Optimization in hospital logistics : scheduling production processes in hospital catering

Fatima Abderrabi, Matthieu Godichaud, Alice Yalaoui, Farouk Yalaoui, Lionel Amodeo, Ardian Qerimi, Eric Thivet

Optimization in hospital logistics : scheduling production processes in hospital catering

F. Abderrabi [*, **]
M. Godichaud [*], **A. Yalaoui** [*], **F. Yalaoui** [*], **L. Amodeo** [*]
A. Qerimi [**], **E. Thivet** [**]

[*] *University of Technology of Troyes, Logistics Industrial Systems Optimization (LOSI), ICD, Troyes, France*
(e-mail: fatima.abderrabi@utt.fr, matthieu.godichaud@utt.fr, alice.yalaoui@utt.fr, farouk.yalaoui@utt.fr, lionel.amodeo@utt.fr)
[**] *Hospital Center of Troyes, Troyes, France*
(e-mail: fatima.abderrabi@ch-troyes.fr, ardian.qerimi@ch-troyes.fr, eric.thivet@ch-troyes.fr)

Abstract

Nowadays, hospital logistics has become an essential component of healthcare institutions. It allows the synchronization of all the flows inside a hospital to ensure the efficiency of the health care system. For many years, the management was commonly focused on improving the quality of medical care, while less attention was usually devoted to operations management. In recent years, the need of considering the costs while increasing the competitiveness along with the new policies of national health service hospital financing forced hospitals to necessarily improve their operational efficiency. In this work, we focus on a real optimization problem of the supply chain of the hospital center of Troyes (HCT). The HCT is currently seeking to review and improve its logistics processes. The implementation of techniques and methods of operational research must provide solutions to improve the efficiency of logistics activities. The present work focuses on developing operational decision support models and algorithms for scheduling of production processes in hospital catering. The aim is to optimize and better organize the daily work of employees of the catering service of HCT, by supporting them with automated tools for their daily decision making. A novel mathematical model and different metaheuristics for the production scheduling of multi-products and multi-stages food processes are developed. The computational results of these methods have proven their effectiveness for scheduling operations in the food production processes.

Keywords : hospital logistics, hospital catering, scheduling production process, mathematical model, flexible job shop scheduling, sequence-dependent setup time, job-splitting, genetic algorithm, local search methods, iterated local search algorithm.

1 Introduction

The efficient use of resources and the search for optimal patient service stimulates logistical thinking in hospitals. The difficulties of optimising flows are leading managers to find difficult balances and to discover new ways for rationalizing expenditure and seeking refined solutions to these new problems. In

this context, logistics allows the hospital to reduce the financial impact of product consumption, reduce inventory, limit waste and provide better inventory tracking and trace-ability of service products. On the other hand, the increase in hospital spending leads to worrying deficits. The obligation to control these expenses is therefore in itself an additional and sufficient reason to undertake steps to reorganize the supply chain. These problems confirm the need to restructure the hospital logistics flows that deserve to be analysed and evaluated. Cremadez and Grateau (1997) oppose this idea by stressing that the aim of the hospital system is to ensure the well-being of the patient regardless of the cost.

In this context, the constant search for savings for health officials is a natural approach to examine how logistics activities are managed in order to increase productivity. These issues require major changes in hospital management practices by seeking profitability, lower costs, efficiency and quality of service. This requires the management of a care establishment to ensure that the resources available are used appropriately to meet the demand for care. This explains the current tendency of hospitals to try to control the entire supply chain from production sites to consumption sites.

Health facilities are required to ensure an impeccable quality of service to patients, and to optimize their supply chain. These constraints have led the hospital system to embark on a process of deep reorganization and control of the growth of health expenditure and budgets. These constraints stimulate reflections on how to use the available resources more efficiently (Landry, (2000); Blouin, (2001); Sampiere, (2004)). The deep evolution and transformations of the environment, and the economic regression lead the hospital to reformulate its health policy, which requires controlling expenditure in this sector ([Mucchielli,(1993) and Cauvin, (1997)). This transformation requires a reorganization of the supply chain to adapt to the patient's valued needs. Naylor (1999) summarizes this situation by requiring the optimization of the antithetic triad of health system objectives: quality, accessibility and cost.

Generally, the provision of care is based on various products and services that are deployed in the establishment through a set of activities that Landry et al (2002) incorporate under the term hospital logistics. These logistics activities can make a significant contribution to the performance of a hospital center including the presence of the right products at the right time to support the delivery of services. Hospital logistics is a complex process characterized by a diversity of needs, users, products and distribution channels. The main mission of this logistics is to control and optimize physical flows from suppliers to patients. It is an indispensable tool for the reorganization of chain process. This reorganization seeks the best performance at the best cost that respects technical, economic and regulatory conditions for optimal delivery to patients. According to Fixari and Tonneau 1993 management logistics activities at the hospital have very specific characteristics, which directly influence the definition of the patient's role and place in care. We could even go further by considering that hospital flows are much more critical and sensitive as those in the industrial sector since it is health, and therefore life even, patients who are at stake. On the other hand, coordinating these activities requires logistical expertise that few establishments will succeed in developing on their own (Ruiz, (2002)). This has led researchers to take an interest in the supply chain management in hospitals for several years. The evolution of the hospital sector as well as the richness and originality of the hospital environment contributed to its growth. Bonniol (1999) indicates that physical flow has always played an important role in the maintenance and delivery of health care (saving lives, preventing the spread of disease, helping to diagnose, improving the quality of life and relieving pain).

According to Gasquet (2004) the logistical functions of hospitals are evolving as much as medical and medico-technical activities, but probably in a less visible way, especially in the healthcare providers and the public. These logistical functions can not only be the source of savings in the logistics activities themselves, but can also free up resources for patient care through better support for service delivery. Whatever its field of intervention, logistics is not a science with immutable laws but an approach that

adapts its contributions from the strategic level to the operational according to its environment and defined objectives. As for other sectors of activity, the challenge ahead is for the logistics engineer to make a contribution to all of the hospital's processes. Logistics is always at the service of the missions and objectives of a company or hospital establishment project. Therefore, it naturally accompanies the evolution of a system.

The advantage of research on hospital logistics is that it is considered as a key to the evolution and optimization of the management of flows in an establishment, particularly with regard to its contribution to the core business (essential activities) of hospitals. In a period of financial constraints, logistics managers play an important role in the hospital centre because they have to develop cost reduction strategies. The study of Thevenin (1999) carried out in French healthcare establishments, underlines that major transformations of the logistic activities require a favorable agreement of those responsible for the medical and pharmaceutical functions of the establishment, the weight of the logistics department alone being insufficient. Moreover, the author notes that these transformations are not part of a one-off process but rather a more global process associated with the development of the settlement project.

To meet effectively the needs of patients and to improve working conditions and employee well-being, the hospital center of Troyes implements important measures to improve its daily efficiency. It is in this context that our study of optimization of the activities of hospital's catering service takes place. The hospital is carrying out a project to revise its supply chain, which must in particular consider the management of food flows within the hospital. Indeed, the hospital center of Troyes, as the main component of the Champagne Sud Hospitals, provides hospital centers, nursing homes and psychiatric clinics with meals (Figure 1). Our study in the central food production unit of this hospital aims to determine the best plan to meet customers' demands in terms of food flow and to propose ways to improve the well-being and working conditions of employees of the catering service of the hospital centre of Troyes. Given the complexity and scale of the flows concerned, the project to reorganize the management of food flows is the only subject of this study. Regarding physical flows, we will therefore analyze only the supply chain management for meals and we focus on the work organization, the objective is to provide methods and tools for scheduling production processes over the day.

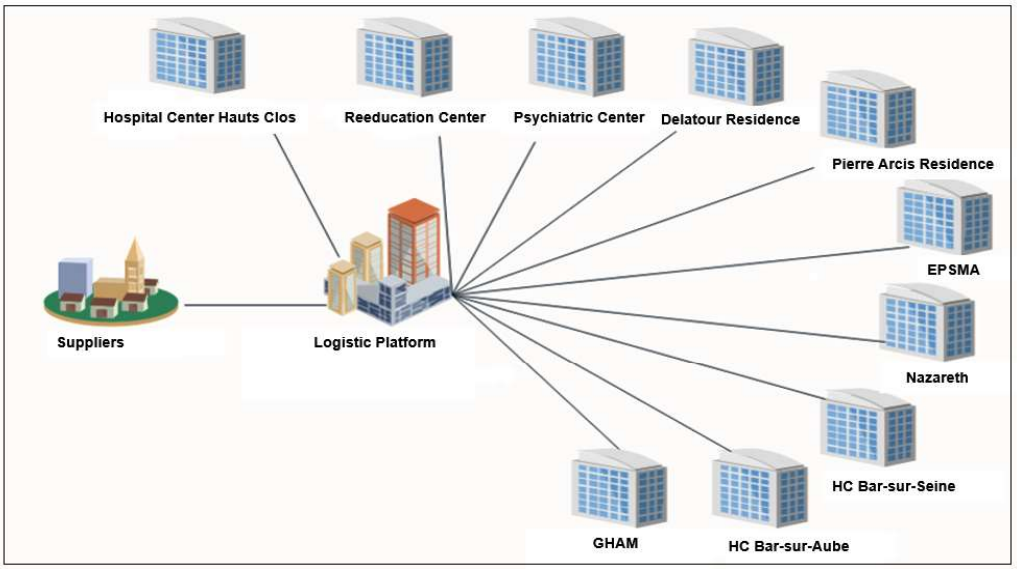

Figure 1: Logistics network of hospital center of Troyes.

The remainder of this paper is organized as follows : section 2 presents a state-of-the-art regarding the problem of scheduling food production. Then, the problem statement are defined in section 3. In section 4 the mathematical model developed for the scheduling food production problem and the computational results of this model are presented. Finally, the genetic algorithm proposed for the problem studied, the different elements of this metaheuristic and the computational results are presented in section 5.

2 Literature Review

The production scheduling problem in food industries represents a famous class of problems referred to as scheduling with sequence-dependent setups that are well known to be NP-hard (Sun et al 1999). In recent years, there has been great interest in the development of intelligent solutions for this problem in various fields of applications. The promising results of scheduling methods (such as reduction of production costs, increased throughput and smoother operation of the production equipment, improvement of working conditions and the well-being of employees) have stimulated a considerable research effort. Most of existing works in literature on scheduling food production are from the food industry and dairy industry where the production system is a flow shop or parallel machine system in the most cases (Table 1), as there is an increasing interest in investigating changeovers in scheduling approaches from this sector. Akkerman and van Donk (2009) developed a methodology for the analysis of the scheduling problems in food processing. In (1988) Smith Daniels and Ritzman develop a general lot sizing model for process industries and apply their method to a situation representative of a food processing facility. Kopanos et al (2012) offered an efficient mathematical framework for detailed production scheduling in the food processing industries. Wauters et al (2012) introduced an integrated approach to real world production scheduling for the food processing industries. In (2016) Tempelmeier and Copil considered a capacitated dynamic lot sizing problem with parallel machines for the food industry, in which all of a given product, produced during a specified time period, is used to satisfy the related demand. Niaki et al (2017) addressed the integrated lot sizing and scheduling problem of food production in batch manufacturing systems with multiple shared-common resources and proposed a new mixed integer linear programming formulation with multiple objective functions. In (2009) Ahumada and Villalobos review models for the agri-food business where products may be perishable or not, but their focus is on procurement and harvesting planning and the only goods they are interested in are crops. Sel et al (2015) introduced the planning and scheduling decisions considering of the shelf-life restrictions, product dependent machine speeds, demand due dates, regular and overtime working hours in the perishable supply chain. In (1999) Arbib et al consider a three-dimensional matching model for perishable production scheduling, which is studied under two independent aspects: the relative perishability of products and the feasibility of launching/completion time. Basnet et al (1999) have described an exact algorithm to solve scheduling and sequencing problem in the same industry. Chen et al (2019) provided a review of literature on the integration of scheduling and lot sizing for perishable food products and they categorized the papers by the characteristics of lot-sizing and scheduling that were included in their models, and the strategies used to model perishability. In (1993) Claassen and Van Beek propose an approach to solve a planning and scheduling problem for the bottleneck packaging facilities of the cheese production division of a large dairy company. Nakhla (1995) emphasizes the need for flexibility for operations scheduling in the dairy industry, and proposes a rule-based approach for scheduling packaging lines. In (2005) Entrup et al presented three different mixed integer linear programming for scheduling problems in fresh food industry in the packing stage of stirred yogurt production. They accounted for shelf life issues and fermentation capacity limitations. Marinelli et al (2007) addressed a solution approach for a capacitated lot sizing and scheduling problem with parallel machines and shared buffers, arising in a packaging company producing yoghurt. In (2007) Doganis and Sarimveis propose a model that aims the optimal production scheduling in a single yoghurt production line. The

model takes into account all the standard constraints encountered in production scheduling (material balances, inventory limitations', machinery capacity). It also considers special features that characterize yoghurt production which are limitations in production sequencing mainly due to different fat contents and flavors of various products and sequence dependent setup times and costs. However the model is limited to single production line. In another study, Doganis and Sarimveis (2008) present a methodology for optimum scheduling of yoghurt packaging lines that consist of multiple parallel machines. The methodology incorporates features that allow it to tackle industry specific problems, such as multiple intermediate due dates, job mixing and splitting, product specific machine speed, minimum, maximum lot size and sequence dependent changeover times and costs. However the model does not incorporate multi-stage production decisions, and ignores some industry-specific characteristics, such as shelf life. Finally, it is worth mentioning that, to the best of our knowledge, there is almost no study addressing the problem of scheduling food production in hospital catering. Therefore, the aim of the present work is to propose a new mathematical model for this problem.

