Qtprov(p, e_j) − v(p, c_j);
        L = L + v(p, c_j) ∗ l(p, c_j);
        at(p, c_j) = at(p, c_j) + v(p, c_j);
      end
    end
  end
end
```

Algorithm 1: Upper bound to genetic algorithm.

the mutation rate of their genes. This is done by randomly choosing a gene and replacing it with another element so that the solution remains feasible. This new element is also chosen randomly (Figure 9).

(4) Selection. Selection can be executed in various manners. Here, this will be done in two different ways: tournament and selection of the best individual. After creating the child elements of the population in the previous steps, a new population will be formed to run a new iteration. With the option of choice of both parent and child components for the next generation, the tournament will randomly select a predetermined number of elements and among them the ones with the best objective function; this process is repeated until a new generation is built. Through the method of selecting the best individual, the best individuals comprise the new population.

Algorithm 1 is proposed to calculate the upper bound of the elements of genetic algorithm. In this algorithm we introduce L as the desired upper bound; $Qt(p, e_j)$ is the amount of product p produced between the steps e_{j-1} and e_j, $at(p, c_j)$ is the amount of the product p met in customer c_j, and $D_{(p,i)}$ is the vector of customer indices, sorted by the profit of product p from the period e_i to period e_n.

We can represent the metaheuristics with Figure 10.

2.2.3. Efficiency. To evaluate the efficiency of greedy and genetic algorithm, a new ILP model is presented. It is necessary because greedy algorithm is calculated through priority list defined in (9) and genetic algorithm through of upper bound calculation defined in Algorithm 1. This model is easy to resolve due to little quantity of integer variables. To all tests the runtime was less than 10 seconds.

For the model to follow, each method will supply a resulting scheduling to the model that evaluates the efficiency, which will obtain the value of a function of an ILP problem as efficiency of the method.

The index P, E, C, and T and the parameters $d(c, p)$, $l(p, c)$, $\text{cost}_2(c, t)$, and $\text{cap}(t)$ were defined in Section 2 and given $Qt(p, e)$ as the amount of product p available for shipment during period e calculated by scheduling informed as a parameter. And we obtain the real variable $v(c, p)$ being the amount of product p sold at the customer c and the integer variables $env(c, t, p, e)$ are the quantity of product p shipped to customer c by truck t over period e and $z(c, t, e)$ is the truck release t to customer t in the period e.

Constraints

(1) Demand: inequality defined in (3).

(2) Total sales of each product to the market

$$\forall p, c \quad v(c, p) = \sum_{t \in T} \sum_{e \in E} en(c, t, p, e). \tag{11}$$

(3) Shipment of products according to the period

$$\forall p, e \quad \sum_{(e' \in Ee' \leq e)} \sum_{t \in T} \sum_{c \in C} en\left(c, t, p, e'\right) \leq Qt(p, e). \tag{12}$$

(4) Truck capacity

$$\forall c, t, e \quad \sum_{p \in P} en(c, t, p, e) \leq \text{Cap}(t) * z(c, t, e). \tag{13}$$

The objective function is defined by (7).

Thus, we have the integer linear programming problem that will give us the result of each method for the presented problem

$$\begin{aligned} &\text{Max} \quad (7) \\ &\text{s.t.} \quad (3), (11), (12), (13) \\ &\qquad \text{where } v \in R_+, \\ &\qquad en, z \in Z_+. \end{aligned} \tag{14}$$

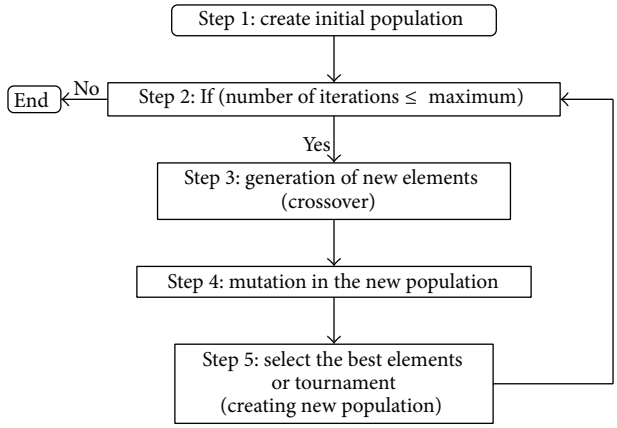

Figure 10: Representation of the genetic algorithm.

