Chapter 1

Using Multiobjective Genetic Algorithm and Multicriteria Analysis for the Production Scheduling of a Brazilian Garment Company

Dalessandro Soares Vianna, Igor Carlos Pulini and
Carlos Bazilio Martins

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/53701

1. Introduction

The Brazilian garment industry has been forced to review its production processes due to the competition against Asiatic countries like China. These countries subsidize the production in order to generate employment, which reduces the production cost. This competition has changed the way a product is made and the kind of production. The industry has focused on customized products rather than the ones large-scale produced. This transformation has been called "mass customization" [1].

In this scenario the Brazilian garment industry has been forced to recreate its production process to provide a huge diversity of good quality and cheaper products. These must be made in shorter periods and under demand. These features require the use of chronoanalysis to analyze the production load balance. Since the production time becomes crucial, the task[1] allocation must regard the distinct production centers[2]. Most of a product lead time – processing time from the beginning to the end of the process – is spent waiting for resources. In the worse case, it can reach 80% of the total time [2]. So the production load balance is critical to acquire a good performance.

It is hard to accomplish production load balance among distinct production centers. This balance must regards the available resources and respect the objectives of the production.

[1] Tasks: set of operations taken on the same production phase.

[2] Production centers: internal or external production cell composed by a set of individuals which are able to execute specific tasks.

Lindem [3] argues that these scheduling problems are NP-Complete since the search space is a factorial of the number of variables. These problems may be solved by using exact methods. However due to time constraints, heuristics must be used in order to find good quality solutions within a reasonable time.

Nowadays the ERP (Enterprise Resource Planning) systems used by the Brazilian garment industry do not consider the finite source of resources and the constraints of the real production environment [3]. Task scheduling is done manually through simple heuristics techniques like FIFO (First In First Out) and SPT (Shortest Processing Time). Although those techniques can generate feasible solutions, these ones usually have poor quality.

In real optimization problems, as the problem addressed in this work, is generally desirable to optimize more than one performance objective at the same time. These objectives are generally conflicting, i.e., when one objective is optimized, the others become worse. The goal of multiobjective combinatorial optimization (MOCO) [4] [5] is to optimize simultaneously more than one objective. MOCO problems have a set of optimal solutions (instead of a single optimum) in the sense that no other solutions are superior to them when all objectives are taken into account. They are known as *Pareto optimal* or *efficient* solutions.

Solving MOCO problems is quite different from single-objective case, where an optimal solution is searched. The difficulty is not only due to the combinatorial complexity as in single-objective case, but also due to the research of all elements of the efficient set, whose cardinality grows with the number of objectives.

In the literature, some authors have proposed exact methods for solving specific MOCO problems, which are generally valid to bi-objective problems but cannot be adapted easily to a higher number of objectives. Also, the exact methods are inefficient to solve large-scale NP-hard MOCO problems. As in the single-objective case, the use of heuristic/metaheuristic techniques seems to be the most promising approach to MOCO problems because of their efficiency, generality and relative simplicity of implementation [5] [6] [7]. Genetic algorithms are the most commonly used metaheuristic in the literature to solve these problems [8].

The objective of this work is to develop a method to carry out the production scheduling of a Brazilian garment company, placed at Espírito Santo state, in real time, which must regularly balance the product demands with the available resources. This is done in order to: reduce the total production time; prioritize the use of internal production centers of the company rather than the use of external production centers; and reduce the downtime of the internal production centers.

With this purpose, initially a mixed integer programming model was developed for the problem. Then, we implemented a multiobjective genetic algorithm (MGA) based on the NSGA-II [4] model, which generates a set of sub-optimal solutions to the addressed problem. After we used the multicriteria method Weighted Sum Model – WSM [9] to select one of the solutions obtained by the MGA to be applied to the production scheduling. The mixed integer programming model, the MGA developed and its automatic combination with the multicriteria method WSM are original contributions of this work.

2. Addressed problem

The production planning process of the Brazilian garment industry may be split into many phases from demand provision to tasks scheduling at each machine. Tubino [2] says that the production planning is defined by the demand from the Planning Master of Production (PMP). This demand is sent to the Material Requirements Planning (MRP) that calculates the material required. Then it becomes available to the Issuance of Production Orders and Sequencing. These steps are depicted at Figure 1.

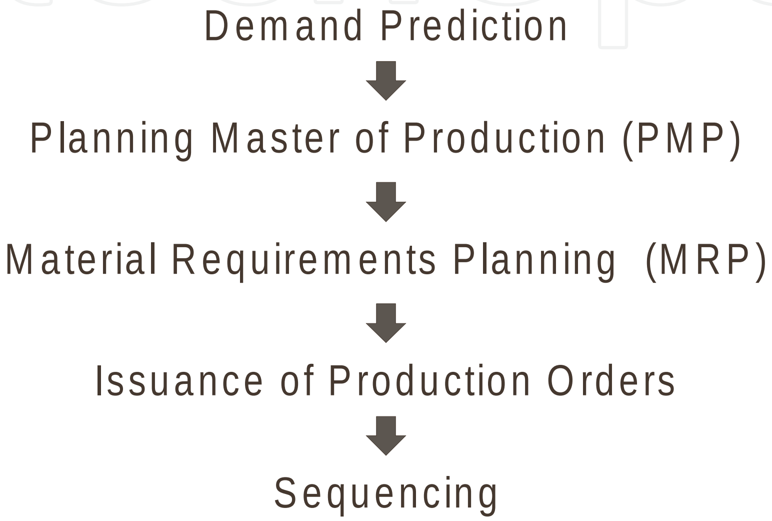

Demand Prediction
Planning Master of Production (PMP)
Material Requirements Planning (MRP)
Issuance of Production Orders
Sequencing

Figure 1. Production planning.

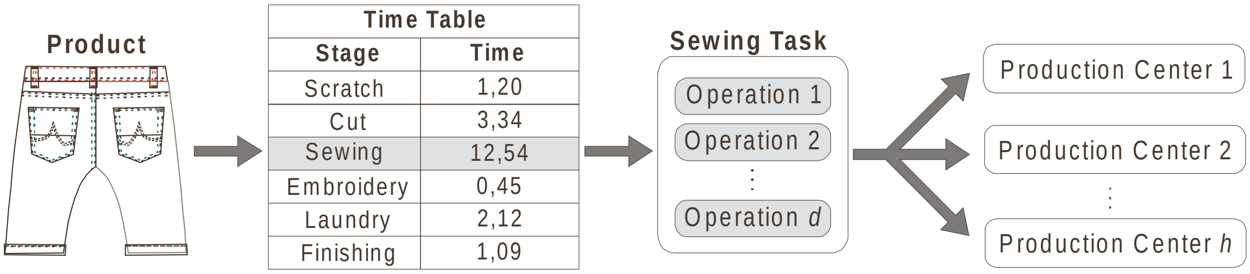
This work approaches the scheduling phase where a set of tasks has to be distributed among production centers. As said before, production center is an internal or external production cell composed by a set of specialized individuals. Each task may be done by a set of production centers and each production center is able to execute many tasks. The objectives of this work are: i) to minimize the total production time (*makespan* – time from the beginning of the first task to the end of the last task); ii) to maximize the use of internal production centers – the use of internal production centers does not imply cost overhead[3] since employees' salary are already at the payroll of the company; iii) to minimize the internal production centers downtime.

These three objectives have been chosen in order to meet the needs of the analyzed company. Some couple of them are conflicting, i.e., when one has an improvement the other tends to get worse. Others objectives are not conflicting, but the optimization of one does not guarantee the optimization of the other. As an example of conflicting objectives, we have the objectives "to minimize the total production time" and "to maximize the use of internal production centers": for minimizing the total production time it is necessary to make the best use of the available production centers, regardless of whether they are internal or exter-

[3] Except when the company has to pay overtime.

nal. The objectives "to minimize the total production time" and "to minimize the internal production centers downtime" are not conflicting: by decreasing the downtime of the production centers, the total production time also tends to decrease. However, if tasks are allocated to an internal production center, which together have an execution time shorter than the total production time, it is possible to arrange them in different ways without changing the total production time. The objective "to minimize the internal production centers downtime" requires the best arrangement of the tasks within each internal production center.

In order to better describe the addressed problem, Figure 2 depicts the steps toward the production of a short. The production process is composed by a set of production stages. Each stage has a set of operations to be performed. In this work, this set is called task. In this example, there are 6 production stages (scratch, cut, sewing, embroidery, laundry and finishing). The sewing task lasts 12.54 minutes and is composed by *d* operations. There are *h* production centers qualified to perform the sewing task.



Time Table	
Stage	Time
Scratch	1,20
Cut	3,34
Sewing	12,54
Embroidery	0,45
Laundry	2,12
Finishing	1,09

Figure 2. Example of a production process.