Author	Year	Product	Nb Products	Production System	Modeling	Domain
Our problem	2019	-	Multi products	Flexible Job Shop	MLP	Hospital catering
Wei et al.	2018	-	Multi products	Flow Shop	MILP	Food industry
Sargut and Isık	2017	-	Single product	Single machine	-	Food industry
Tempelmeier et al.	2016	-	Multi products	Parallel machine	-	Food industry
Stefansdottir et al.	2016	Cheese	Single product	Flow Shop	MILP	Dairy industry
Acevedo-Ojeda et al.	2015	-	Single product	Single machine	MIP	Food industry
Bilgen and Çeleb	2013	-	Multi product	Flow Shop	MILP	Dairy industry
Kopanos et al.	2012	Ice cream	Single product	Flow Shop	MILP	Food industry
Kilic et al.	2011	Milk	Single product	Flow Shop	MILP	Dairy industry
Karray et al.	2011	-	Multi products	Single machine	ILP	Food industry
Kopanos et al.	2010	Yogurt	Single product	Flow Shop	MILP	Dairy industry
Günther et al.	2006	Sausage	Single product	Flow Shop	MILP	Food industry

MLP: Mixed linear programming, MILP : Mixed integer linear programming, ILP : Integer linear programming

Table 1: Bibliographic summary on food production scheduling problems.

In this work, the production system studied is considered as flexible job shop system. In most of the existing work in the literature on flexible job shop problems, the criterion to optimize is the minimisation of the makespan, but the minimisation of flow time which is the target criterion in this work is very little studied. In (2015) Chaudhrya and Khanb published a literature review on the methods used to solve the flexible job shop problems and the different objective functions studied (Figure 2).

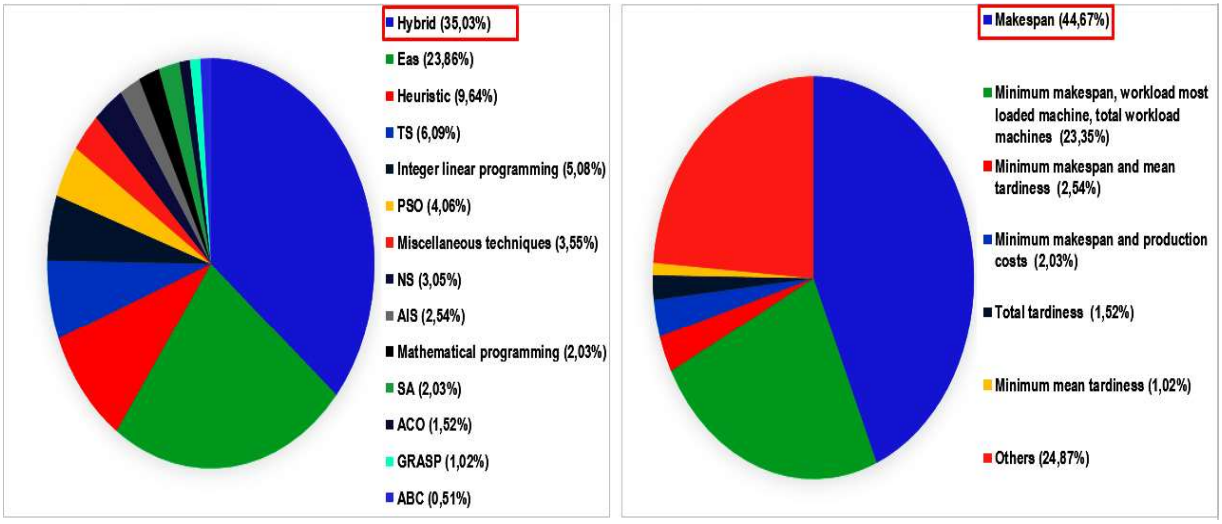

Figure 2: Bibliographic summary on flexible job shop problem resolution methods.

3 Problem description

In the problem of scheduling food production, we were particularly interested in the process from the pretreatment of the raw materials until the cooling and stock of finished products as shown in the figure (Figure 3) which represents the different stages of the meal preparation process from the reception of raw materials until the distribution of meals to the patients.

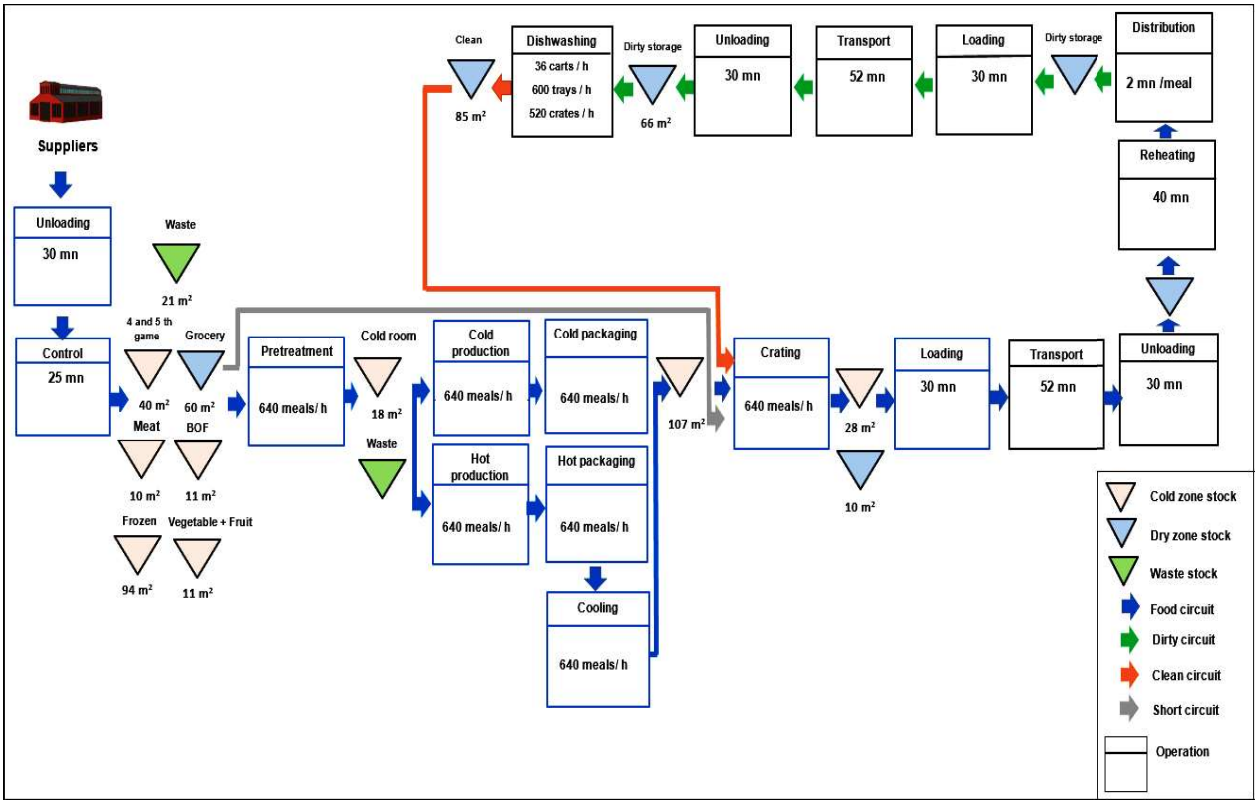

Figure 3: Representative scheme of the meal preparation process.

The problem of scheduling food production can be described by a set of N jobs, where each job i corresponds to the preparation of a dish characterized by a number of portions Q_i (quantity), and a set of operations J_i necessary for the preparation of the dish (from raw material to finished product). It is worth to highlight that the dishes to be prepared do not have the same operating ranges (set of operations necessary for the preparation of dish). In this study, we identified eight possible operating ranges (Figure 4) for all the dishes to be prepared and it is possible that several dishes may have the same operating range.

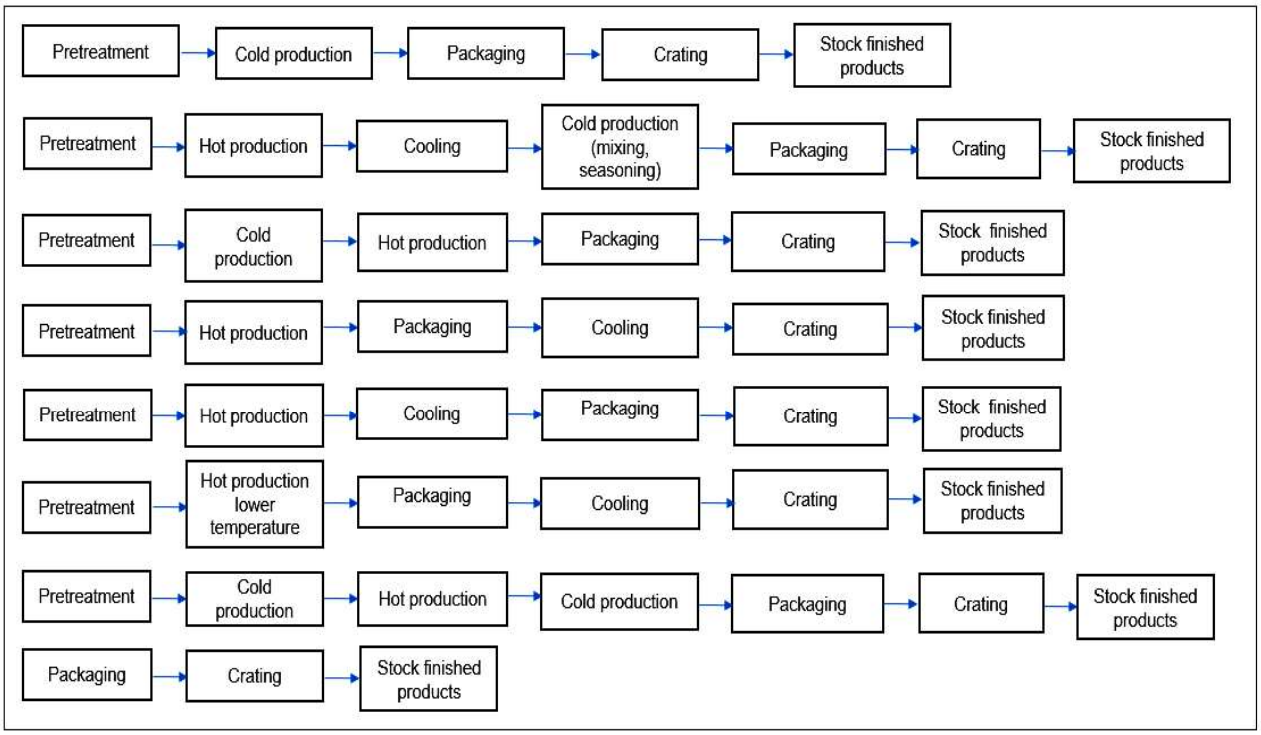

Figure 4: Set of production operating ranges for the preparation of dishes.