3. Results and Discussion

The following are the results of a presentation by a real example of baking industry and the analysis of the complexity of each method. The tests were carried out on a computer equipped with Intel Core i7, 2.93 GHz, 6 GB of memory, and the 64-bit Windows 7 operating system. The resolution of integer linear programming models was calculated by the solver XPress 7.1. The greedy heuristic and genetic algorithm methods were implemented in C-language.

3.1. Algorithm Performance. In order to analyze the performance of the greedy heuristic, we will calculate the processing time of the algorithm due to the growth of its dimensions. First we will examine the additional time as a function of the growing number of products and markets. Accordingly, we fix the number of production lines at 40 and the number of vehicles at 5.

Greedy Heuristic. The analysis will be executed with randomly generated data, calculating the average run time of the greedy heuristic as shown in Figure 11.

The plotted points suggest the adjustment of a linear curve, where we can calculate the parameters of the curve by the method of least squares and get the value of $R^2 = 0.9966$, where $R^2 = \sum(\text{error}_i)^2/(n-1)$ and error is the difference between the value of the curve and points and n the total points used, which shows a linear correlation between the data; therefore the complexity depending on the products and markets is approximately (no. product) $*$ (no. markets) that is, $O(p * c)$.

Now if we fix the number of products and markets we can analyze the variation in the runtime depending on the number of lines and production options for each product. We will set this for 80 products, 20 destinations, and 5 vehicles.

Figure 12 presents the result of complexity in terms of lines and options for each product line.

We can see by Figure 12 that the curve is a 2-degree polynomial and, by applying least squares, we get $R^2 = 0.9746$ showing a correlation between the variables, and it can be said that complexity in terms of production lines is $O(l^2)$.

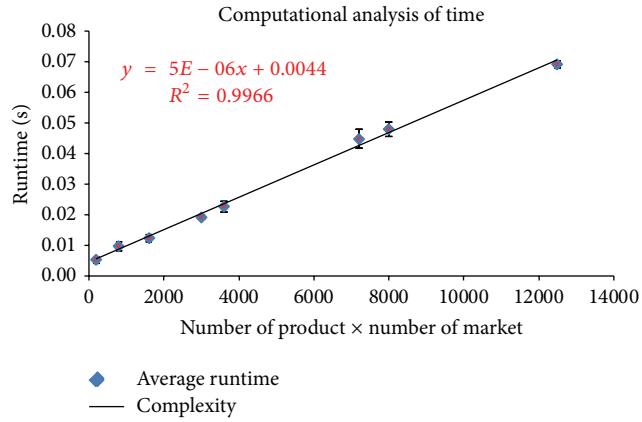

Figure 11: Complexity analysis of the greedy heuristic to quantity of product × market.

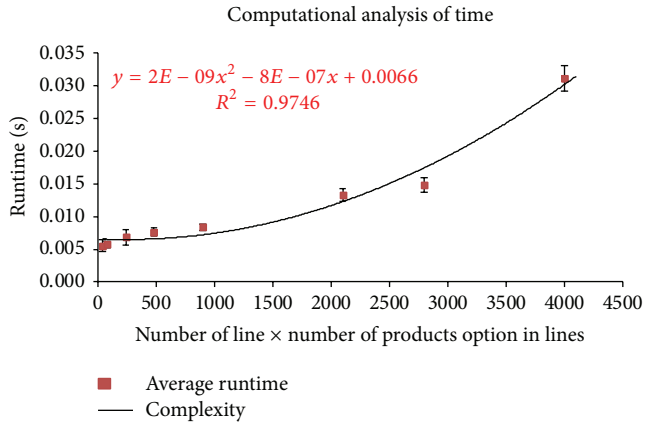

Figure 12: Complexity analysis of the greedy heuristic to number of lines × products option in lines.