The execution time of a task is the sum of the execution time of its operations. This time is used during the scheduling, which hides the complexity of the operation distribution inside a stage. So it can be seen as a classical task scheduling where each production center is a machine and the operations set of each production stage is a task.

During the scheduling process the following constraints must be respected: i) for each product exists an execution order of tasks, i.e., there is a precedence order among tasks; ii) each task can only be executed in production centers that are qualified to it, i.e. production centers are specialized; iii) employees stop working regularly for lunch and eventually for others reasons like training or health care; iv) depending on the workload it is possible to work overtime; v) the time spent to go from one to another production center must be considered.

The addressed problem is similar to the flexible job shop problem, in which there is a set of work centers that groups identical machines operating concurrently; inside a work center, a task may be executed by any of the machines available [10]. Figure 3 depicts an example of adapting the flexible job shop to the addressed problem. In this figure, three products are made: Product 1 requires tasks T_{11}, T_{12}, T_{13}, T_{14}, T_{15} and T_{16}; Product 2 requires tasks T_{21}, T

$_{22}$, T_{23}, T_{25} and T_{26}; Product 3 requires tasks T_{31}, T_{32}, T_{33}, T_{34} and T_{36}. All tasks are distributed among production centers C_1, C_2, C_3, C_4 and C_5.

In Figure 3 the problem is divided into 2 subproblems: A and B. At subproblem A the tasks are distributed among the production centers that can execute them. At this step is important to prioritize internal production centers in order to take profit of the company processing power that is already available. At subproblem B the tasks must be scheduled respecting the precedence order of tasks.

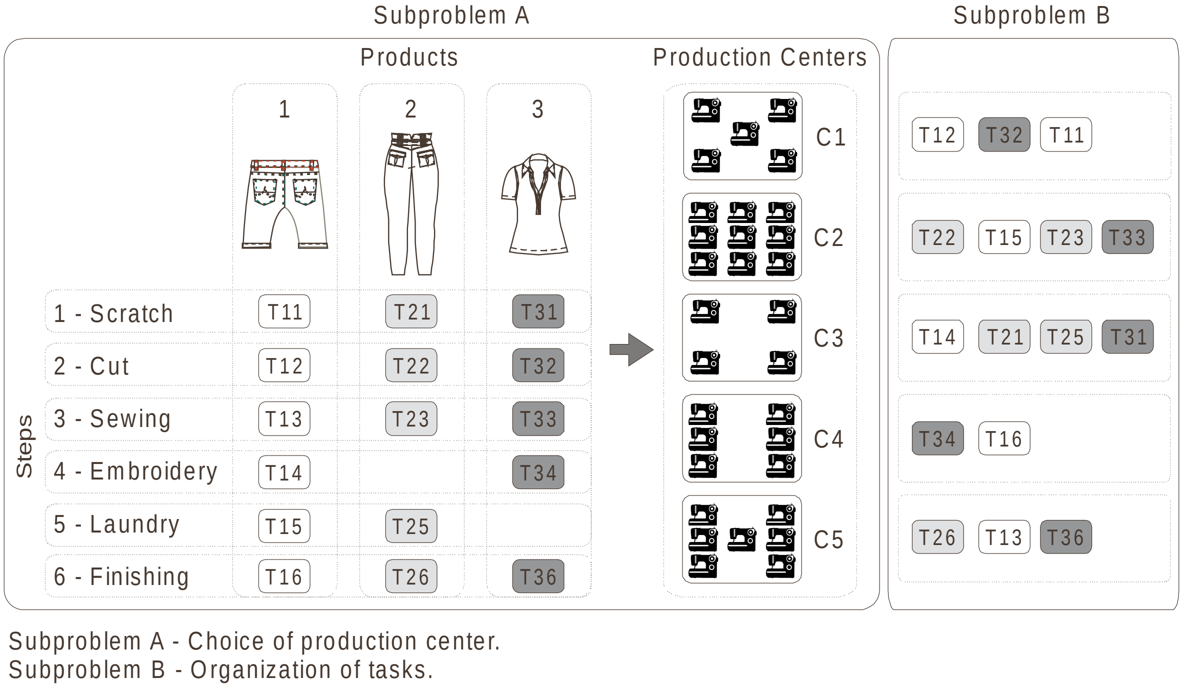


Figure 3. Task distribution among the production centers.

Figure 4(1) depicts an example of scheduling for the tasks listed in Figure 3. Note that the precedence relation among tasks is respected, that is, a task T_{ij}, where i means the product to be made and j the production stage, can be started only after all tasks T_{ik} ($k < j$) have been finished. The Figure 4(2) shows the downtime (gray arrows) in the production centers. For instance, task T_{13} at production center C_5 waits for the task T_{12} at C_3 before starts executing. Figure 4(3) shows that the tasks T_{25} and T_{31} at production center C_1 and T_{38} at C_5 (black boxes) were ready but had to be frozen because of the unavailability of the production centers C_1 and C_5.

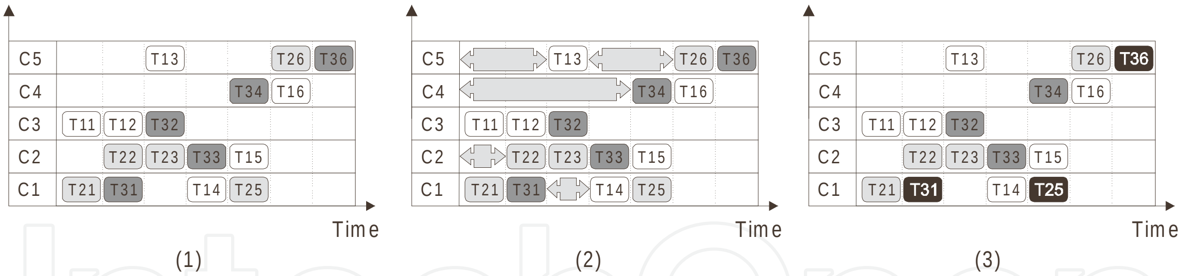


Figure 4. Example of tasks scheduling.

The addressed problem is similar to some works found in the literature, like Senthilkumar and Narayanan [11], Santosa, Budiman and Wiratino [12], Abdelmaguid [13], Dayou, Pu and Ji [14], Chang and Chyu [15] and Franco [16]. However, these works do not consider real-time tasks sequencing or are not applied to real problems.

It is important to note that the chronoanalysis method used here is not the focus of this work. However, in this work, the production time includes tolerance, rhythm and others variables from the chronoanalysis.

2.1. Mathematical modeling

For this modeling was created a sequencing unit (*SU*) which defines a time-slice of work. Each production center has distinct sequencing units, in which tasks are scheduled all day long. Figure 5 depicts a set of sequencing units that describes the behavior of a particular production center. The overtime work is treated as a distinct sequencing unit, since they have particular features like cost.

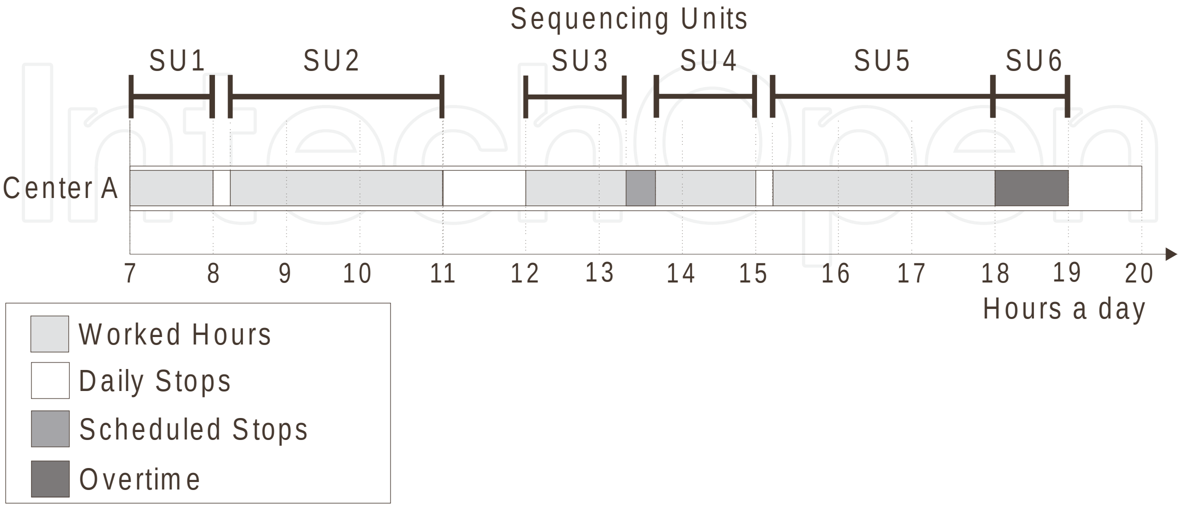


Figure 5. Sequencing units organization.