For each operation of an operating range there is a set of material resources that can realize it. Among these material resources, we cite as an example : ovens, packaging machines, cooling cells, etc. For each material resources there is a setup time to take into account which corresponds to the preparation time of the resource before carrying out an operation and the cleaning time of the resource between two consecutive operations. The problem of scheduling food production treated in this study is considered as a flexible job shop scheduling with sequence-dependent setup time. Since, the jobs do not have the same order of operations and each job has its own order of operation, and each operation is not to be processed by a predefined machine, but rather has to be assigned to one among a set of possible machines (Figure 5).

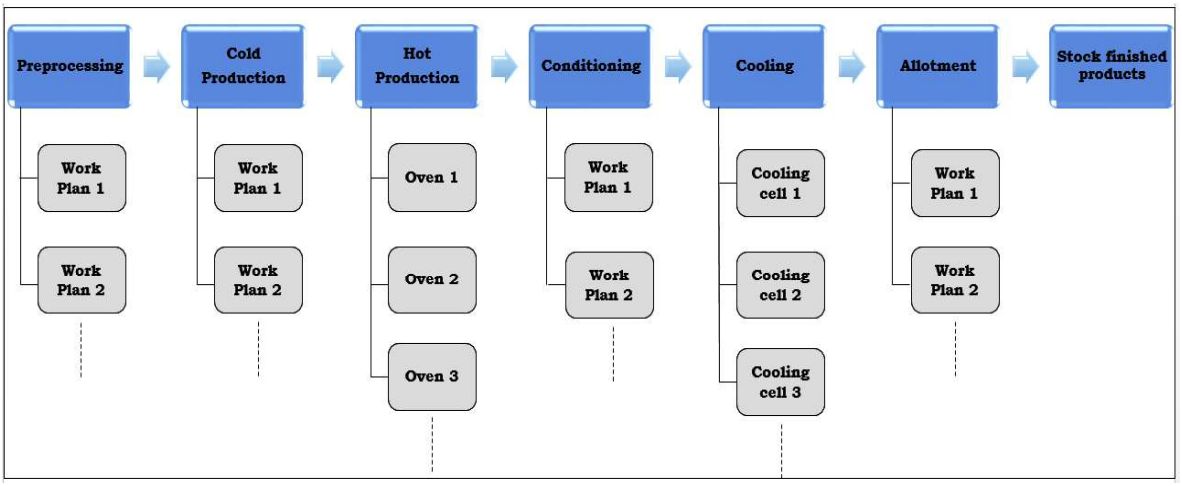

Figure 5: Example of an operating range consisting of a set of operations from pretreatment of raw materials to stock of finished products.

Note that the corresponding machines may not be identical, involving different processing times according to the chosen machine. The setup times of machines are sequence dependent because it dependents on the preceding operation on the same machine. The scheduling food production involves two steps: (i) assignment of operations to machines i.e., each operation must be assigned to a machine among those that can process the considered operation, (ii) sequencing of operations on machines i.e., determining an operation sequence for each machine.

As mentioned previously, in order to respect the production capacity of material resources, a job can be splitted into smaller sub-lots, in such a way that the operations of sub-lots of a job can be performed simultaneously on different machines. This strategy, which is useful when machine capacity does not allow the treatment of the whole job, also enables a more efficient processing scheme. The criterion to minimize in the present study is the flow time of jobs in the production system. The choice of this criterion is based on the fact that in a food process we must ensure the respect of the cold chain at each stage of the product life cycle which aims to constantly maintain a low temperature (positive or negative depending on the product) to ensure the maintenance of all the qualities (hygienic, nutritional and gustatory) of food.

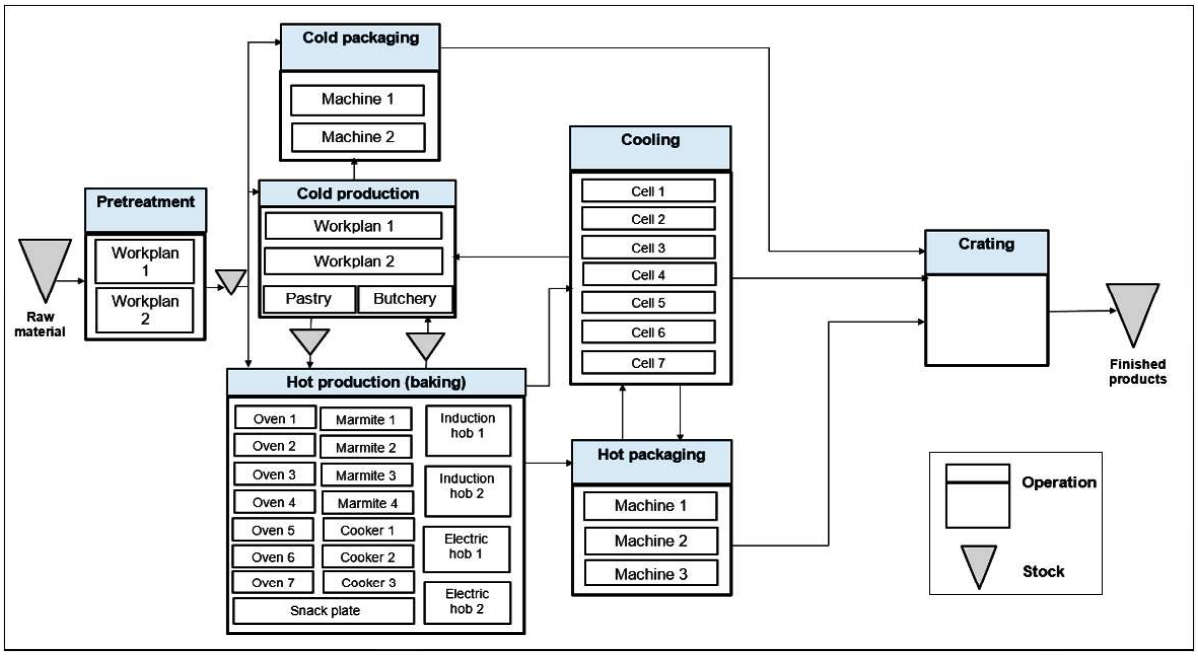

Figure 6: Operation orders of the operating ranges with the material resources that can perform each operation.

4 Mathematical model

In this section we present a mixed integer linear programming model. This model formalises the problem studied and can be used to solve small-sized problems using a branch-and-bound algorithm. Solutions from such small-sized problems can be used to validate the correctness of the developed metaheuristic methods.

4.1 Assumptions

The mathematical model for the scheduling food production inherits its main assumptions from the standard flexible job shop scheduling problem and flexible job shop scheduling problem with sequence-dependent setup times, in addition to some specific features due to the job splitting :

- Jobs are independent of each other,
- A job can be split into sub-lots,
- The sub-lots of jobs can be grouped on the machines to be treated at the same time,
- Each sub-lot of a job consists of a set of operations that must be processed consecutively (precedence constraints between operations of sub-lots of jobs),
- Each operation of sub-lot has a given processing time,
- The preemption of operations of sub-lots of jobs is not allowed, i.e. operation processing on a machine cannot be interrupted,
- Each job has a given due date (finish date of production at latest),
- Sub-lot sizes (number of portions) are discrete,
- Sub-lots creation is consistent throughout the processing sequence, meaning that job splitting and sub-lot sizes remain constant for all operations,

- Machines are independent,
- A machine can process at most one operation at a time,
- The setup times of machines are dependent on the sequence of operations of sub-lots of jobs,
- Material resource has a given availability time windows that must be taken into account.

Accounting for these assumptions, the objective is to find a schedule involving sub-lots assignment to machines and sub-lot sequencing for each machine, in such a way that each job's demand is fulfilled, different constraints of problem are respected and the flow time of jobs in production system is minimized.

4.2 Notations

The definition of the proposed mathematical model parameters relies on the following sets and indexes :

- M : set of all material resources, where $m = |M|$.
- N : set of jobs (dishes to prepare), where $n = |N|$ and $\{0, n+1\}$ are two dummy jobs.
- J_i : set of operations of job $i \in N$, such that the operation $j \in J_i$ is done before the operation $j+1 \in J_i$ and $|J_0| = |J_{n+1}| = 1$.
- Q_i : number of portions (quantity) of job $i \in N$.
- q_i : number of portions in each sub-lot of job $i \in N$.
- L_i : set of sub-lot of job $i \in N$, with $|L_0| = |L_{n+1}| = 1$ and $l_i = |L_i|$ such that $l_i = \lceil \frac{Q_i}{q_i} \rceil$.
- d_i : due date of job $i \in N$.
- $M_{ij} \subset M$: set of material resources that can perform the operation $j \in J_i$ of job $i \in N$.
- R_k : maximum capacity in number of portions of the material resource $k \in M$.
- $M_1 \subset M$: set of material resources that have a capacity of one portion and that can not be processed several jobs at the same time (material resources that can perform preprocessing and cold production operations).
- $M_2 \subset M$: set of material resources that have a capacity greater than one portion and which can not be processed several jobs at the same time (ovens,...).
- $M_3 \subset M$: set of material resources that have a capacity greater than one portion and that can processed several jobs at the same time (cooling cells).
- P'_{ijk} : unit processing time of operation $j \in J_i$ of job $i \in N$ on the material resource $k \in M_1$.
- P_{ijk} : processing time of operation $j \in J_i$ of job $i \in N$ on the material resource $k \in M_2 \cup M_3$.
- s_{ijhgk} : setup time of material resource $k \in M_{ij} \cap M_{hg}$, if operation $j \in J_i$ of job $i \in N$ precedes directly operation $g \in J_h$ of job $h \in N$ on the material resource $k \in M_{ij} \cap M_{hg}$.
- T_k : preparation time of the material resource k at the beginning of scheduling.
- E_k : preparation time of the material resource k at the end of scheduling.
- $[A_k, Y_k]$: time window of availability of material resource $k \in M$.
- B : big integer.

4.3 Decision variables

- X_{iljk} : binary variable, equals to 1, if operation $j \in J_i$ of sub-lot $l \in L_i$ of job $i \in N$ is assigned to the material resource $k \in M_{ij}$, 0 otherwise.
- $F_{iljhl'gk}$: binary variable, equals to 1, if operation $j \in J_i$ of sub-lot $l \in L_i$ of job $i \in N$ precedes directly operation $g \in J_h$ of sublot $l' \in L_h$ of job $h \in N$ on the material resource $k \in M_{ij} \cap M_{hg}$, 0 otherwise.
- $Z_{ill'jk}$: binary variable, equals to 1, if operation $j \in J_i$ of sub-lot $l \in L_i$ of job $i \in N$ starts and finishes at the same time as the operation $j \in J_i$ of sub-lot $l' \in L_i$ of job $i \in N$ on the material resource $k \in M_{ij}$, 0 otherwise.
- S_{iljk} : starting time of operation $j \in J_i$ of sub-lot $l \in L_i$ of job $i \in N$ on the material resource $k \in M_{ij}$.
- C_{iljk}: completion time of operation $j \in J_i$ of sub-lot $l \in L_i$ of job $i \in N$ on the material resource $k \in M_{ij}$.
- C_i: completion time of job $i \in N$.

4.4 Mathematical model

The mathematical model (P2) provided here for the problem of scheduling food production was developed based on the (Buddala and Mahapatra, 2018) formulation (P1), that is designed for the basic flexible job shop scheduling problem. This mathematical model was adapted and improved for the problem of scheduling food production by integrating the different constraints that were not taken into consideration in the work of Buddala and Mahapatra (2018). The following table represents the characteristics of the two mathematical models (P1) and (P2) :

Constraints and objective	(P1)	(P2)
- Processing times for job operations	X	X
- Precedence constraints between job operations	X	X
- Non preemption of job operations	X	X
- Machine production capacity (number of operations processed at the same time)	X	X
- Machine production capacity (in number of portions)		X
- Due date of jobs		X
- Machine availability time windows		X
- Sequence-dependent setup time of machines		X
- Splitting of jobs into sub-lots		X
- Batching of sub-lots of jobs on machines		X
- C_{max}	X	
- $\sum C_i$		X

Table 2: Characteristics of the mathematical models (P1) and (P2).