Genetic Algorithm. We can repeat the analysis for the genetic algorithm.

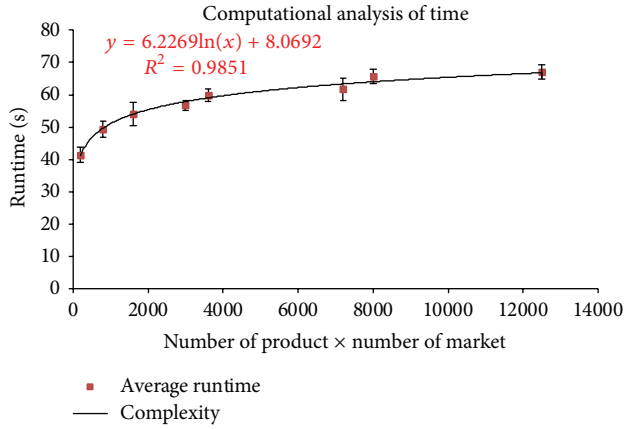

FIGURE 13: Complexity analysis of the genetic algorithm to quantity product × market.

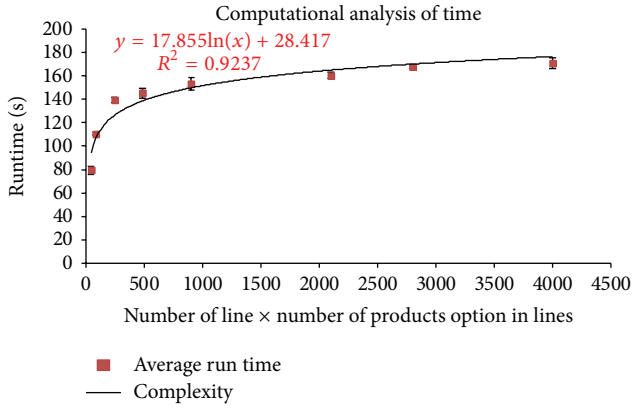

FIGURE 14: Complexity analysis of the genetic algorithm to number of lines × products option in lines.

The set of points (product ∗ destination, time) in Figure 13 suggests a logarithmic function as a setting curve. By finding the parameters by the method of least squares, we obtain the function shown in Figure 13 and we find R^2 with the value of 0.9851; then we can say that the complexity of the algorithm regarding products and destinations is $O(\log(p * m))$.

The set of points (line option ∗ product-line, time) in Figure 14 suggests a setting curve logarithmic function. After finding the parameters by least squares we can say that the complexity is $O(\log(l))$.

3.2. Convergence of Genetic Algorithm. The parameters used to genetic algorithm are in Table 1, and the best settings towards accuracy (approximation of optimal solution) and runtime were AG6, AG16, and AG17, which presented the convergence in 15.

Several parameters were used for genetic algorithm as shown in Table 1; the choice of the best parameters was done by results of run time and accuracy. The genetic algorithm proved little sensitiveness in relation to the variation of

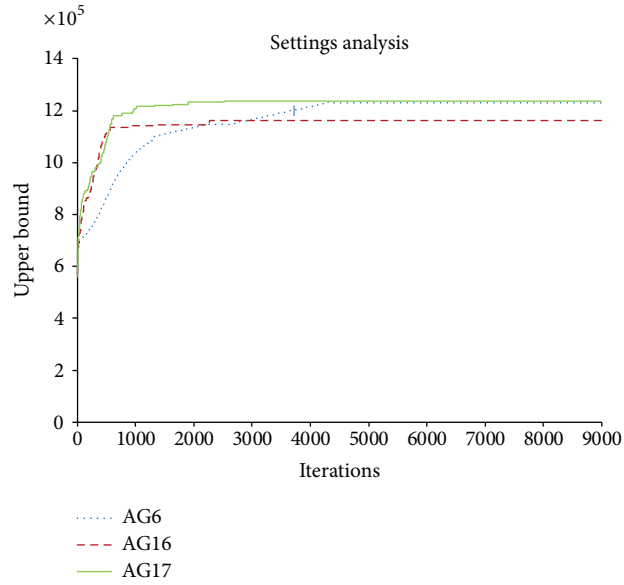

FIGURE 15: Convergence of genetic algorithm to the best parameters.