This model defines a variable N that indicates the total number of tasks, including an additional task that is required for the initialization of the sequencing units.

Below is presented the mixed integer programming model for the addressed problem. The parameters of the problem are presented, followed by the interval indexes, the decision variables and finally by the equations for the three objective functions together with their constraints.

Parameters

NCP – Number of production centers.

NSU – Number of sequencing units.

NJ – Number of tasks to be scheduled.

N – Total number of tasks ($N = NJ + 1$). The last one is the fictitious task that was added to the model as the initial task of every sequencing unit.

M – Large enough value.

WL_i – Workload of task i.

CP_s – Production center of the sequencing unit s.

$Minimum_s$ – Starting time of the sequencing unit s.

$Time_s$ – Amount of time available at sequencing unit s.

CPJ_i – Set of production centers that can execute the task i.

CI – Set of internal production centers.

PRE_i – Set of tasks that are a precondition for the execution of task i.

$OffSet_{c_k c_l}$ – Time for going from production center c_k to c_l.

Indexes

i, j – Indexes of tasks. $i, j \in [1, N]$.

c – Index of production centers. $c \in [1, NCP]$.

s – Indexes of sequencing units. $s \in [1, NSU]$.

Decision variables

$Start_i$ – Non-negative linear variable that represents the starting time of task i.

End_i – Non-negative linear variable that represents the ending time of task i.

WLS_{si} – Non-negative linear variable that represents the workload of task i at the sequencing unit s.

$StartS_{si}$ – Non-negative linear variable that represents the starting time of task i at sequencing unit s.

DT_{sij} – Non-negative linear variable that represents the downtime between tasks *i* and *j* in the sequencing unit *s*.

Y_{sij} – Non-negative linear variable that represents the flow *i*, *j* of the sequencing unit *s*.

MkSpan – Non-negative linear variable that represents the time between the end of the last finished task and the start of the first task.

IntTime – Non-negative linear variable that represents the amount of execution time of tasks in the internal production centers.

DownTime – Non-negative linear variable that represents the amount of internal production centers downtime.

$$Z_{ci} = \begin{cases} 1 & \text{if task } i \text{ is allocated to production center } c. \\ 0 & \text{otherwise} \end{cases}$$

$$U_{si} = \begin{cases} 1 & \text{if task } i \text{ is allocated to sequencing unit } s. \\ 0 & \text{otherwise} \end{cases}$$

$$X_{sij} = \begin{cases} 1 & \text{if the sequence } i,\ j \text{ happens in the sequencing unit } s. \\ 0 & \text{otherwise} \end{cases}$$

Model

$$O.F.1 \ = Min \quad MkSpan \tag{1}$$

$$O.F.2 \ = Max \quad IntTime \tag{2}$$

$$O.F.3 \ = Min \quad DownTime \tag{3}$$

Subject to:

$$MkSpan \geq End_i \quad \forall i \,/\, i \neq N \tag{4}$$

$$IntTime = \sum_{s/CP_s \in CI} \sum_{i/i \neq N} WLS_{si} \tag{5}$$

$$DownTime = \sum_{s/CP_s \in CI} \sum_{i} \sum_{j/j \neq N} DT_{sij} \tag{6}$$

$$DT_{sij} \geq StartS_{sj} - (StartS_{si} + WLS_{si}) - M \times (1 - X_{sij}) \tag{7}$$

$$\sum_{i/CP_s \in CPJ_i} Y_{sij} - \sum_{i/i \neq N, CP_s \in CPJ_i} Y_{sji} = U_{sj} \quad \forall s, \forall j \,/\, j \neq N \, and \quad CP_s \in CPJ_j \tag{8}$$

$$Y_{sij} \le M \times X_{sij} \quad \forall s, \forall i / CP_s \in CPJ_i, \forall j / j \ne N \, and \quad CP_s \in CPJ_j \tag{9}$$

$$U_{sN} = 1 \quad \forall s \tag{10}$$

$$\sum_{c/c \in CPJ_i} Z_{ci} = 1 \quad \forall i / i \ne N \tag{11}$$

$$\sum_{s/CP_s \in CPJ_i} U_{si} \ge 1 \quad \forall i \tag{12}$$

$$U_{si} \le Z_{ci} \quad \forall i, \forall s / CP_s \in CPJ_i, c = CP_s \tag{13}$$

$$\sum_{i/CP_s \in CPJ_i} X_{sij} = U_{sj} \quad \forall s, \forall j / j \ne N \, and \quad CP_s \in CPJ_j \tag{14}$$

$$\sum_{j/j \ne N, CP_s \in CPJ_j} X_{sij} \le U_{si} \quad \forall s, \forall i / i \ne N \tag{15}$$

$$StartS_{si} \ge Minimum_s \times U_{si} \quad \forall s / CP_s \in CPJ_i, \forall i / i \ne N \tag{16}$$

$$StartS_{si} + WLS_{si} \le Minimim_s + Time_s + M \times (1 - U_{si}) \quad \forall s, \forall i / i \ne N \qquad and \quad CP_s \in CPJ_i \tag{17}$$

$$\sum_{s/CP_s = c, CP_s \in CPJ_i} WLS_{si} = WL_{ci} * Z_{ci} \quad \forall i, \forall c \tag{18}$$

$$WLS_{si} \le M \times U_{si} \quad \forall s / CP_s \in CPJ_i, \forall i \tag{19}$$

$$Start_i \le StartS_{si} + M \times (1 - U_{si}) \quad \forall s / CP_s \in CPJ_i, \forall i \tag{20}$$

$$End_i \le StartS_{si} + WLS_{si} - M \times (1 - U_{si}) \quad \forall s / CP_s \in CPJ_i, \forall i \tag{21}$$

$$End_i \ge Start_i \quad \forall i / i \ne N \tag{22}$$

$$StartS_{sj} \ge StartS_{si} + WLS_{si} - M \times (1 - X_{sij}) \quad \forall s, \forall i / CP_s \in CPS_i, \forall j / j \ne N \quad and \quad CP_s \in CPJ_j \tag{23}$$

$$Start_j \ge End_i + OffSet_{c_1 c_2} - M \times (2 - Z_{c_1 i} - Z_{c_2 j}) \quad \forall c_1, \forall c_2, \forall i / i \ne N \quad and \quad c_1 \in CPJ_i, \forall j / j \ne N, i \in PRE_j \, and \quad c_2 \in CPJ_j \tag{24}$$

Where:

1. Objective function that aims to minimize the total production time (*makespan*).
2. Objective function that aims to maximize the use of internal production centers.
3. Objective function that aims to minimize the amount of downtime at the internal production centers.
4. The makespan can be seen as the ending time of the last task.
5. The amount of execution time at the internal production centers.
6. The amount of downtime at the internal production centers.
7. The amount of downtime between tasks *i* and *j* in the sequencing unit *s*.
8. Constrains the flow between tasks *i* and *j*.
9. $X_{sij} = 1$ if there is a flow from task *i* to task *j* at the sequencing unit s.
10. Asserts that task *N* belongs to every sequencing unit.
11. Asserts that each task is executed on just one production center.
12. Asserts that each task is executed on at least one sequencing unit.
13. Asserts that a task *i* can only be executed on a sequencing unit *s* if the task *i* is scheduled to the production center of the sequencing unit *s*.
14. If the task *j* is performed in sequencing unit *s* then there is just one task that immediately precedes *j* in *s*.
15. If the task *j* is performed in sequencing unit *s* then there is at most one task that is immediately preceded by *j* in *s*.
16. Asserts that each task *i* must be started only after the start of the sequencing unit *s* where task *i* is allocated.
17. Asserts that the maximum available time of the sequencing unit is being respected.
18. Asserts that the required workload of task *i* is allocated.
19. Asserts that the workload of task *i* at the sequencing unit *s* is 0 (zero) if task *i* is not scheduled to the sequencing unit *s*.
20. Asserts that the beginning time of task *i*, $Start_i$, must be lower or equal to the beginning time of task *i* at any sequencing unit where it is allocated.
21. Asserts that the ending time of task *i*, End_i, must be greater or equal to the ending time of task *i* at any sequencing unit where it is allocated.
22. Asserts that the ending time of task *i* must be at least equal to its beginning.
23. Asserts that the task *i* starts only after the ending time of the task *j* that immediately precedes *i* in the sequencing unit *s*.

24. Asserts that task j only can starts after the ending time of its predecessor tasks. This restriction takes into account the travel time between the production centers.

3. Proposed method

We propose in this work a method that combines multiobjective genetic algorithm and multicriteria decision analysis for solving the addressed problem. The multiobjective genetic algorithm (MGA) aims to find a good approximation of the efficient solution set, considering the three objectives of the problem. A multicriteria decision analysis method is applied on the solution set obtained by the MGA in order to choose one solution, which will be used by the analyzed garment company.