The mathematical model (P2) is formulated as indicated through equations (1) to (22) :

$$Min \sum_{i \in N} C_i \tag{1}$$

$$C_i \geq \sum_{k \in M_{ij}} C_{iljk}, \ \ \forall\, i \in N,\ l \in L_i,\ j \in J_i \tag{2}$$

$$S_{iljk} + C_{iljk} \leq B * X_{iljk}, \forall\, i \in N, l \in L_i, j \in J_i, k \in M_{ij} \tag{3}$$

$$C_{iljk} - S_{iljk} \geq P_{ijk} - B * (1 - X_{iljk}), \forall\, i \in N, l \in L_i, j \in J_i, k \in M_2 \cup M_3 \tag{4}$$

$$C_{iljk} - S_{iljk} \geq P'_{ijk} * Q_i - B * (1 - X_{iljk}), \forall\, i \in N, l \in L_i, j \in J_i,\ k \in M_1 \tag{5}$$

$$S_{hl'gk} \geq C_{iljk} + s_{ijhgk} - B * (1 - F_{iljhl'gk}), \forall\, i \in N, l \in L_i, j \in J_i, h \in N, l' \in L_h, g \in J_h, k \in M_{ij} \cap M_{hg} \backslash M_3 \tag{6}$$

$$S_{il'jk} \geq C_{iljk} + s_{ijijk} - B * (1 - F_{iljil'jk}) - B * Z_{ill'jk}, \forall i \in N, l, l' \in L_i, j \in J_i, k \in M_{ij} \cap M_{hg} \tag{7}$$

$$S_{iljk} - S_{il'jk} \leq B * (1 - Z_{ill'jk}), \forall i \in N, l, l' \in L_i, j \in J_i, k \in M_2 \cup M_3 \tag{8}$$

$$C_{iljk} - C_{il'jk} \leq B * (1 - Z_{ill'jk}), \forall i \in N, l, l' \in L_i, j \in J_i, k \in M_2 \cup M_3 \tag{9}$$

$$\sum_{h \in N \backslash \{0\}} \sum_{l' \in L_h} \sum_{g \in J_h} F_{iljhl'gk} = 1, \forall i \in N \backslash \{n+1\}, l \in L_i, j \in J_i, k \in M_{ij} \cap M_{hg} \tag{10}$$

$$\sum_{i \in N \backslash \{n+1\}} \sum_{l \in L_i} \sum_{j \in J_i} F_{iljhl'gk} = 1, \forall h \in N \backslash \{0\}, l' \in L_h, g \in J_h, k \in M_{ij} \cap M_{hg} \tag{11}$$

$$\sum_{k \in M_{ij}} S_{iljk} - \sum_{k \in M_{ij-1}} C_{ilj-1k} \geq 0,\ \forall\, i \in N,\ l \in L_i,\ j \in J_i \tag{12}$$

$$\sum_{k \in M_{ij}} X_{iljk} = 1, \ \ \forall\ \ i \in N, \ \ l \in L_i, \ \ j \in J_i \tag{13}$$

$$C_i \leq d_i, \ \ \forall i \in N \tag{14}$$

$$\sum_{l, l' \in L_i} q_i * Z_{ill'jk} \leq R_k, \ \ \forall i \in N, \ \ j \in J_i, k \in M_2 \cup M_3 \tag{15}$$

$$S_{iljk} \geq A_k + T_k,\ \forall\, i \in N, \ \ l \in L_i, \ \ j \in J_i, \ \ k \in M_{ij} \tag{16}$$

$$C_{iljk} \leq Y_k + E_k,\ \forall\, i \in N, \ \ l \in L_i, \ \ j \in J_i, \ \ k \in M_{ij} \tag{17}$$

$$Z_{ill'jk} = 0,\ \forall\, i \in N, l,\ \ l' \in L_i,\ \ j \in J_i,\ \ k \in M_1 \tag{18}$$

$$X_{iljk} \in \{0,1\},\ \forall\, i \in N,\ \ l \in L_i,\ \ j \in J_i,\ \ k \in M_{ij} \tag{19}$$

$$Z_{ill'jk} \in \{0,1\},\ \forall\, i \in N, l, l' \in L_i, j \in J_i, k \in M_{ij} \tag{20}$$

$$F_{iljhl'gk} \in \{0,1\},\ \forall\, i \in N, l \in L_i, j \in J_i, h \in N, l' \in L_h, g \in J_h, k \in M_{ij} \cap M_{hg} \tag{21}$$

$$S_{iljk} \geq 0, C_{iljk} \geq 0, C_i \geq 0, \forall i \in N, l \in L_i, j \in J_i, k \in M_{ij} \tag{22}$$

In the mathematical model presented previously, the first constraint (1) represents the objective function consisting in minimizing of the flow time of jobs in production system, which is defined as the sum of completion time of all jobs. In turn, job completion times are computed as the completion time of the last sub-lot derived from the considered job, as indicated in (2). Note that, due to (3), for given $i \in N$, $l \in L_i$ and $j \in J_i$, variables S_{iljk} and C_{iljk} are equal to zero for all $k \in M_{ij}$ values different from the index of the machine that really processes the considered sub-lot. On the other hand, when X_{iljk} equals to 1, (4) and (5) activate the relationship constraint between starting time and completion time of an operation of a sub-lot. It is worth noting that, in this case, the processing time does not depend on the quantity of job for the material resources $M_2 \cup M_3$ (4), but it depends on the quantity of job for the material resources M_1 (5). Constraints (6) considers sequence dependent setup times between completion time and starting time of two operations of sub-lots which are processed on machine one after another. Equations (7) disable the constraints (6) if two different sub-lot of the same job are performed at the same time by the same material resource. Constraints (8) and (9) require that if two operations of two different sub-lots of the same job are assigned at the same time to a material resource $M2$ or $M3$, they must have the same starting time and completion time respectively. Constraint (10) ensures that only one operation follows immediately j th operation of sub-lot $l \in L_i$ of job $i \in N$ on machine $k \in M_{ij} \cap M_{hg}$ and constrain (11) guaranties that only one operation precedes immediately g th operation of sub-lot $l' \in L_h$ of job $h \in N$ on machine $k \in M_{ij} \cap M_{hg}$. Equations (12) establishes the precedence constraint between two consecutive operations of the same sub-lot. Constraints (13) enforces that each operation of each sub-lot should be assigned to exactly one machine among the possible ones. The respect of the due date of jobs is modeled by (14). The constraints (15) ensure that the capacities of material resources in number of portions are respected. The respect of the time windows of availability of material resources is modeled by (16) and (17). Finally, (19), (20), (21) and (22) define the domain of decision variables.

4.5 Computational results of mathematical model

The mathematical model presented previously was implemented in Java programming language using the Cplex library to solve it. This mathematical model has been tested on more than 100 instances of different types : adapted instances of literature (Behnkel and Geiger (2012), Azzouz et al (2017), Buddala and Mahapatra (2018), Shen et al (2018)), randomly generated instances (Table 5), andomly generated instances of type HCT (Table 4) and real instances of HCT (Table 3). The real instances of HCT were built after having timed the processing times of operations of dishes of some examples of days (example: instance with 82 dishes, 92 sub-lots , 370 operations and 29 machines, ...). From this example, several instances were built by increasing each time the number of jobs, sub-lots and operations to see from what number of jobs, sub-lots and operations the model is not able to find solutions in a reasonable resolution time. The following tables presents the computational results of the mathematical model (P2) on these different types of instances:

Job	Sub-lot	Operation	Machine	$\sum C_i$	Time
2	2	10	29	15.3 h	2 s
3	3	15	29	24.4 h	4 s
4	4	20	29	32.8 h	2 mn
5	5	24	29	39.9 h	4 mn
6	6	29	29	49.2 h	20 mn
7	7	34	29	57.6 h	45 mn
8	10	39	29	69.4 h	2h30
9	11	44	29	-	$>$ 3 h
82	92	370	29	-	$>$ 3 h

Table 3: Computational results of the mathematical model (P2) on real instances of HCT.

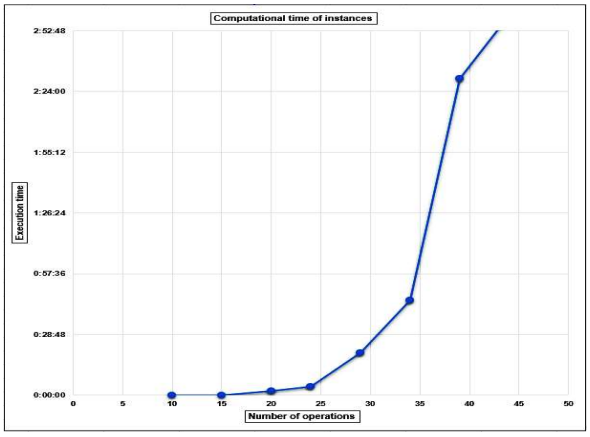

Figure 7: Computational time of instances according to the number of operations.

Job	Sub-lot	Oper	Mach	$\sum C_i$	Time
110	395	402	29	-	$>$ 3 h
100	100	342	29	-	$>$ 3 h
90	90	324	29	-	$>$ 3 h
80	277	286	29	-	$>$ 3 h
60	208	200	29	-	$>$ 3 h
50	50	173	29	-	$>$ 3 h
20	20	74	29	-	$>$ 3 h
15	15	59	29	-	$>$ 3 h
10	10	31	29	-	$>$ 3 h
5	5	19	29	-	$>$ 3 h

Table 4: Computational results on randomly generated instances of type HCT.

Job	Sub-lot	Operat	Mach	$\sum C_i$	Time
3	12	11	6	117	16 s
3	8	13	6	85	24 s
3	10	8	6	45	10 s
3	11	11	6	55	20 s
3	12	12	6	117	104 s
3	11	13	6	78	30 s
3	10	12	6	88	172 s
3	10	13	6	103	240 s
3	10	10	6	76	28 s
3	8	9	6	44	24 s

Table 5: Computational results on randomly generated instances.

From the computational results of the mathematical model presented previously, we observe that the model gives quickly a solution for the small instances with a certain number of jobs, sub-lots, and operations. The execution times of this mathematical model for these instances vary according to the number of jobs, sub-lots, and operations. It is important to note that the execution time is given only for instances where the optimal solution is obtained. In the opposite case, the bar - means that no optimal solution was found after 3 hours of execution. The computational results of the mathematical model proposed on different types of instances and specifically on real instances of HCT show the limits of an exact resolution for the problem of food production scheduling.

5 Genetic algorithms

Solving the classical job shop problem is known to be NP-hard (Garey et al 1976) and so is solving the flexible job shop problem with job splitting. In order to solve this problem efficiently and in a reasonable resolution time, we developed a hybrid method combining a genetic algorithm and three local research methods. The overall operation of this method with the different stages are given in the following flowchart (Figure 8) :

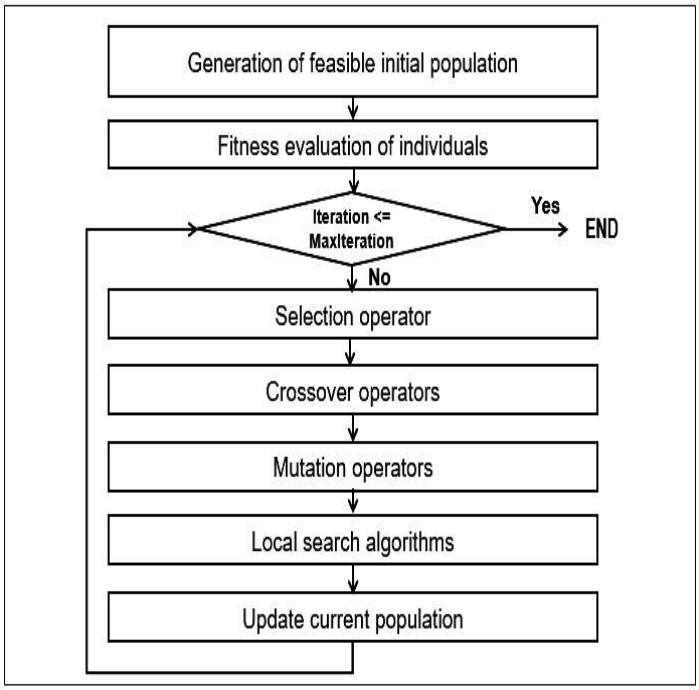

Figure 8: Flowchart of the genetic algorithm developed.