mutation rate, but high sensitiveness in relation to the selection mode and population size.

Regarding selection mode, the best selection mode found quicker and better solutions if compared to the tournament mode. The convergence of best selection is quicker than the tournament one because it prioritizes the best solutions to continue the methods while tournament tries to select the best participants in a random subset of a given population. This is done in an attempt to find better solutions in other feasible regions.

Therefore, we conclude that the best solutions can be found around a best solution elected by the mode; this is noticed when greedy heuristic and genetic algorithm (when using best select mode with high mutation rate (0.10) and large population size (100 elements)) get good solutions; that is, genetic algorithm gets the best solutions in finding good points and exploring this region around the elected location, and this is why we use the AG17 set.

Analyzing the set of parameters that use the best selection mode, in Figure 15 we can observe the convergence of the sets. AG17 presented the greatest benefit in general: runtime, accuracy, and convergence, while AG16 set had a faster convergence but it converged to a worse solution than AG17 and AG6. The AG6 set got the solution next to AG17, but its convergence was slower if compared to AG17 because it did not explore other feasible regions due to its small population size, taking a bigger runtime.

3.3. Result of Solutions. In Table 3 the efficiency of algorithms that represent the quality of solutions obtained and the instances tested in Table 2 is presented. The data from the tests executed can be found in [27].

In comparing greedy heuristic with genetic algorithm results the first one had a quicker runtime but a worse accuracy. The runtime difference can be explained due to

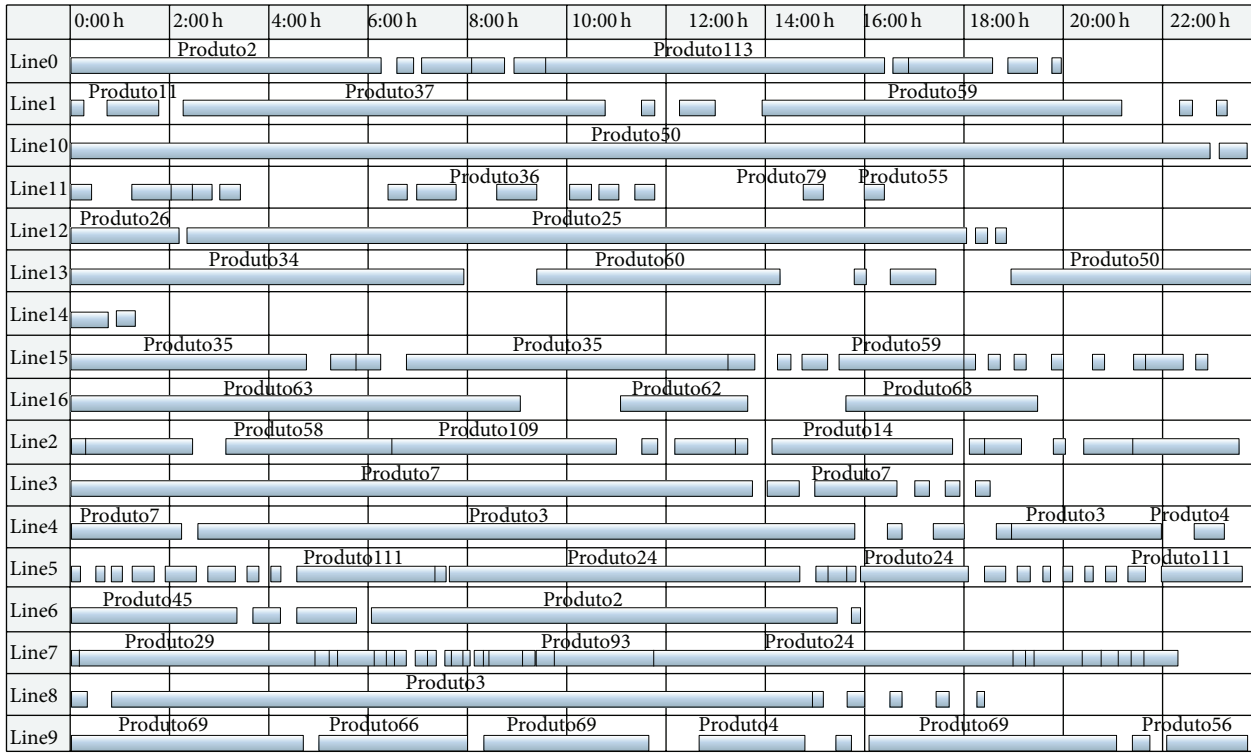

Figure 16: Best production planning (genetic algorithm), Gantt Chart.