Deb [4] presents a list of evolutionary algorithms for solving problems with multiple objectives: *Vector Evaluated GA* (VEGA); *Lexicographic Ordering GA*; *Vector Optimized Evolution Strategy* (VOES); *Weight-Based GA* (WBGA); *Multiple Objective GA* (MOGA); *Niched Pareto GA* (NPGA, NPGA 2); *Non dominated Sorting GA* (NSGA, NSGA-II); *Distance-based Pareto GA* (DPGA); *Thermodynamical GA* (TDGA); *Strength Pareto Evolutionary Algorithm* (SPEA, SPEA 2); *Mult-Objective Messy GA* (MOMGA-I, II, III); *Pareto Archived ES* (PAES); *Pareto Envelope-based Selection Algorithm* (PESA, PESA II); *Micro GA-MOEA* (μGA, μGA2); and *Multi-Objective Bayesian Optimization Algorithm* (mBOA).

In this work, we have chosen the NSGA-II [17] evolutionary algorithm since it works with any number of objectives, which can be easily added or removed. This feature facilitates the company to adapt to the market demands – the current objectives may not be sufficient in the future, requiring the company to also focus on other goals –. Besides, there are another Brazilian garment companies interested in using the proposed method, which may have different objectives.

The main methods of multicriteria decision analysis are [18]: Weighted Sum Model, Condorcet method, analytic hierarchic process, ELECTRE methods, Promethee method and MacBeth method.

The Weighted Sum Model – WSM is used in this work due to its simplicity and, mainly, due to its structure of candidates and voters. In this work, WSM performs as a decision maker by considering each solution returned by the MGA as a candidate and each objective of the problem as a voter.

The method proposed in this work is detailed in Section 3.2. But first, in Section 3.1, we describe the multiobjective combinatorial optimization, in order to facilitate the understanding of the proposed method.

3.1. Multiobjective combinatorial optimization

According to Arroyo [19], a Multiobjective Combinatorial Optimization (MOCO) problem consists of minimizing (or maximizing) a set of objectives while satisfying a set of constraints. In a MOCO problem, there is no single solution that optimizes each objective, but a set of efficient solutions in such way that no solution can be considered better than another solution for all objectives.

Among the different ways of defining an optimal solution for MOCO problems, Pitombeira [20] highlights the method proposed by the economist Vilfredo Pareto in the nineteenth century, which introduces the dominance concept. He argues that an optimal solution for a MOCO problem must have a balance between the different objective functions, so that the attempt of improving the value of one function implies the worsening of one or more of the others. This concept is called *Pareto optimal*.

MOCO aims to find the Pareto optimal set (also known as *Pareto frontier*) or the best approximation of it. However, it is necessary to define a binary relationship called *Pareto dominance*: a solution x_1 dominates another solution x_2 if the functional values of x_1 are better than or equal to the functional values of x_2 and at least one of the functional values of x_1 is strictly better than the functional value of x_2 [4]. The Pareto optimal set consists of all non-dominated solutions of the search space.

Deb [4] says that in addition to finding a solution set near to the Pareto frontier, it is necessary that these solutions are well distributed, which allows a broader coverage of the search space. This fact facilitates the decision making, because, regardless of the weight assigned to each criterion, a quality solution will be chosen.

3.2. Multiobjective genetic algorithm proposed

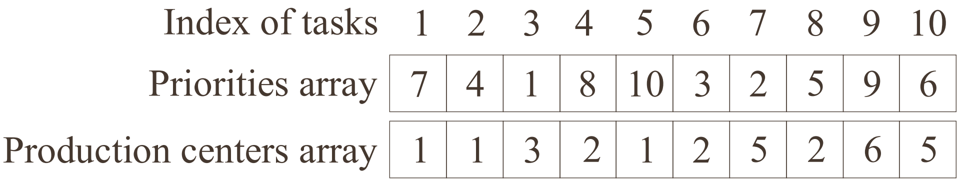
As we have already mentioned, the model adopted for the development of the multiobjective genetic algorithm (MGA) proposed is the NSGA-II. According to Deb [4], it is an elitist search procedure, which preserves the dominant solutions through the generations. The process starts by building a population (*P*), with *nPop* individuals (solutions). The populations of the next generations are obtained through the application of mutation, selection and crossover operators. The purpose is to find a diversified solution set near to the Pareto frontier. With the *crowding distance* [4], we can qualify the space around the solution, allowing a greater diversity of the solution set and, thereby, leading more quickly to a highest quality solution. The crowding distance (*d*) of an individual in the i^{th} position of the population *P*, considering *r* objectives, is given by Equation 25, where f_k^{min} and f_k^{max} represent, respectively, the minimum and maximum values in *P* for the objective function f_k $(1 \leq k \leq r)$. For any solution set, the solution that brings the highest level of diversity is the one with the greatest crowding distance.

$$d_i = \frac{f_1^{(i+1)} - f_1^{(i-1)}}{f_1^{\max} - f_1^{\min}} + \frac{f_2^{(i+1)} - f_2^{(i-1)}}{f_2^{\max} - f_2^{\min}} + \ldots + \frac{f_r^{(i+1)} - f_r^{(i-1)}}{f_r^{\max} - f_r^{\min}} \tag{25}$$

Section 3.2.1 presents the solution representation used in this work. The steps done by the MGA proposed, from the building of the initial population to the choice of the solution to be used by the analyzed garment company, is detailed in Section 3.2.2.

3.2.1. Solution representation

The solution (individual) is represented by two integer arrays: priorities array and production centers array. Tasks are represented by the indexes of both arrays. The priorities array defines the allocation sequence of the tasks and the production centers array indicates the production center responsible for the execution of each task. Figure 6 depicts an example of the solution representation used in this work, in which the first task to be allocated is the task 3 – represented by the position (index) with value 1 in the priorities array – and the production center responsible by its execution is the production center 3 – value of the position 3 of the production centers array –; the second task to be allocated is the task 7, which will be executed in the production center 5; and so on. This representation is based on the representations described in [14] [21] [22] and [23].



Index of tasks	1	2	3	4	5	6	7	8	9	10
Priorities array	7	4	1	8	10	3	2	5	9	6
Production centers array	1	1	3	2	1	2	5	2	6	5

Figure 6. Solution representation.

A task T_i can only start after the end of the predecessor task T_j plus the travel time from the production center responsible by T_i to the production center responsible by T_j. Thus, when a task is selected to be allocated, a recursive search is done in order to allocate the predecessor tasks of it.

3.2.2. Population evolution

The MGA proposed is described by the flowchart of Figure 7, which starts by building the initial population and finalizes when the stop criterion is reached. Mutation, selection and crossover genetic operators are applied in the current population in order to build new individuals (*offsprings*). At the end of each generation, the less evolved individuals are eliminated and the evolutionary process continues with the best individuals.

Step 1 – Building the initial population

Two arrays of size N are created for each individual, where N is the number of tasks to be allocated. The priorities array stores the allocation sequence and the production centers array determines the production center responsible for each task. These arrays are randomly created.

Step 2 – Generating the offspring population

An offspring population, P_{aux}, with *nPop* individuals is created from P, using the tournament selection method [24] and mutation and crossover genetic operators. The tournament method used in this work randomly selects four individuals from P and the best two are selected as the parent individuals to be used by the crossover operator.

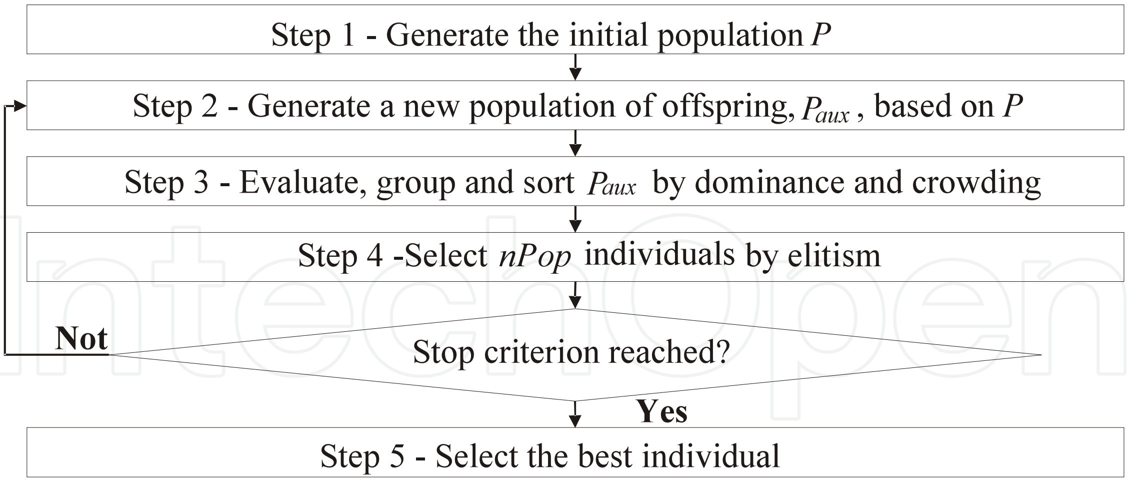

Step 1 - Generate the initial population P
Step 2 - Generate a new population of offspring, P_{aux}, based on P
Step 3 - Evaluate, group and sort P_{aux} by dominance and crowding
Step 4 -Select $nPop$ individuals by elitism
Stop criterion reached?
Not
Yes
Step 5 - Select the best individual

Figure 7. NSGA-II algorithm.