The various elements of this method are presented in the following subsections.

5.1 Solution representation

By solving a flexible job shop scheduling problem using a genetic algorithm, Kacem (2003) uses a solution representation coding both the assignment and the sequencing of operations on different machines. Similar representations can be used to solve the flexible job shop scheduling problem with job splitting, if each sub-lot is considered a job. To illustrate this representation of the solution, consider a small example of the problem with three jobs and four machines. The number of operations, of sub-lots for each job, and all the machines eligible for each operation are given in the figure (Figure 9). By considering each sub-lot as a job, and using the technique proposed by Kacem (2003), a representation of assignment of machines to operations, and sequencing is coded in a chromosome (Figure 9). In this chromosome, each gene is represented by a quadruple (i, l, j, k), designating the assignment of the operation j of the sub-lot l of job i to machine k. The sequence of genes in the chromosome represents the sequencing of operations on machines. For example, the assignment and sequencing of operations on machine 1 can be decoded as follows : $(i1, l3, j1) \rightarrow (i3, l2, j3) \rightarrow (i3, l3, j3)$. This information is obtained from genes 8, 20 and 21 in the chromosome, where $k = 1$. In this chromosome, for a given i and l, the gene (i, l, j, k) is always located on the right of all the other genes (i, l, j', k') with $j' < j$. This ensures that the precedence requirement of the operations of a particular sub-lot are not violated.

Job	Operations	Sublots	Set of eligible machines for operation		
			$o1$	$o2$	$o3$
$j1$	3	3	$\{m1, m2\}$	$\{m3\}$	$\{m2, m4\}$
$j2$	2	2	$\{m3, m4\}$	$\{m2\}$	
$j3$	3	3	$\{m3\}$	$\{m2, m4\}$	$\{m1, m3\}$

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	
3, 2, 1, 3	1, 2, 1, 2	2, 1, 1, 3	3, 1, 1, 3	2, 2, 1, 4	3, 2, 2, 4	3, 3, 1, 3	1, 3, 1, 1	3, 1, 2, 2	1, 1, 1, 2	3, 3, 2, 2	1, 2, 2, 3	1, 1, 2, 3	3, 1, 3, 3	1, 3, 2, 3	2, 1, 2, 2	1, 1, 3, 4	2, 2, 2, 2	1, 3, 3, 4	3, 2, 3, 1	3, 3, 3, 1	1, 2, 3, 2	i, l, j, k

i = job index, l = sublot index, j = operation index, k = machine index

Figure 9: Representation of the assignment of operations to machines and their sequencing.

Operation assigned to production run								
Machine	r1	r2	r3	r4	r5	r6	r7	r8
m1	(i1,l3,j1)	(i3,l2,j3)	(i3,l3,j3)					
m2	(i1,l2,j1)	(i3,l1,j2)	(i1,l1,j1)	(i3,l3,j2)	(i2,l1,j2)	(i2,l2,j2)	(i1,l2,j3)	
m3	(i3,l2,j1)	(i2,l1,j1)	(i3,l1,j1)	(i3,l3,j1)	(i1,l2,j2)	(i1,l1,j2)	(i3,l1,j3)	(i1,l3,j2)
m4	(i2,l2,j1)	(i3,l2,j2)	(i1,l1,j3)	(i1,l3,j3)				

Figure 10: Operation assignment and sequencing decoded from Figure 8.

5.2 Initial population

The initial population plays an important role in the performance of genetic algorithms. An initial population of better solutions, with more diversity can avoid falling into a premature convergence or a local minimum. The initial population cannot be formed until the operations of each sub-lot of job have been assigned and sequenced on machines. Assignment and sequencing rules must be developed for a flexible job shop problem with splitting of jobs into sub-lots, while taking into account the specific characteristics of the material resources. For the assignment problem, three heuristics have been proposed :

- **Random assignment (RA)** : the operations are randomly assigned to the machines.
- **SPT assignment (SPTA)** : the operations are assigned to machines according to the SPT (Shortest Processing Time) rule. For each operation, the machine with a smaller processing time is selected to perform this operation.
- **LPT assignment (LPTA)** : the operations are assigned to machines according to the LPT (Longest Processing Time) rule. For each operation, the machine with a longer processing time is selected to perform this operation.
- **Minimum machine workload assignment (MMWA)** : the operations are iteratively assigned to machines based on their processing times and machine workloads. The workload of a machine depends on its type. For the set of machines $M1$ (material resources that can perform only one operation at the same time), the workload of machine is equal to the sum of processing times of

operations assigned to the machine. For the set of machines $M2$ and $M3$ (set of machines that can perform several operations at the same time, exp : ovens, cooling cells, etc.), the workload of machine is equal to the total machine occupation time. The procedure consists in finding, for each operation, the machine with the minimum workload. The aim of this heuristic is to balance the workload between machines.

The procedure for assigning operations to machines using the MMWA heuristic is explained in the following table through an example with 4 machines and 3 jobs. Each operation can be processed by any of the 4 machines. The data and the assignment of operations to machines using MMWA heuristic are shown in the following table (Table 6) :

Job	Operation	Machine			
		M1	M2	M3	M4
J	O	(Setup time, Processing time)			
1	1	(10, 20)	(10, 50)	(15, 10)	(10, 30)
	2	(10, 40)	(15, 60)	(15, 80)	(10, 40)
	3	(15, 90)	(10, 70)	(15, 20)	(10, 20)
2	1	(25, 70)	(20, 60)	(20, 40)	(20, 50)
	2	(10, 40)	(10, 80)	(30, 50)	(20, 60)
	3	(15, 90)	(20, 50)	(15, 40)	(15, 70)
3	1	(10, 80)	(15, 60)	(25, 30)	(15, 50)
	2	(15, 30)	(15, 50)	(20, 80)	(15, 30)

J	O	Initial Data				1st Assignment				2nd Assignment				···	Final Assignment			
		M4	M1	M3	M2	M4	M1	M3	M2	M4	M1	M3	M2		M4	M1	M3	M2
3	1	50	80	30	60	50	80	[30]	60	50	80	30	60		50	80	[30]	60
	2	30	30	80	50	30	30	**110**	50	[30]	30	110	50		[30]	30	80	50
2	1	50	70	40	60	50	70	**70**	60	**80**	70	70	60		50	70	40	[60]
	2	60	40	50	80	60	40	**80**	80	**90**	40	80	80	···	60	[40]	50	80
	3	70	90	40	50	70	90	**70**	50	**100**	90	70	50		70	90	[40]	50
1	1	30	20	10	50	30	20	**40**	50	**60**	20	40	50		[30]	20	10	50
	2	40	40	80	60	40	40	**110**	60	**70**	40	110	60		40	[40]	80	60
	3	20	90	20	70	20	90	**50**	70	**50**	90	50	70		[20]	90	20	70

Table 6: Assignment of operations to machines according to MMWA heuristic.

The order of jobs and operations in the first two columns of the above table are randomly permutated when forming the initial population. Here the random order is job 3, job 2, and job 1.

Now that assignment rules are in place to determine which machine should be used for a particular flexible operation, the sequence of the operations in the chromosomes still needs to be determined before forming the initial population. Four heuristics are proposed to determine the sequencing of operations on the machines :

- **Random sequence (RS)** : randomly orders the operations on each machine.
- **SPT sequence (SPTS)** : the operations with the shortest processing time will be firstly processed.
- **Most Number of Operations Remaining (MNOR)** : the sequence of operations depends on the job that has the most remaining operations to schedule. It consists of processing first, the operations of sub-lots of job which has the most remaining operations.
- **Most Work Remaining (MWR)** : the operations which have the most remaining processing time will be processed in priority.

Using the assignment of operations to machines given in the table above (Table 6), the following table (Table 7) shows the procedure for sequencing operations on machines using the MWR rule :

Work Remaining (WR) Calculation			Operation Sequence			
J	**O**	**WR**	**Machine**	**Run1**	**Run2**	**Run3**
1	**1**	$(10+30)+(10+40)+(10+20)=120$				
	2	$(10+40)+(10+20)=80$				
	3	$(10+20)=30$	**M1 :**	J2,O2	J1,O2	
2	**1**	$(20+60)+(10+40)+(15+40)=185$	**M2 :**	J2,O1		
	2	$(10+40)+(15+40)=105$				
	3	$(15+40)=55$	**M3 :**	J3,O1	J2,O3	
3	**1**	$(25+30)+(15+30)=100$	**M4 :**	J1,O1	J3,O2	J1,O3
	2	$(15+30)=45$				

Table 7: Sequencing of operations on machines according to the MWR heuristic.

In all the heuristics presented previously, the solutions obtained must respect the constraints of precedence between the operations of sub-lots of jobs, the due dates of jobs, the time windows of availability of material resources and the production capacities of machines.

5.3 Fitness evaluation

When the chromosomes are coded, they must be decoded in order to calculate the fitness function. The flow time of the schedule corresponding to a given chromosome is used as the fitness function of this chromosome. The chromosomes with better fitness function have a smaller flow time. The chromosome evaluation function takes into account : (1) the precedences between the operations of sub-lots of jobs, (2) the production capacity in number of portions of machines, (3) the due dates of jobs, (4) machines availability time windows, (5) the setup times of machines which depend on the sequence of operations. The process of decoding a chromosome to evaluate its fitness function is as follows :

Algorithm 1 Evaluation steps of the fitness function of a chromosome.

- **Step 01 :** Set $P = 1$
- **Step 02 :** Set i, l, j, k, the index values of the gene located at the P position of the chromosome.
- **Step 03 :** Calculate the completion time. C_{iljk}
  - If operation j of sub-lot l of job i is the first operation assigned to the machine k and $j = 1$

$$S_{iljk} = A_k + T_k;\ C_{iljk} = S_{iljk} + P_{ijk};\ C_i = Max\{C_i, C_{iljk}\}$$

  - If operation j of sub-lot l of job i is the first operation assigned to the machine k, $j > 1$ and the operation $j-1$ is assigned to the machine k'

$$S_{iljk} = Max\{A_k + T_k, C_{ilj-1k'}\};\ C_{iljk} = S_{iljk} + P_{ijk};\ C_i = Max\{C_i, C_{iljk}\}$$

  - If operation j' of sub-lot l' of job i' is the operation to be processed immediately before the operation j of sub-lot l of job i on the machine k and $j = 1$

$$S_{iljk} = s_{i'j'ijk} + C_{i'l'j'k};\ C_{iljk} = S_{iljk} + P_{ijk};\ C_i = Max\{C_i, C_{iljk}\}$$

  - If operation j of sub-lot l of job i and the operation j of sub-lot l' of job i are assigned to the machine k and $j = 1$
    - If ($k \in M_2 \cup M_3$ and $2 * q_i <= R_k$)

$$S_{iljk} = S_{il'jk};\ C_{iljk} = S_{iljk} + P_{ijk};\ C_i = Max\{C_i, C_{iljk}\}$$

    - If ($k \in M_2 \cup M_3$ et $2 * q_i > R_k$) ou ($k \in M_1$)

$$S_{iljk} = F_{il'jk};\ C_{iljk} = S_{iljk} + P_{ijk};\ C_i = Max\{C_i, C_{iljk}\}$$

  - If operation j' of sub-lot l' of job i' is the operation to be processed immediately before the operation j of sub-lot l of job i on the machine k, $j > 1$ and the operation $j-1$ is assigned to the machine k'

$$S_{iljk} = Max\{s_{i'j'ijk} + C_{i'l'j'k}, C_{ilj-1k'}\};\ C_{iljk} = S_{iljk} + P_{ijk};\ C_i = Max\{C_i, C_{iljk}\}$$

  - If operation j of sub-lot l of job i and the operation j of sub-lot l' of job i are assigned to the machine k, $j > 1$ and the operation $j-1$ is assigned to the machine k'
    - If ($k \in M_2 \cup M_3$ and $2 * q_i <= R_k$)

$$S_{iljk} = S_{il'jk};\ C_{iljk} = S_{iljk} + P_{ijk};\ C_i = Max\{C_i, C_{iljk}\}$$

    - If ($k \in M_2 \cup M_3$ and $2 * q_i > R_k$) or ($k \in M_1$)

$$S_{iljk} = Max\{C_{il'jk}, C_{ilj-1k'}\};\ C_{iljk} = S_{iljk} + P_{ijk};\ C_i = Max\{C_i, C_{iljk}\}$$

- **Step 04 :** If P is less than the total number of operations of sub-lots of jobs, increment its value by 1 and go to Step 2; otherwise, go to Step 05
- **Step 05 :** Calculate the fitness function of the solution $\sum_i C_i$.