Table 1: Genetic algorithm—settings.

Settings	Mutation	Select	Population size
AG1	0.01	Tournament	25
AG2	0.05	Tournament	25
AG3	0.10	Tournament	25
AG4	0.01	Best select	25
AG5	0.05	Best select	25
AG6	0.10	Best select	25
AG7	0.01	Tournament	50
AG8	0.05	Tournament	50
AG9	0.10	Tournament	50
AG10	0.01	Best select	50
AG11	0.05	Best select	50
AG12	0.10	Best select	50
AG13	0.01	Tournament	100
AG14	0.05	Tournament	100
AG15	0.10	Tournament	100
AG16	0.01	Best select	100
AG17	0.05	Best select	100
AG18	0.10	Best select	100

Table 2: Instances to evaluate efficiency.

Problem	Product	Market	Line	Trucks
AE1	11	11	1	5
AE2	25	10	5	5
AE3	50	10	10	5
AE4	60	13	10	5
AE5	70	14	10	5
AE6	80	16	17	5
AE7	90	18	17	5
AE8	115	21	17	5

greedy heuristic which executed less and simpler iterations than genetic algorithm. In the other hand it explores a smaller space to find good solutions.

Genetic algorithm got better accuracy than greedy heuristic which can be seen in Table 3, which is expected because metaheuristics does a search for best solution in a bigger region inside feasible region than greedy heuristic. Although genetic algorithm had worse runtime, it is acceptable because it is within expectation in practical (for the biggest instance was 1.5 minutes) and the accuracy was very satisfactory (for small instances less than 10% of upper bound obtained for ILP).

The objective of the problem is to maximize the profit through the best operation including production sequencing and shipments. Figure 16 and Table 4 demonstrate the best solution according to genetic algorithm to real data (AE8), with the sequencing of each production line over a period of 24 hours.

We can ascertain that the genetic algorithm in its AG17 parameterization had the best performance; for instance AE1 approached the upper bound of the ILP and obtained most of the best solutions. The greedy heuristic presented satisfactory solutions for some instances, but for others it was well below

Table 3: Comparison of results.