The crossover operator used in this work is based on the variable one-point cut operator [24]. Figure 8 depicts examples of the crossover (8a and 8b) and mutation (8c and 8d) operators developed in this work. As can been seen in Figure 8(a), the priorities array of the offspring individual is composed by the genes of the priorities array of the parent individual *Parent* $_1$ until the cut-point and, from this point, it is composed by the remaining priorities in the order that they appear in the priorities array of the parent individual *Parent* $_2$. In the production centers array, the crossover method is applied by using the same cut-point and the production centers array of the offspring individual is composed by the genes of the production centers array of the parent individual *Parent* $_1$ until the cut-point and, from this point, it is composed by the genes of the production centers array of the parent individual *Parent* $_2$, as can be visualized in Figure 8(b).

The mutation operator is applied at the priorities array as shown in Figure 8(c), where two genes are randomly selected and their contents are exchanged. The mutation operator acts in the production center array as shown in Figure 8(d), where a gene (position *i*) is randomly selected and replaced by another production center capable of execute the task *i*. This production center is randomly chosen. The mutation operator is performed on 5% of the genes of each offspring individual generated by the crossover operator.

Step 3 – Evaluation, sorting and grouping of individuals by dominance and crowding distance

The offspring population P_{aux} is added to the population P, defining a new population of 2×$nPop$ individuals – $nPop$ individuals from P and $nPop$ individuals from P_{aux} –. It is sorted in ascending order by the dominance level[4]. The crowding distance is used as a tie-breaker, i.e., when two individuals have the same dominance level, it is chosen the one with the greatest crowding distance.

[4] The dominance level of an individual *x* is the number of individuals in the population that dominates *x*; for example, an individual dominated by only one individual in the population has dominance level of 1.

Step 4 – Selection of individuals by elitism

The *nPop* best individuals from the new population (P_{aux} added to P) are selected to continue the evolutionary process, while the others are discarded.

Step 5 – Selection of the best individual

At the end of the evolutionary process, the MGA returns a set of individuals with dominance level of 0 (zero), that is, individuals of the current population that no other individual dominates. This set of individuals represents an approximation of the Pareto frontier.

The Weight Sum Model (WSM) [25] multicriteria decision method is applied for choosing a solution from the set returned by the MGA that will be used by the analyzed garment company. In the WSM method, the candidates are ranked by the preferences of the decision maker, in which the best candidate for a particular preference receives 1 (one) point, the second one receives 2 (two) points, and so on. The points received for each preference are summed, and the best candidate is the one with the smallest sum.

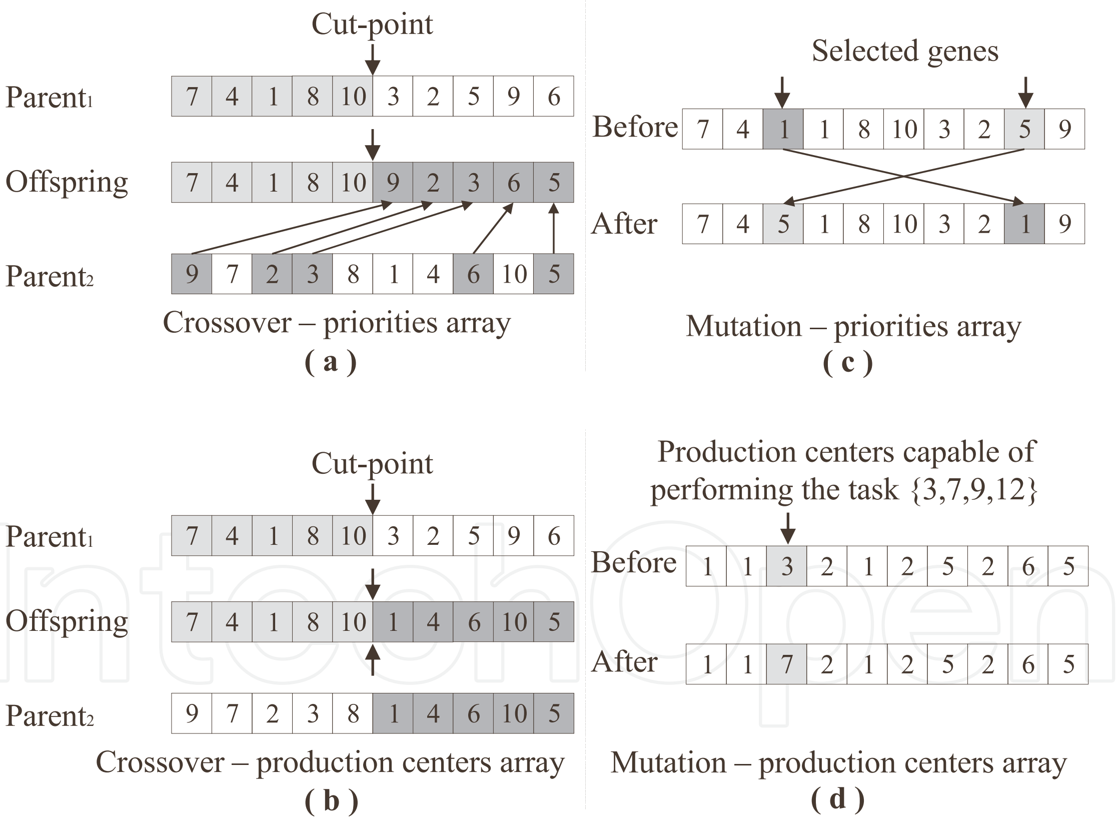

Cut-point
Parent1 7 4 1 8 10 3 2 5 9 6
Offspring 7 4 1 8 10 9 2 3 6 5
Parent2 9 7 2 3 8 1 4 6 10 5
Crossover – priorities array
(a)
Selected genes
Before 7 4 1 1 8 10 3 2 5 9
After 7 4 5 1 8 10 3 2 1 9
Mutation – priorities array
(c)
Cut-point
Parent1 7 4 1 8 10 3 2 5 9 6
Offspring 7 4 1 8 10 1 4 6 10 5
Parent2 9 7 2 3 8 1 4 6 10 5
Crossover – production centers array
(b)
Production centers capable of performing the task {3,7,9,12}
Before 1 1 3 2 1 2 5 2 6 5
After 1 1 7 2 1 2 5 2 6 5
Mutation – production centers array
(d)

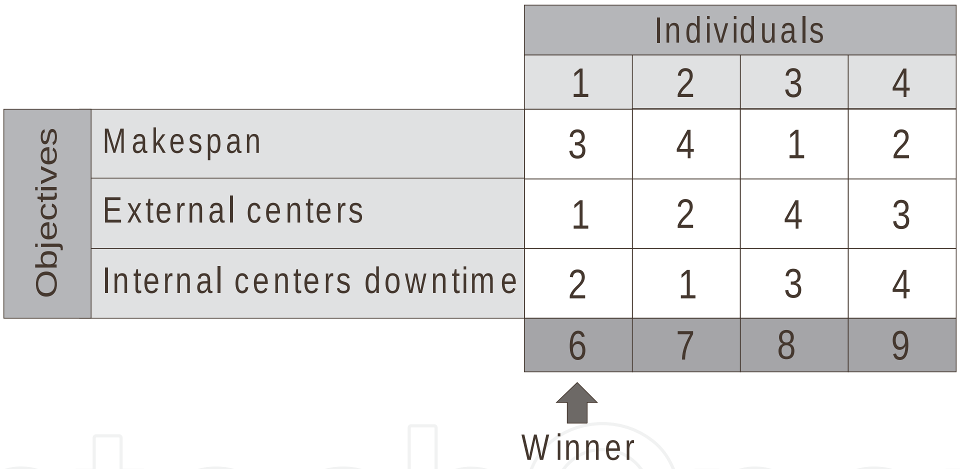
Figure 8. Genetic operators of crossover and mutation.

In this work, the WSM method replaces the grade given by the voters to the candidates. This replacement ranks each solution returned by the MGA according to each objective. Figure 9 illustrates the use of the WSM method in this work where four solutions (columns) must be evaluated according to three objectives (rows). For the first objective, to minimize the total production time (*makespan*), the solution 3 has the best value, obtaining the rank 1; the solution 4 has the best second value, obtaining the rank 2; and the solutions 1 and 2 obtain, respectively, the ranks 3 and 4. The same ranking is done for the others objectives. After summing the rank obtained for each objective, the solution 1 is chosen because it has the smallest sum.