This process of fitness function evaluation, in particular Step 03 is based on the property of the chromosomes that, for a given i and l, the gene (i, l, j, k) is always located on the right of all the other genes

(i, l, j', k') with $j' < j$. Based on this property, when the completion time of operation (i, l, j, k) on machine k is to be calculated, the completion time of operation $(i, l, j-1, k')$ is already calculated and available, regardless to which machine this preceding operation is assigned. Moreover, the completion time of the operation (i', l', j', k) to be processed on machine k immediately before operation (i, l, j, k) is also calculated and available.

5.4 Genetic operators

Genetic operators evolve the population to promising regions of the research space. Their design is so crucial that the convergence behavior of the algorithm depends largely on them. These operators are generally categorised as selection, crossover, and mutation operators.

5.4.1 Selection operator

The process of selecting two parents from the population for reproduction is called selection. The aim of the selection operator is to highlight individuals with best fitness, in hopes that their resulting offsprings are fitter individuals. In the proposed genetic algorithm, we used k-way tournament selection operator which was introduced in Goldberg et al (1989). Depending on the value of k, there can be a variation of chromosomes selected to form the new population. A large value of k will lead to lower variation of chromosomes in the new population. A small value of k will lead to a higher variation of chromosomes in the new population. The selection operator is involved in holding competition among k randomly selected individuals, and choosing the one with the best fitness (smallest flow time). This individual is added to the mating population to form the next generation. Then, the k individuals in the tournament are placed back in the current population, and the process is repeated. This process continues until the number of individuals added to the mating population is equal to the population size.

5.4.2 Crossover operator

After the selection of chromosomes for reproduction, crossover operator is applied to combine the features of the parent chromosomes in order to produce a new childs, and to enriching the population with best chromosomes. The individuals in the mating population are randomly paired to form parents for the next generation. Then for each pair, the algorithm arbitrarily selects one of the available crossover operators, and applies it with a certain probability to create two child individuals, by exchanging information contained in the parent chromosomes. In the following, we provide the description of the crossover operators developed in this study for flexible job shop scheduling with job splitting problem.

The crossover operators for flexible job shop scheduling with job splitting problem can be categorized as assignment or sequence crossover operators. The assignment crossover operators consist generating offsprings, by exchanging the assignment properties of the parent chromosomes. The role of sequence crossover operators are to produce two new offsprings, by exchanging the sequencing properties of parent chromosomes. The assignment and sequence crossover operators used in the proposed genetic algorithm are: OMAC (Operation to Machine Assignment Crossover), JLOSC (Job Level Operations Sequence Crossover), SLOSC (Sublot Level Operations Sequence Crossover).

From a given pair of parent chromosomes, OMAC creates two child chromosomes, where each child chromosome retains the order of the operations supplied by the other parent chromosome. The creation of child 1 by this operator, retaining the order of the operations as obtained from parent 1, as illustrated in Figure 13 and Figure 14. In the genetic algorithm proposed, we developed three crossover operators OMAC 1, OMAC 2 and OMAC 3. The objective of crossover operator OMAC 2 is to balance

the workload between the machines. Whereas, in the crossover operator OMAC 3, the goal is to reduce the ending dates of jobs, with a larger completion times. The difference between these three crossover operators is in the choice of operations that we seek to change their assignments. In the first crossover operator OMAC 1, a set of operations of sub-lot of jobs is chosen randomly. In OMAC 2, we choose a set of operations of sub-lots assigned to the loaded machines, while in OMAC 3, a set of operations of sub-lots of jobs with larger completion times is chosen.

The steps of the different operators OMAC 1, OMAC 2 and OMAC 3 are as follows :

Algorithm 2 Crossover operator OMAC 1

- **Step 01 :** Randomly select a set of operations from the chromosome parent 1.
- **Step 02 :** Keep the remaining unmodified sequence of the set of operations from the chromosome parent 1, and copy onto the child chromosome.
- **Step 03 :** Assignment properties from the chromosome parent 2 are copied onto the child chromosome.
- **Step 04 :** Create child 2 by starting to select the operations from the chromosome parent 2, and continue the above process.

Algorithm 3 Crossover operator OMAC 2

- **Step 01 :** Choose a set of operations assigned to the loaded machines in the chromosome parent 1.
- **Step 02 :** All the genetic informations of the parent 1 except the assignment informations of the selected operations is copied to child 1.
- **Step 03 :** The assignment properties of the selected operations are copied from parent 2 to complete child 1.
- **Step 04 :** Create child 2 by starting to select the operations from parent 2, and continue the above process.

The procedure for choosing the set of operations in Step 01 is as follows :

Algorithm 4 Procedure for choosing operations in OMAC 2

- **Step 01 :** Calculate machine workloads.
- **Step 02 :** Sort machines in descending order from the most loaded to the least loaded machine.
- **Step 03 :** Choose the α most loaded machines.

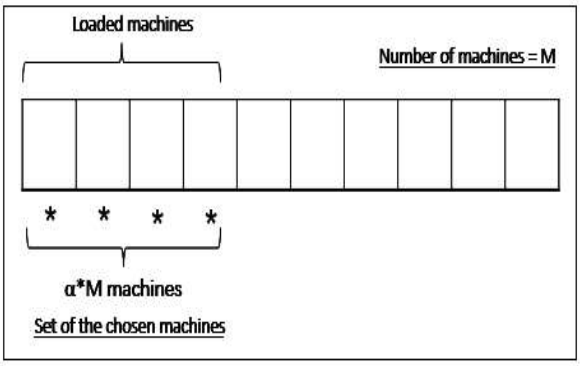

Figure 11: Machines selection procedure in OMAC 2.

Algorithm 5 Crossover operator OMAC 3

- **Step 01 :** Choose a set of operations of sub-lots of jobs with larger completion times in the chromosome parent 1.
- **Step 02 :** All the genetic informations of the chromosome parent 1 except the assignment informations of the selected operations is copied to child 1.
- **Step 03 :** The assignment properties of the selected operations are copied from parent 2 to complete child 1.
- **Step 04 :** Create child 2 by starting to select the operations from chromosome parent 2, and continue the above process.

The procedure for choosing the set of operations in Step 01 is as follows :

Algorithm 6 Procedure for choosing operations in OMAC 3

- **Step 01 :** Calculate the completion times of jobs.
- **Step 02 :** Sort the jobs in descending order of completion time of jobs.
- **Step 03 :** Choose the α jobs with a larger completion time.

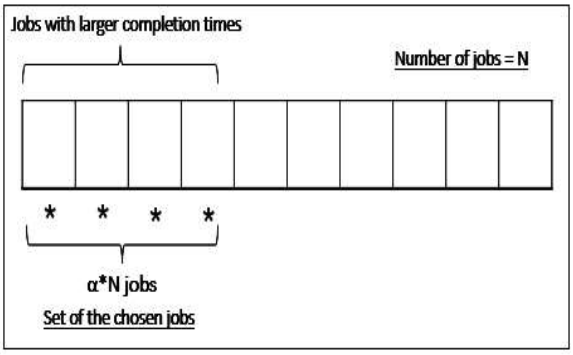

Figure 12: Jobs selection procedure in OMAC 3.

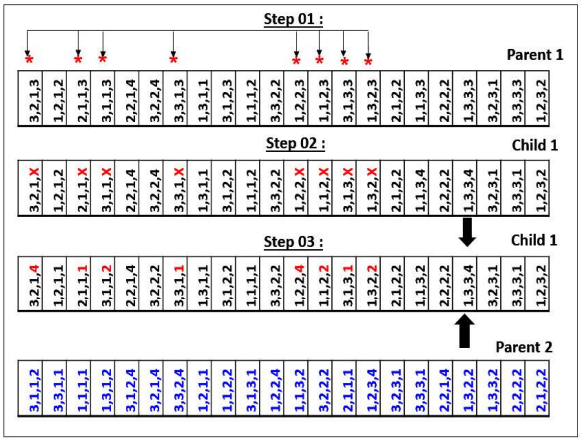

Figure 13: OMAC 2 assignment operator.

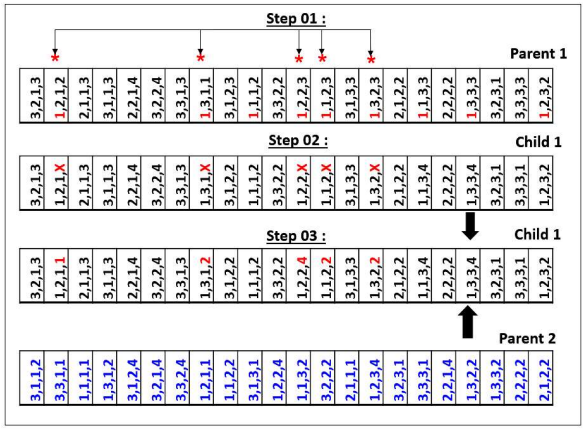

Figure 14: OMAC 3 assignment operator.

The crossover sequencing operators JLOSC and SLOSC produce two new offsprings, by exchanging the sequencing properties of parent chromosomes, while leaving the assignment of the machines in a gene unmodified. They are applied with certain probabilities. The creation of the child chromosome by JLOSC and SLOSC, preserving the operation assignment informations of parent chromosomes, is illustrated in Figure 16 and Figure 17. In the genetic algorithm proposed, we developed two crossover operators JLOSC 1, JLOSC 2 and two crossover operators SLOSC 1, SLOSC 2. The objective of the crossover operators JLOSC 2 and SLOSC 2 is to keep the sequencing of operations of jobs and the sequencing of operations of sub-lots of jobs with smaller completion times.

The steps of creation of a child chromosome by JLOSC 1, SLOSC 1, JLOSC 2 and SLOSC 2 are as follows:

Algorithm 7 Crossover operator JLOSC 1

- **Step 01 :** Choose arbitrary a set of operations of sub-lots of jobs from the chromosome parent 1.
- **Step 02 :** For the set of chosen operations, keep the remaining unmodified operations of all the sublots of jobs and copy onto the child chromosome.
- **Step 03 :** For the remaining jobs, retain and copy the same sequence from the second parent onto the child chromosome.
- **Step 04 :** Create child 2 by starting to select the operations in the parent 2 and continue the above process.

The crossover operator SLOSC 1 is similar to JLOSC 1, but instead of keeping the unmodified operations of all the sublots, only the information from the sublot of the selected operation is kept and copied to the child chromosome.

Algorithm 8 Crossover operator JLOSC 2

- **Step 01 :** Choose a set of jobs with a smaller completion time in the chromosome parent 1.
- **Step 02 :** The operations of sub-lots of jobs chosen in Step 01 are copied to child 1.
- **Step 03 :** The operations of sub-lots assigned to the same machines as the operations of sub-lots of jobs chosen in Step 01 and the previous operations of these operations of the same job are copied to child 1.
- **Step 04 :** Child 1 is completed with the remaining operations, in the same order as they appear in parent chromosome 2 while their assignment properties are kept unchanged as they were in parent chromosome 1.
- **Step 05 :** Create child 2 by starting to select the operations in the parent 2 and continue the above process.