Problem	Greedy heuristic		AG06		AG16		AG17		ILP		
	FO	Time (s)	FO	Time (s)	FO	Time (s)	FO	Time (s)	FO	Best bound	Time (s)
AE1	12,733	<0.001	19,259	1.22	19,227	2.76	**20,354**	10.25	6,904	22,750	723.8
AE2	56,077	<0.001	54,127	9.49	54,778	20.59	**59,823**	20.57	54,324	60,567	1134.5
AE3	**120,162**	0.004	115,224	9.59	108,431	46.94	117,770	41.50	—	—	—
AE4	234,464	0.005	379,281	10.93	376,080	51.07	**394,101**	55.73	—	—	—
AE5	417,773	0.005	**494,619**	11.32	491,676	51.31	22,750	57.56	—	—	—
AE6	1,044,610	0.006	1,042,940	14.85	1,151,700	55.98	**1,151,920**	59.75	—	—	—
AE7	1,141,410	0.006	**1,237,610**	18.45	1,177,440	58.86	1,235,990	62.41	—	—	—
AE8	1,174,730	0.007	1,247,800	18.74	1,180,000	67.64	**1,250,010**	92.64	—	—	—

Table 4: Best production planning (genetic algorithm)—real data.

Line00	Product2\22	Product47\2	Product45\6	Product113\4	Product47\4	Product113\41	Product47\2	Product113\10
	Product2\2	Product45\1						
Line01	Product70\1	Product11\4	Product37\34	Product71\1	Product48\3	Product59\29	Product15\1	Product48\1
	Product10\2							
Line02	Product32\1	Product109\9	Product58\12	Product109\19	Product14\1	Product32\4	Product109\1	Product14\11
	Product32\1	Product109\3	Product58\1	Product14\3	Product109\9			
Line03	Product7\49	Product44\2	Product7\6	Product44\1	Product7\1	Product44\1		
Line04	Product7\8	Product3\44	Product4\1	Product32\2	Product3\1	Product3\10	Product4\2	
Line05	Product42\1	Product51\1	Product52\1	Product87\2	Product90\2	Product12\3	Product88\1	Product24\1
	Product111\14	Product110\1	Product24\39	Product111\1	Product91\2	Product104\1	Product24\12	Product111\2
	Product88\1	Product75\1	Product89\1	Product72\1	Product87\1	Product54\2	Product111\8	Product74\1
Line06	Product45\20	Product2\2	Product45\7	Product2\33	Product47\1			
Line07	Product56\1	Product29\19	Product36\8	Product98\1	Product24\4	Product27\1	Product13\1	Product92\1
	Product29\1	Product41\1	Product12\1	Product101\1	Product55\1	Product13\1	Product36\3	Product24\4
	Product92\1	Product40\1	Product93\8	Product24\40	Product38\1	Product56\1	Product30\4	Product40\1
	Product13\2	Product92\1	Product36\8	Product56\4				
Line08	Product46\2	Product3\47	Product109\1	Product46\2	Product58\1	Product4\1	Product46\1	
Line09	Product69\14	Product66\9	Product69\10	Product68\1	Product4\7	Product32\1	Product69\15	Product67\1
	Product66\6							
Line10	Product50\77	Product20\2						
Line11	Product85\1	Product86\2	Product112\2	Product85\1	Product82\1	Product78\1	Product85\2	Product86\2
	Product83\1	Product84\1	Product81\1	Product79\1	Product86\1			
Line12	Product26\9	Product25\65	Product26\1	Product25\1				
Line13	Product34\34	Product60\28	Product21\1	Product23\4	Product60\28			
Line14	Product31\2	Product76\1						
Line15	Product35\19	Product11\2	Product114\2	Product35\26	Product114\2	Product39\1	Product80\2	Product59\10
	Product114\1	Product80\1	Product48\1	Product35\1	Product39\1	Product10\1	Product114\3	Product80\1
Line16	Product63\7	Product62\2	Product63\3					

Legend: product *XX* *N* batches.

the best solution; however, the run time was low, unlike the genetic algorithm, which has a greater runtime and ILP, which has a nonviable runtime for executing a plan that needs to be put in place within hours.

Among the parameterizations of the genetic algorithm, AG17 was efficient and stable; although in two instances AG6 was more efficient, in others it was well below the objective function value of AG17, showing some instability with changes in the dimensions of the problem. AG16 failed the best objective function values compared to the others.