4. Computational results

All computational experiments were performed on a Dell Vostro 3700 notebook with 1.73 GHz Intel Core I7 processor and 6 Gbytes of RAM memory.

The computation experiments regard to real data that represent the May 2012 production demand of the analyzed company: 567 products, 1511 production orders, 3937 tasks and 181 production centers.



		Individuals			
		1	2	3	4
Objectives	Makespan	3	4	1	2
	External centers	1	2	4	3
	Internal centers downtime	2	1	3	4
		6	7	8	9

Figure 9. Weighted sum model.

In the experiments, 12 hours of execution time was established as the stop criterion of the genetic algorithm. This value was defined because it represents the available time between two work days. The population size (*nPop*) and the mutation rate (*tx*) were empirically set at *nPop*=200 individuals and *tx*=5%.

In the first experiment, we compare the results of the proposed method with the results manually obtained by the analyzed company at May 2011. Figure 10 depicts the production deviation of each stage between May 2011 and May 2012, where we can note an increase of the production at almost all stages.

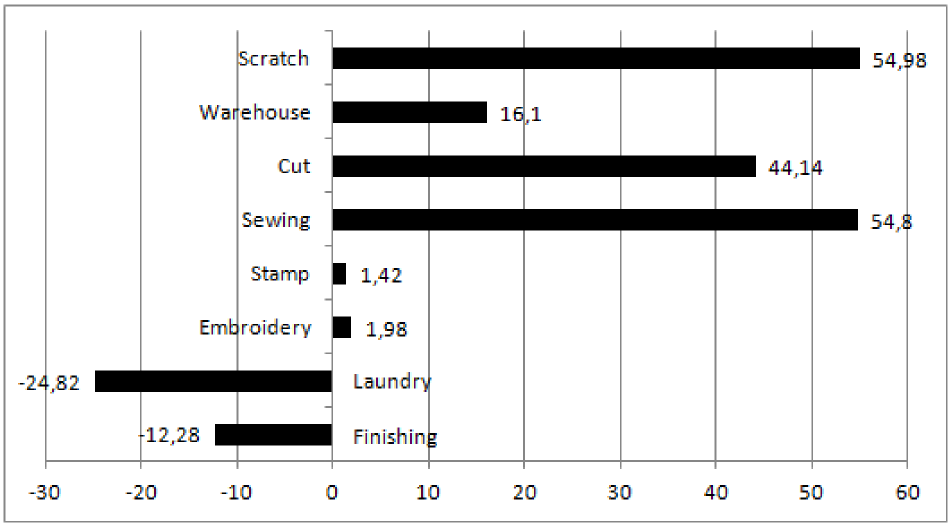


Figure 10. Production deviation of each stage (May 2011 x May 2012).

The results obtained by the analyzed company at May 2011 were: 42 production days to execute all tasks; 20% of the tasks were performed in internal production centers; and the downtime rate of the internal production centers was 37%.

In this experiment, five runs of the proposed method were done, obtaining the following average results: 36 production days to execute all tasks; 32% of the tasks were performed in internal production centers; and the downtime rate of the internal production centers was 16%. It is worth to mention that the worst results obtained are: 38 production days to execute all tasks; 35% of the tasks were performed in internal production centers; and the downtime rate of the internal production centers was 18%. The obtained results were better than the ones manually got at May 2011, even considering the increase of the production between May 2011 and May 2012 (see Figure 10).

We mean “selected solution” as the solution (individual) of the population of generation g that would be returned by the proposed method if the genetic algorithm ended at generation g. Figures 11, 12 and 13 depict the obtained values for each objective of the selected solutions during 12 hours of execution. In these figures are also used the average results obtained after five runs of the proposed method. We can note that only after 8 hours we can get a good solution - about 40 production days, between 15 and 35% of the tasks performed in internal production centers and downtime rate of internal production centers near to 18%.

We highlight that the genetic algorithm parameters were adjusted considering an execution time of 12 hours. A genetic algorithm (GA) that works with a large population takes longer to found a good solution than a GA with a small population. However, it explores a larger solution space, thus obtaining better solutions. If a smaller execution time is required for running the proposed method, we should adjust the GA parameters in order to find good quality solutions.

We can also see in Figures 11, 12 and 13 that the objectives “to minimize the makespan” and “to minimize the internal production centers downtime” are not conflicting, i.e., when the value of one objective has an improvement, the value of the other also tends to improve. The objective “to maximize the use of internal production centers” has conflict with the others objectives.

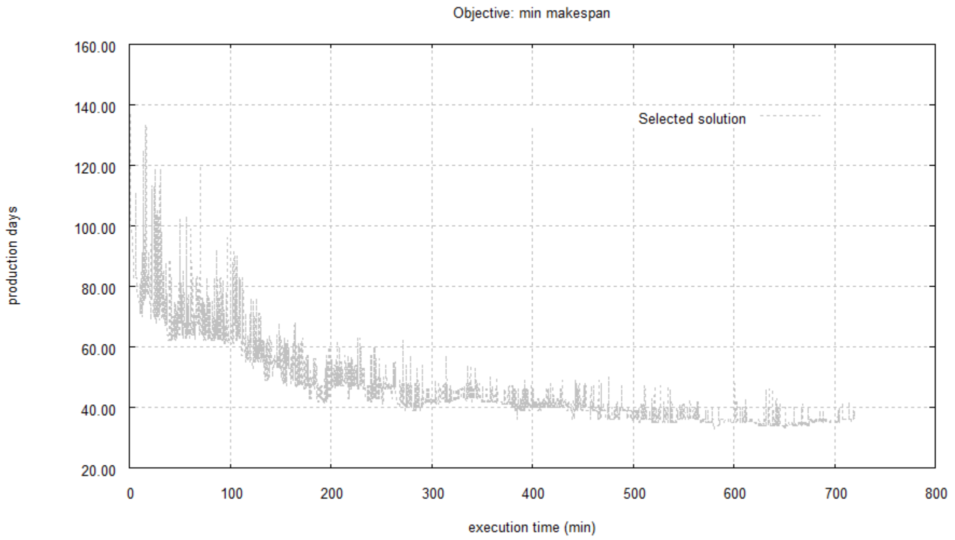


Figure 11. Selected solutions during 12 hours of execution. Objective: to minimize the makespan.

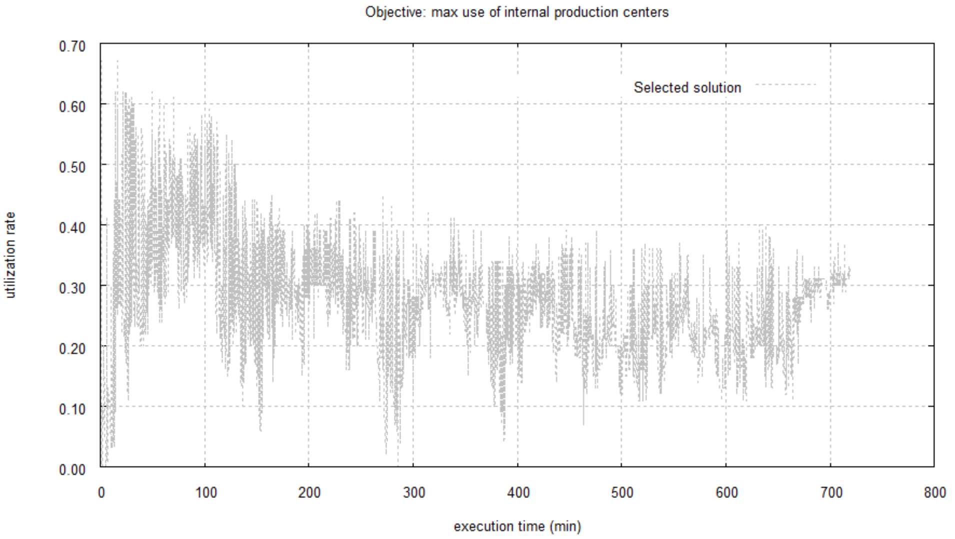


Figure 12. Selected solutions during 12 hours of execution. Objective: to maximize the use of internal production centers.