The procedure for choosing the set of operations in Step 01 is as follows :

Algorithm 9 Procedure for choosing operations in JLOSC 2

- **Step 01 :** Calculate the completion times of jobs.
- **Step 02 :** Sort the jobs in ascending order of completion time of jobs.
- **Step 02 :** Choose the α jobs with a smaller completion time.

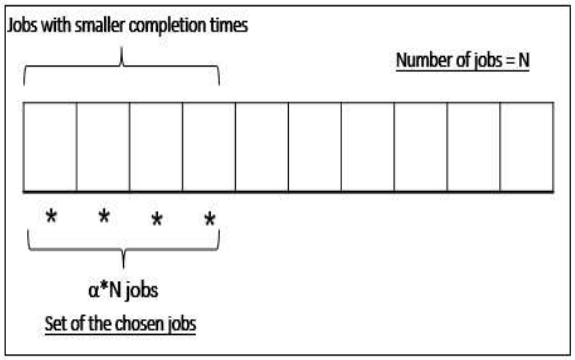

Figure 15: Jobs selection procedure in JLOSC 2.

The SLOSC 2 crossover operator is similar to the JLOSC 2 crossover operator except that in Step 01, we choose a set of sub-lots of jobs with smaller completion time.

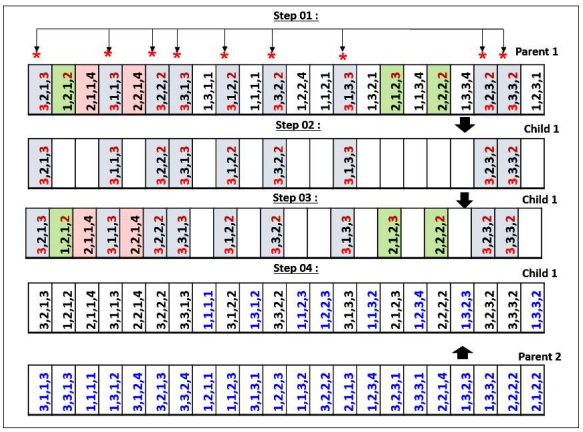

Figure 16: JLOSC 2 sequence operator.

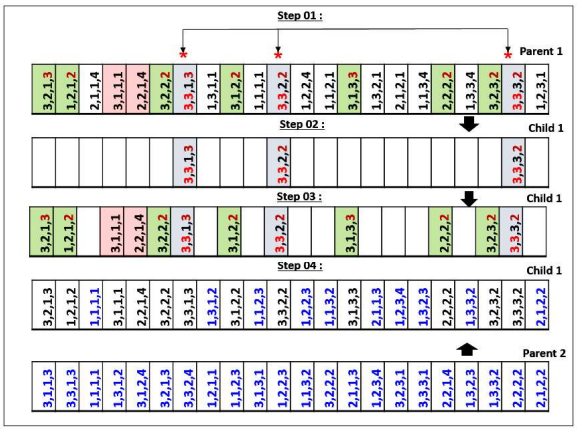

Figure 17: SLOSC 2 sequence operator.

It is important to note that, after the application of any crossover operator, the child chromosome have always for a given i and l, the gene (i, l, j, k) is located after any gene (i, l, j, k), where $j' < j$. This ensures that precedence constraints between operations of a particular sub-lot are not violated in the new created child chromosome.

5.4.3 Mutation operator

Crossover operators do not change the genetics of the existing chromosomes. They only alter the assignment and, sequencing of the chromosome. Mutation operators are able to introduce new genetic material into the population, by altering the information contained in one of the genes. The role of mutation operators is to prevent the algorithm from being trapped in local optimum, and to maintain genetic diversity in the population. Unlike the role of crossover operator in exploiting the current solution for better ones, mutation operator plays the role of exploring whole search space (Sivanandam and Deepa, 2007). The mutation operators are usually applied on chromosomes with certain probabilities. Assignment and sequence operators also exist amongst the mutation operators. Assignment mutation operators change the assignment of operations to machines without changing the sequencing of these operations, and sequence operators only change the sequencing property of the chromosome undergoing the mutation, while the assignment property is reserved. The mutation operators used in the proposed genetic algorithm are : ROAM (Random Operation Assignment Mutation) and OSSM (Operations Sequence Shift Mutation). The ROAM mutation operator is applied with a certain probability on a set of operations of a given individual chromosome, and changes the assignment property of these operations to another alternative machines. The OSSM mutation operator is in class of sequence mutation operator. Whenever this operator is applied on an individual, an operation is selected and then moved to another position on the chromosome in such a way that no precedence constraint is violated. Figure 18 and 19 illustrate the creation of new offsprings using these operators.

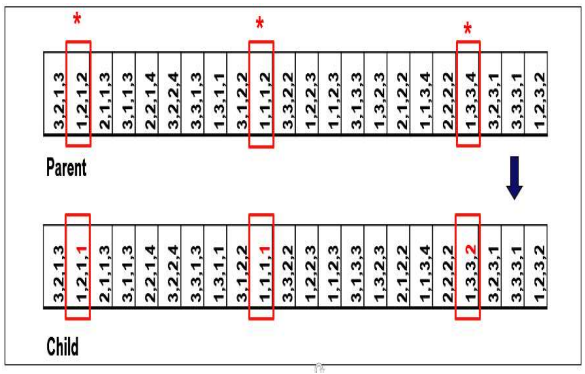

Figure 18: ROAM assignment operator.

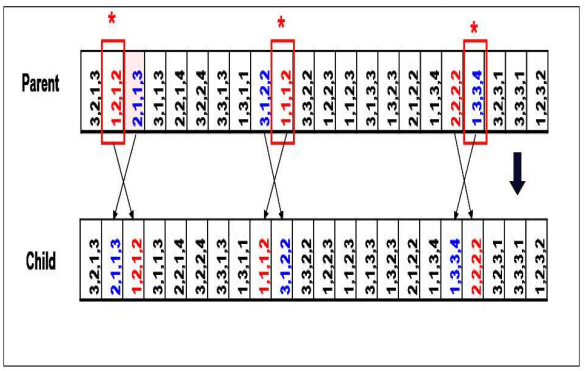

Figure 19: OSSM sequence operator.

5.5 Local search methods

5.5.1 Local search by gene movement

This local search method improves the quality of any solution built after the application of the crossover and mutation operators. The procedure for this method is illustrated through an example in Figure 20. The steps of this local search method are as follows :

Algorithm 10 Local search procedure by gene movement

- **Step 01 :** Choose a set of operations of jobs with larger completion time in the current chromosome.
- **Step 02 :** Each operation chosen in Step 01 is positioned just before the previous operation assigned to the same machine (left movement) or just after (right movement) the following operation assigned to the same machine, while respecting the constraints of precedence between the operations of sub-lots of jobs. This process changes the sequencing of operations on the machines.
- **Step 03 :** This process is repeated a certain number of iterations.

5.5.2 Local search by grouping sub-lots

This local search method consists in grouping the operations of sub-lots of jobs on machines, if the capacities of machines allow to process several operations at the same time. This method can only be applied for machines such as ovens, cooling cells, etc. The procedure for this local search method is illustrated through an example in Figure 21. The grouping of operations of sub-lots of jobs on machines must take into account the precedence between operations of these sub-lots.

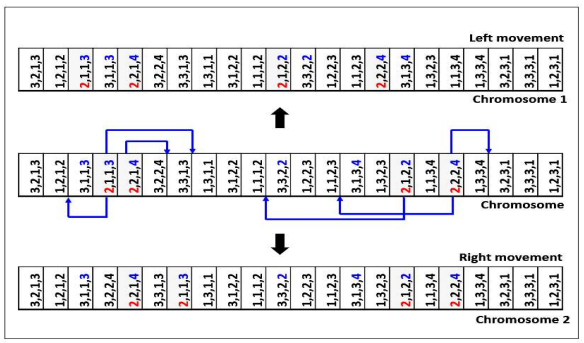

Figure 20: Gene movement procedure.

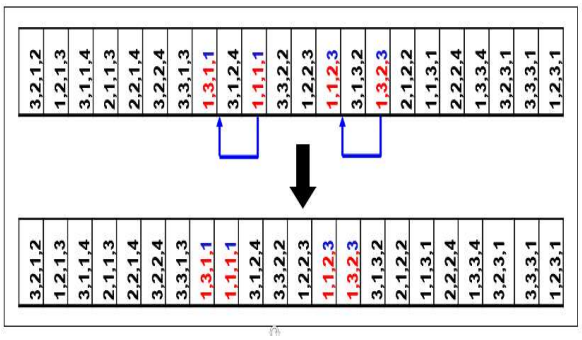

Figure 21: Procedure of grouping of sub-lots.

5.5.3 Local search by intelligent assignment of operations

In this local search approach, a set of operations assigned to the most loaded machines is reassigned to the least loaded compatible ones. The workload of a machine is the sum of processing times of operations assigned to the machine, for all the material resources that can perform only one operation at the same time, and it is equal to the total duration of occupation of the material resource for the set of machines that can perform several operations at the same time.

5.6 Pseudo codes of developed genetic algorithms

In this subsection, we present the pseudo codes of the two developed genetic algorithms. These algorithms differ in the generation methods of the initial population, and in the crossover operators. In both algorithms, several affectation and sequence heuristics were tested for the generation of the initial population. In the first genetic algorithm, crossover operators are based on random choices of operations, while in the second algorithm different operators specific and more adapted to the problem studied have been developed. The pseudo codes of the two algorithms are given in algorithms 11 and 12.

Algorithm 11 Genetic algorithm 1

1. **Initialization :** the initial solutions are chosen randomly or using the heuristics described above
2. **Evaluation :** evaluation of initial solutions using the procedure of calculating the fitness function nbIterations $\leftarrow 0$; nbIndividuals $\leftarrow 0$
3. Application of the selection operator to choose the parent chromosomes to cross and mutate
4. Randomly choose one of the crossover operators OMAC 1, JLOSC 1 or SLOSC 1
5. Application of the crossover operator chosen in 4
6. Randomly choose one of the mutation operators ROAM, IOAM or OSSM
7. Application of the mutation operator chosen in 6
8. Application of local research by grouping sub-lots on child chromosomes obtained after crossover and mutation
9. nbIndividuals $\leftarrow$ nbIndividuals + 1
10. **If** the population size is reached **then** go to 11 **else** go to 3
11. Sorting solutions of the current population in ascending order of fitness functions
12. Construction of the new population with solutions of the previous population before and after sorting 13. Application of local research by gene movement on the α solutions of the new population
14. nbIterations $\leftarrow$ nbIterations + 1
15. **If** the maximum number of iterations is not reached **then** go to 3 **else** go to 16
16. **End of algorithm**

Algorithm 12 Genetic algorithm 2

1. **Initialization :** the initial solutions are chosen using the assignment and sequence heuristics described above
2. **Evaluation :** evaluation of initial solutions using the procedure of calculating the fitness function nbIterations $\leftarrow 0$; nbIndividuals $\leftarrow 0$
3. Application of the selection operator to choose the parent chromosomes to cross and mutate
4. Randomly choose one of the crossover operators OMAC 2, OMAC 3, JLOSC 2 or SLOSC 2
5. Application of the crossover operator chosen in 4
6. Randomly choose one of the mutation operators ROAM, IOAM or OSSM
7. Application of the mutation operator chosen in 6
8. Application of local research by grouping sub-lots on child chromosomes obtained after crossover and mutation
9. nbIndividuals $\leftarrow$ nbIndividuals + 1
10. **If** the population size is reached **then** go to 11 **else** go to 3
11. Sorting solutions of the current population in ascending order of fitness functions
12. Construction of the new population with solutions of the previous population before and after sorting 13. Application of local research by gene movement on the α solutions of the new population
14. nbIterations $\leftarrow$ nbIterations + 1
15. **If** the maximum number of iterations is not reached **then** go to 3 **else** go to 16
16. **End of algorithm**