The genetic algorithm was efficient for solutions with data from actual dimensions with an acceptable runtime and better objective function value compared to the greedy heuristic, showing whether it is applicable to the problem and stable on the variation of the dimensions of the problem.

In the evaluation of all methods with all the parameterizations presented, the one that performed best was the AG17 parameterization of the genetic algorithm, as it had the best approach for small problems and the best objective function value compared to other possibilities presented. AG17 also

had low run-time, losing to the instances of the greedy heuristic and for some tests below AG16 parameterization, but the quality of the solution of the greedy heuristic is well below the genetic algorithm and the AG16 instance in terms of execution time was very close to the AG17 instance.

4. Conclusions

The aim of this study was to represent a widely regarded scheduling problem in the baking industry, which has conflicting variables, and to propose a mathematical solution to solve it through methods such as genetic algorithm and greedy heuristic.

In this study it was possible to formalize the problem with a mathematical representation so that it can be solved as an ILP problem, because in literature we cannot find a representation of the problem as a whole, only part of the problem.

One of the contributions of this paper is a new model to scheduling problem that is not based in traveling salesman problem (TSP). Due to complexity of due date and the maximization of profit and not minimization of makespan this problem cannot be modeled as TSP.

It was demonstrated that the real scheduling problem is able to be modeling as mixed integer linear program problem different the classical models. And it is possible to resolve this problem utilizing metaheuristics to find good solutions.

The solutions from greedy heuristic presented a very low runtime and the value of objective functions was the next best solution obtained. And the algorithm was shown to have polynomial order of complexity in practical, which shows that even if the problem grows over time it is still acceptable.

However, the genetic algorithm showed strong adaptation to the problem, so the solution was easily represented to be used by the genetic algorithm, making the algorithm easy to implement. The results showed themselves to be very efficient, obtaining good quality solutions for some instances which came close to the upper bound obtained in solving the ILP. The algorithm also proved to be efficient for larger instances, obtaining good solutions in an acceptable runtime, that is, within the limits to enable a viable plan. In the complexity analysis, the algorithm proved itself to be efficient in having logarithmic complexity, which shows that even with the growth of the problem the running time should not vary greatly, thus enabling the execution of the algorithm for larger problems.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

References

[1] H. L. Gantt, *Work, Wages, and Profits*, Engineering Magazine, New York, NY, USA, 1916.

[2] S. M. Johnson, "Optimal two- and three-stage production schedules with setup times included," *Naval Research Logistics Quarterly*, vol. 1, no. 1, pp. 61–68, 1954.

[3] R. Bellman, "The theory of dynamic programming," *Bulletin of the American Mathematical Society*, vol. 60, no. 6, pp. 503–515, 1954.

[4] Z. A. Lomnicki, "A branch-and-bound algorithm for the exact solution of the three-machine scheduling problem," *Operational Research Quarterly*, vol. 16, pp. 89–100, 1965.

[5] E. Ignall and L. Schrage, "Application of the branch -and-bound technique to some flowshop scheduling problems," *Operations Research*, vol. 13, no. 3, pp. 400–412, 1965.

[6] J. D. C. Little, K. G. Murty, D. W. Sweeney, and C. Karel, "An algorithm for the traveling salesman problem," *Operations Research*, vol. 11, no. 6, pp. 972–989, 1963.

[7] M. R. Garey, D. S. Johnson, and R. Sethi, "The complexity of flowshop and jobshop scheduling," *Mathematics of Operations Research*, vol. 1, no. 2, pp. 117–129, 1976.

[8] J. Sridhar and C. Rajendran, "Scheduling in a cellular manufacturing system: a simulated annealing approach," *International Journal of Production Research*, vol. 31, no. 12, pp. 2927–2945, 1993.

[9] C. Rajendran and H. Ziegler, "Heuristics for scheduling in flowshops and owline-based manufacturing cells to minimize the sum of weighted owtime and weighted tardiness of jobs," *Computers & Industrial Engineering*, vol. 37, no. 4, pp. 671–690, 1999.