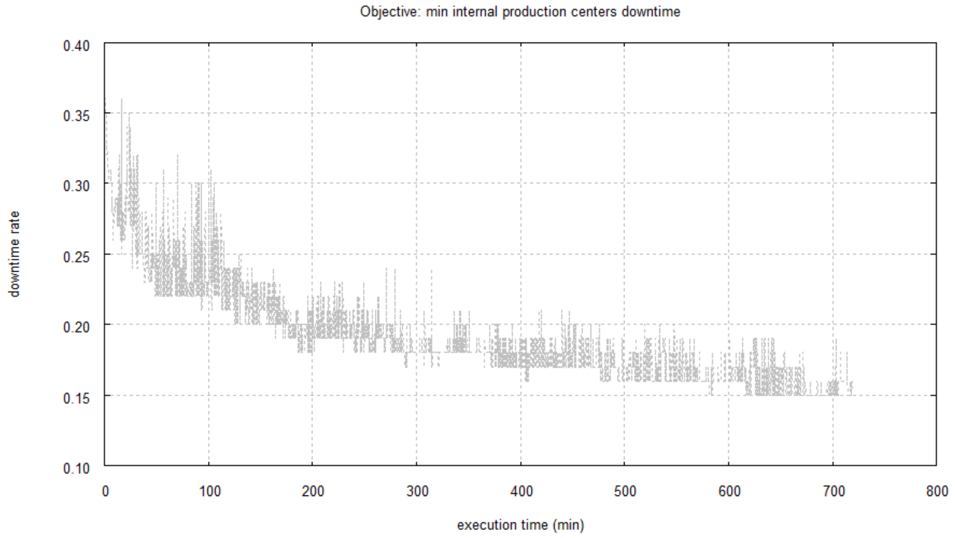


Figure 13. Selected solutions during 12 hours of execution. Objective: to minimize the internal production centers downtime.

Figure 14, 15 and 16 depict the obtained values for each selected solution and also for the best and worst solutions of the population at each generation. By analyzing the graphs presented in these figures, we can realize the diversification of the population throughout the generations. Again we can note that the objective “to maximize the use of internal production centers” (Figure 15) is conflicting with the other two. While the selected solutions tend to get close to the best solutions for the other two objectives (Figures 14 and 16), for this objective the selected solutions tend to get close to the worst solutions.

In the second experiment, the proposed method was compared with the commercial application PREACTOR, which is the leading software in the sector of finite capacity production planning in Brazil and worldwide, with over 4500 customers in 67 countries [26]. However, it was necessary to execute the proposed method considering only the objective "to minimize the makespan", because it was not possible to configure PREACTOR for working with three objectives.

Although the proposed method and PREACTOR perform the task scheduling, they have different purposes. PREACTOR is a universal tool of finite capacity production planning, which uses priority rules to perform the scheduling. The users of this tool can interact with the generated production planning. The proposed method is specific for garment companies, in which the large number of tasks makes difficult a manual evaluation. The purpose of the comparison between these methods is to validate the scheduling obtained by our proposal.

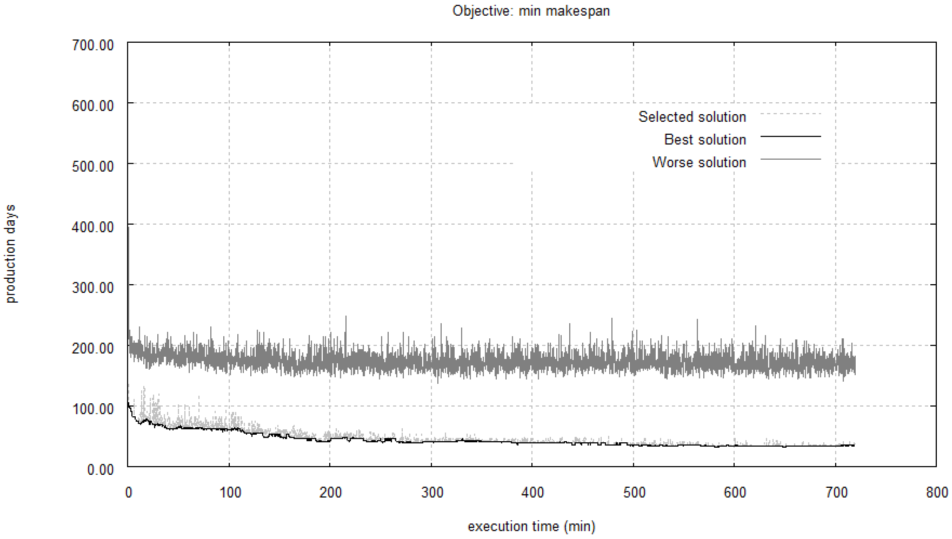


Figure 14. Best, worst and selected solutions during 12 hours of execution. Objective: to minimize the makespan.

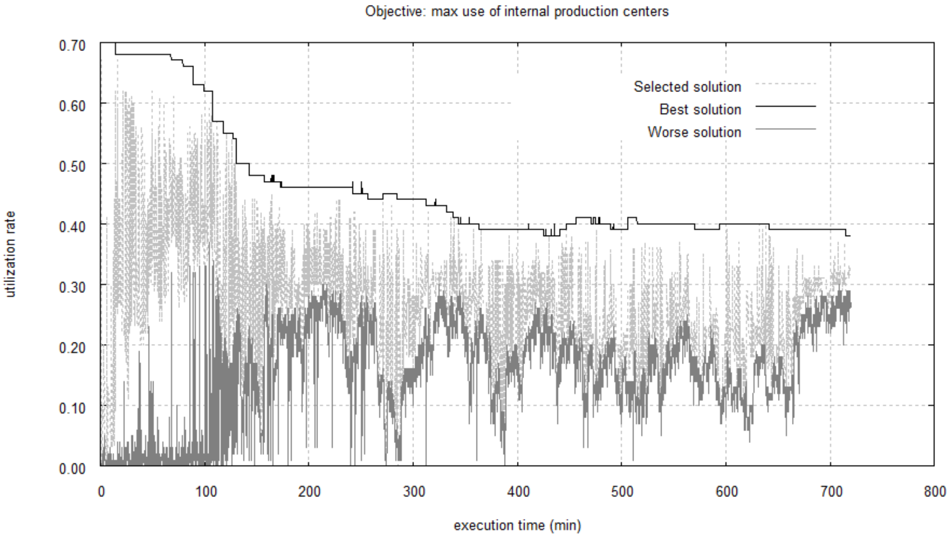


Figure 15. Best, worst and selected solutions during 12 hours of execution. Objective: to maximize the use of internal production centers.

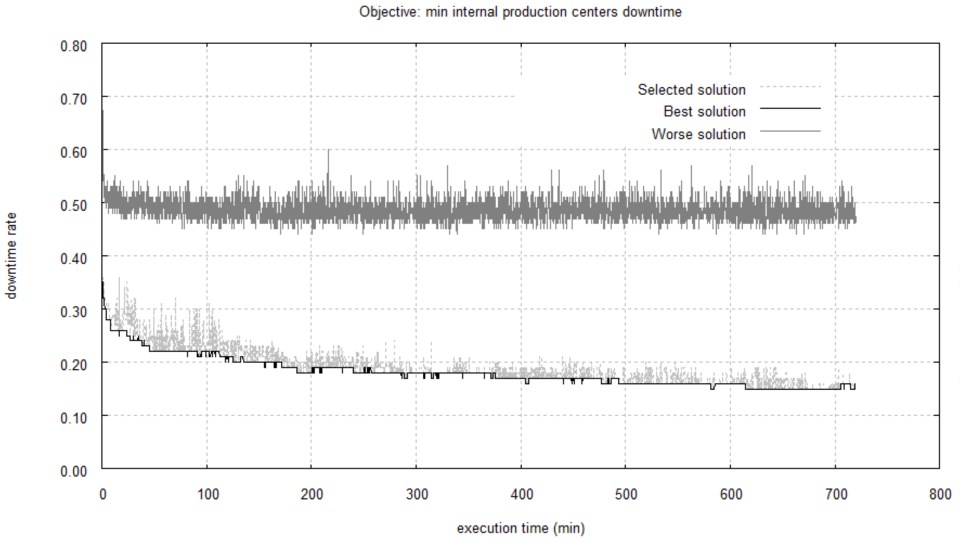


Figure 16. Best, worst and selected solutions during 12 hours of execution. Objective: to minimize the internal production centers downtime.

In this experiment, five runs of the proposed method were done. After 12 hours of execution, the proposed method has obtained an average of 32 days production planning, 17.9% lower than the 39 days production planning proposed by PREACTOR. The worst result obtained by the proposed method was 33 days production planning. It is worth to mention that PREACTOR took 12 minutes to reach its result. Figure 17 depicts that the proposed method overcomes the result obtained by PREACTOR after about 100 minutes of execution.