[10] J. E. Schaller, J. N. D. Gupta, and A. J. Vakharia, "Scheduling a flowline manufacturing cell with sequence dependent family setup times," *European Journal of Operational Research*, vol. 125, no. 2, pp. 324–339, 2000.

[11] J. Schaller, "A new lower bound for the flow shop group scheduling problem," *Computers & Industrial Engineering*, vol. 41, no. 2, pp. 151–161, 2001.

[12] P. M. França, J. N. D. Gupta, A. S. Mendes, P. Moscato, and K. J. Veltink, "Evolutionary algorithms for scheduling a flowshop manufacturing cell with sequence dependent family setups," *Computers & Industrial Engineering*, vol. 48, no. 3, pp. 491–506, 2005.

[13] S. H. Hendizadeh, H. Faramarzi, S. A. Mansouri, J. N. D. Gupta, and T. Y. ElMekkawy, "Meta-heuristics for scheduling a flowline manufacturing cell with sequence dependent family setup times," *International Journal of Production Economics*, vol. 111, no. 2, pp. 593–605, 2008.

[14] F. D. Croce, R. Tadei, and G. Volta, "A genetic algorithm for the job shop problem," *Computers & Operations Reseach*, vol. 22, no. 1, pp. 15–24, 1995.

[15] J. Adams, E. Balas, and D. Zawack, "The shifting bottleneck procedure for job shop scheduling," *Management Science*, vol. 34, no. 3, pp. 391–401, 1988.

[16] C. H. Dagli and S. Sittisathanchai, "Genetic neuro-scheduler: a new approach for job shop scheduling," *International Journal of Production Economics*, vol. 41, no. 1–3, pp. 135–145, 1995.

[17] T. S. Arthanari and K. G. Ramaswamy, "An extension of two machine sequencing problem," *Operation Research*, vol. 8, pp. 10–22, 1971.

[18] S. A. Brah and J. L. Hunsucker, "Branch and bound algorithm for the flowshop with multiple processors," *European Journal of Operational Research*, vol. 51, no. 1, pp. 88–99, 1991.

[19] T. J. Sawik, "A scheduling algorithm for flexible flow lines with limited intermediate buffers," *Applied Stochastic Models and Data Analysis*, vol. 9, no. 2, pp. 127–138, 1993.

[20] F.-Y. Ding and D. Kittichartphayak, "Heuristics for scheduling flexible flow lines, computers," *Industrial Engineering*, vol. 26, no. 1, pp. 27–34, 1994.

[21] A. Guinet, M. M. Solomon, P. K. Kedia, and A. Dussauchoy, "A computational study of heuristics for two-stage flexible flowshops," *International Journal of Production Research*, vol. 34, no. 5, pp. 1399–1415, 1996.

[22] E. Nowicki and C. Smutnicki, "The flow shop with parallel machines: a tabu search approach," *European Journal of Operational Research*, vol. 106, no. 2-3, pp. 226–253, 1998.

[23] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, "Greedy algorithms," in *Introduction to Algorithms*, pp. 379–399, MIT Press, Cambridge, Mass, USA, 2001.

[24] G. Bendall and F. Margot, "Greedy type resistance of combinatorial problems," *Discrete Optimization*, vol. 3, no. 4, pp. 288–298, 2006.

[25] D. L. Applegate, R. E. Bixby, V. Chvatal, and W. J. Cook, *The Travelling Salesman Problem: A Computational Study*, Princeton University Press, Princeton, NJ, USA, 2006.

[26] A. Allahverdi, C. T. Ng, T. C. E. Cheng, and M. Y. Kovalyov, "A survey of scheduling problems with setup times or costs," *European Journal of Operational Research*, vol. 187, no. 3, pp. 985–1032, 2008.

[27] F. A. M. da Silva, *The application of scheduling in the industry [M.S. dissertation]*, University of Campinas, São Paulo, Brazil, 2011.