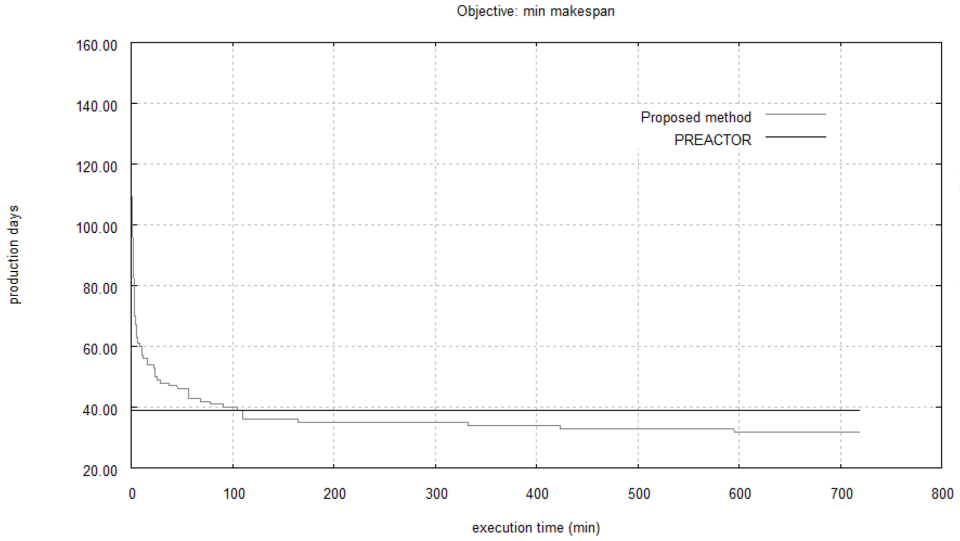


Figure 17. Proposed method × PREACTOR.

5. Conclusion remarks

The objective of this work is to develop a method to carry out the production scheduling of a Brazilian garment company in real time. Three objectives were considered: to minimize the total production time; to maximize the use of internal production centers; and to reduce the downtime of the internal production centers.

To achieve this goal, initially we have defined a mixed integer programming model for the addressed problem. Based on this model, we have proposed a method that combines a NSGA-II multiobjective genetic algorithm with the multicriteria method Weighted Sum Model - WSM. The mathematical model, the multiobjective genetic algorithm developed and its automatic combination with the multicriteria method WSM are contributions of this work.

Computational experiments were done in order to evaluate the proposed method. It was used real data provided by the analyzed garment company, which are related to May 2012 production demand. In the first experiment, the average results obtained by the proposed method were compared with the results manually obtained by the analyzed company at May 2011. Even with the increase of the production between these periods, the proposed method has decreased of 16.3% the production days. It has also got a higher rate of use and a smaller downtime rate of internal production centers. We have highlighted that the proposed method can obtain good quality solutions even when a smaller execution time is available. However, it is necessary to make an adjustment of the genetic algorithm parameters considering the available execution time.

In the second experiment, the proposed method was compared with the commercial software PREACTOR, considering only the objective "to minimize the makespan". The average obtained result was 17.9% better than the one obtained by PREACTOR. It was also shown that the proposed method overcame the result obtained by PREACTOR after about 100 minutes of execution.

It is worth to mention another advantage of the proposed method: as it is based on NSGA-II model, we can easily add and remove objectives. To do that a slight modification in the procedure that evaluates solutions is necessary. Thus, the proposed method can be suited to new needs of the garment industry or to other industrial branches.

Author details

Dalessandro Soares Vianna[1*], Igor Carlos Pulini[2] and Carlos Bazilio Martins[3]

*Address all correspondence to: dalessandrosoares@yahoo.com.br

1 Fluminense Federal University, Department of Computation – RCM, Rio das Ostras, RJ, Brazil

2 Candido Mendes University, Candido Mendes Research Center – CEPECAM, Campos dos Goytacazes, RJ, Brazil

3 Fluminense Federal University, Department of Computation – RCM, Rio das Ostras, RJ, Brazil

References

[1] C. M. Vigna and D. I. Miyake, "Capacitação das operações internas para a customização em massa: estudos de caso em indústrias brasileiras (in portuguese)", Produto & Produção, Bd. 10, Nr. 3, pp. 29--44, outubro 2009.

[2] D. F. Tubino, Planejamento e Controle da Produção (in portuguese), 2nd Edition, Atlas, 2009.

[3] R. Lindem, Algoritmos genéticos: uma importante ferramenta da inteligência computacional (in portuguese), 2nd Edition, Rio de Janeiro: Brasport, 2008.

[4] K. Deb, Multiobjective optimization using evolutionary algorithms, John Wiley & Son, 2008.

[5] J. E. C. Arroyo, P. S. Vieira and D. S. Vianna, "A GRASP algorithm for the multi-criteria minimum spanning tree problem", Annals of Operations Research, V. 159, pp. 125-133, 2008.

[6] M. Ehrgott and X. Gandibleux, "A survey and annotated bibliography of multiobjective combinatorial optimization", OR Spektrum, V. 22, pp. 425-460, 2000.

[7] D. S. Vianna, J. E. C. Arroyo, P. S. Vieira and R. R. Azeredo, "Parallel strategies for a multi-criteria GRASP algorithm", Produção (São Paulo), V. 17, pp. 1--12, 2007.

[8] D. F. Jones, S. K. Mirrazavi and M. Tamiz., "Multi-objective metaheuristics: An overview of the current state-of-art", European Journal of Operational Research, V. 137, pp. 1-19, 2002.

[9] E. Triantaphyllou, Multi-criteria decision making: a comparative study, Dordrecht: Kluwer Academic Publishers (now Springer), 2000.

[10] M. L. Pinedo, Scheduling: Theory, Algorithms, and Systems, 3rd Edition, Springer, 2008.

[11] P. Senthilkumar and S. Narayanan, "GA Based Heuristic to Minimize Makespan in Single Machine Scheduling Problem with Uniform Parallel Machines", Intelligent Information Management, V. 3, pp. 204-214, 2011.

[12] B. Santosa, M. A. Budiman and S. E. Wiratino, "A Cross Entropy-Genetic Algorithm for m-Machines No-Wait Job-Shop Scheduling Problem", Journal of Intelligent Learning Systems and Applications, V. 3, pp. 171-180, 2011.

[13] T. F. Abdelmaguid, "Representations in Genetic Algorithm for the Job Shop Scheduling Problem: A Computational Study", Journal Software Engineering & Applications, V. 3, pp. 1155-1162, 2010.

[14] L. Dayou, Y. Pu and Y. Ji, "Development Of A Multiobjective GA For Advanced Planning And Scheduling Problem", The International Journal of Advanced Manufacturing Technology, V. 42, pp. 974-992, 2009.

[15] W. S. Chang and C. C. Chyu, "A Multi-Criteria Decision Making for the Unrelated Parallel Machines Scheduling Problem", Journal Software Engineering & Applications, pp. 323-329, 2009.

[16] I. S. Franco, "Algoritmos híbridos para a resolução do problema de job shop flexível (in portuguese)", Master Thesis. Universidade Candido Mendes, Campos dos Goytcazes, RJ - Brazil, 2010.

[17] K. Deb, A. Pratap, S. Agarwal and T. Meyarivan, "A Fast and Elitist Multiobjective Genetic Algorithm: NSGA II", IEEE Transactions on Evolutionary Computation, V. 6, N. 2, pp. 182-197, April 2002.

[18] M. Köksalan, J. Wallenius and S. Zionts, Multiple Criteria Decision Making: From Early History to the 21st Century, Singapore: World Scientific, 2011.

[19] J. E. C. Arroyo, "Heurísticas e Metaheurísticas para Otimização Combinatória Multiobjetivo (in portuguese)", PhD Thesis, UNICAMP, SP - Brazil, 2003.

[20] A. S. Pitombeira-Neto, "Modelo híbrido de otimização multiobjetivo para formação de células de manufatura (in portuguese)", PhD Thesis, USP-São Carlos, SP - Brazil, 2011.

[21] H. Zhang and M. Gen, Effective Genetic Approach for Optimizing Advanced Planning and Scheduling in Flexible Manufacturing System, Seatle, 2006.

[22] Y. Li and Y. Chen, "A Genetic Algorithm for Job-Shop Scheduling", Journal Of Software, V. 5, pp. 269-274, 2010.

[23] A. Okamoto, M. Gen and M. Sugawara, "Robust Scheduling for APS using Multobjective Hybrid Genetic Algorithm", in proceedings of Asia Pacific Industrial Engineering and Management Systems Conference, Bangkok, Thailand, 2006.

[24] Z. Michalewicz, Genetic Algorithms + Data Structures = Evolution Programs, Springer, 1998.

[25] L. M. C. Dias, L. M. A. T. Almeida and J. Clímaco, Apoio multicritério à decisão (in portuguese), Coimbra: Universidade de Coimbra, 1996.

[26] TECMARAM, "Soluções para o gerenciamento de sistemas de produção (in portuguese", [Online]. Available: http://www.tecmaran.com.br. [Acessed at 04/08/2012].

[27] M. R. Garey and D. S. Johnson, Computers and Intractability: A Guide to the Theory of NP-completeness, San Francisco: W. H. Freeman, 1979.

[28] M. R. R. Olazar, "Algoritmos Evolucionários Multiobjetivo para Alinhamento Múltiplo de Sequências Biológicas (in portuguese)", PhD Thesis, UFRJ, RJ - Brazil, 2007.