6 Iterated Local Search

This section describes our proposed iterated local search (ILS) algorithms for solving the problem of scheduling production process. ILS is a simple, robust and highly effective local search procedure, which explores local optima in a given neighborhood (Lourenço et al, 2003). ILS starts from an initial solution and obtains local optimum in its neighborhood, by a local search procedure. To improve upon the current local optimum, ILS performs the local search procedure to a perturbation solution of the current local optimum and finds a new local optimum. An acceptance criterion is employed to determine which local optimum will become the current local optimum in the next iteration. The above process is repeated until a termination criterion is met. The ILS consists of three main components: initial solution construction, local search procedure and perturbation mechanism which are detailed in the following. The overall operation of the iterated local search algorithm with the different stages are given in the following flowchart (Figure 22) :

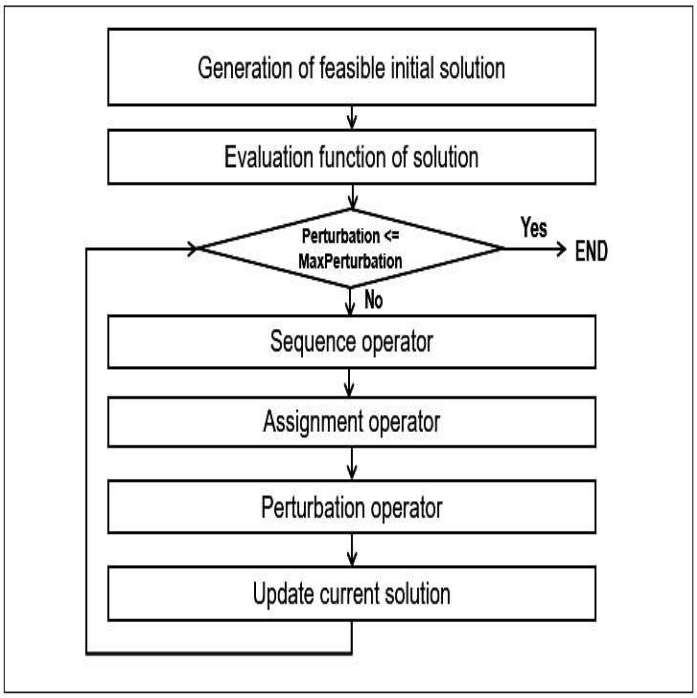

Figure 22: Flowchart of the iterated local search algorithm developed.

To generate the initial solutions of the iterative local search algorithms, the affectation and sequence heuristics described above were used.

In the proposed iterative local search, the local search procedure is based on two operators : affectation operator (AO) and sequence operator (SO). The affectation operator changes the assignment of a set of operations to the machines without changing their sequencing on the machines, while the sequence operator changes the sequencing of operations without changing their affectation to machines. In the ILS algorithm proposed, we developed three affectation operators AO 1, AO 2, AO 3, and three sequence operators SO 1, SO 2 ans SO 3. The goal of OA 2, is to balance the workload between the machines, while AO 3, SO 2 and SO 3 allow to reduce the ending dates of jobs with a larger completion times. The different steps of these operators are described in the following.

With reducing of solution space, local search is easy to trap in a local optimum. To solve this problem, we carry out a number of insertion moves to the current local optimum and obtain a perturbation solution. The ILS algorithm restarts local search from this perturbation solution. If the number of perturbations is too large, the perturbation solution is far from the current local optimum. On the other hand, if the perturbation number is too small, the perturbation solution is very close to the current local optimum. The ILS algorithm will fall into a local optimum and the diversification of the search space will be very limited. Therefore, appropriate the number of perturbations should be determined for the ILS algorithm.

At each iteration of the main loop, an initial feasible solution is constructed for the ILS loop. At each ILS iteration, the local search procedure takes as input the current solution, and returns an improved solution of this one, which is accepted as the new best current solution if it is feasible and has a flow time strictly smaller than the initial solution. Then a new starting solution for the local search procedure is generated by perturbing the new best current solution. This procedure repeats until a predefined number of iterations have been executed.

The steps of the different sequence and affectation operators are as follows :

Algorithm 13 Sequence operator (SO)

Input : Solution to improving
Output : Improved solution of the current one after resequencing of operations on machines
nbIterations ← 0
while *the maximum number of iterations is not reached* **do**
1. Randomly choose one of the operators SO 1, SO 2 or SO 3
2. Choose an operation in the current solution according to the operator chosen in 1
3. Shift the selected operation in 2 to the left until the permutation of this operation with the previous operation assigned to the same machine is effective
4. If the left shift is blocked (precedence constraints are violated) **then** from the solution obtained in 3, shift the selected operation in 2 to the right until the permutation of this operation with the next operation assigned to the same machine is effective
5. nbIterations← nbIterations + 1
end

Algorithm 14 Affectation operator (AO)

Input : Solution to improving
Output : Improved solution of the current solution after reassigning and resequencing of operations
nbIterations ← 0
while *the maximum number of iterations is not reached* **do**
1. Randomly choose one of the assignment operators AO 1, AO 2 or AO 3
2. Choose an operation in the current solution according to the operator chosen in 1
3. Affectation of the operation chosen in 2 to another machine among a set of alternative machines
4. If the solution obtained after the reassignment of the operation chosen in 2 is not better than the current solution **then** apply a sequence operator (SO) on the operation chosen in 2 with the affectation to machine obtained in 3
5. nbIterations← nbIterations + 1
end

Algorithm 15 Perturbation operator (PO)

nbPerturbations ← 0
while *the maximum number of perturbations is not reached* **do**
1. Randomly choose one of the sequence operators S1, SO 2, SO 3 or one of the affectation operators AO 1, AO 2, AO 3
2. Application of the sequence or affectation operator chosen in 1
3. nbPerturbations← nbPerturbations + 1
end

The difference between affectation operators, and sequence operators is in the choice of operations that we seek to change their assignments and sequencing. In AO 1 and SO 1 a set of operations of sub-lot of jobs is chosen randomly. The operator AO 2 choose a set of operations of sub-lots assigned to the loaded machines. In AO 3, SO 2 and , we select a set of operations of jobs with larger completion times, while in SO 3 a set of operations of sub-lots of jobs with larger completion times is chosen. The different steps of the iterative local search method are illustrated by an example in the figure 23.

6.1 Pseudo codes of developed ILS algorithms

In this section, we present two iterated local search algorithms. The difference between the two algorithms lies in the generation of the initial solution which is generated by using different heuristics in the two iterated local search algorithms. In the first algorithm, the operations to be selected to change their sequencing and assignments are chosen randomly, while in the second iterated local search algorithm, the choice of these operations is based on developed and specific rules more suited to the problem studied. The pseudo codes of the two iterated local search algorithms are given by algorithms 16 and 17 :

Algorithm 16 Iterative local search algorithm 1

1. **Initialization :** the initial solution is generated by using assignment and sequencing heuristics previously described
   current solution ← initial solution
2. **Evaluation :** evaluation of the initial solution
   nbPerturbation ← 0
3. Application of the SO 1 sequence operator
4. **If** the solution is improved in 3 **then**
   current solution ← solution obtained in 3
   Else go to 5
5. Application of the AO 1 affectation operator
6. **If** the solution is improved in 5 **then**
   current solution ← solution obtained in 5
   Else go to 7
7. Application of the perturbation operator on the current solution
8. nbPerturbation ← nbPerturbation + 1
9. **If** the maximum number of perturbations is not reached **then** go to 3
   Else go to 10
10. **End of algorithm**

Algorithm 17 Iterative local search algorithm 2

1. **Initialization :** the initial solution is generated by using assignment and sequencing heuristics previously described
   current solution ← initial solution
2. **Evaluation :** evaluation of the initial solution
   nbPerturbation ← 0
3. Application of the SO 2 or SO 3 sequence operator
4. **If** the solution is improved in 3 **then**
   current solution ← solution obtained in 3
   Else go to 5
5. Application of the AO 2 or AO 3 affectation operator
6. **If** the solution is improved in 5 **then**
   current solution ← solution obtained in 5
   Else go to 7
7. Application of the perturbation operator on the current solution
8. nbPerturbation ← nbPerturbation + 1
9. **If** the maximum number of perturbations is not reached **then** go to 3
   Else go to 10
10. **End of algorithm**

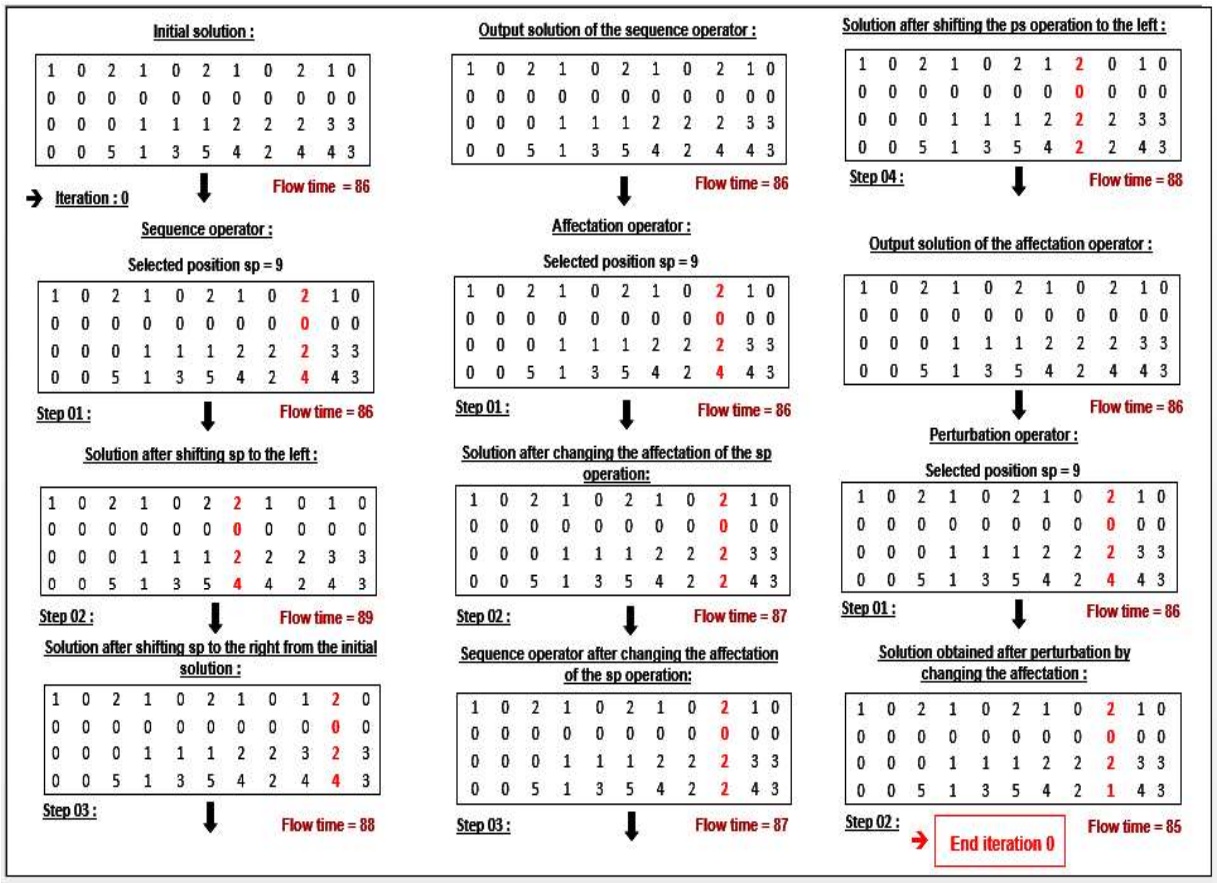

Figure 23: Steps of the iterative local search method on an iteration.

7 Parameter optimization using Taguchi design of experiments

The choice of optimal parameters apparently has a significant effect on the efficiency of metaheuristic algorithms. When the number of factors affecting the performance of the genetic algorithm is greater, the full factorial experiment and exhaustive approach to investigate the effect of different parameters becomes increasingly complicated and impractical.

In this work, the Taguchi method is used to optimize the parameters of metaheuristic algorithm developed. The parameters of a metaheuristic that are needed to be optimized act like controllable factors in the design of experiments (DOE). The aim is to find an optimal combination of the parameters such that the flow time is minimized. The Taguchi method is a special case of the fractional factorial design in which some special orthogonal arrays are used (Roy, 1990). Orthogonal arrays under Taguchi method would help to study a large number of decision variables with a limited number of experiments. Decision variables are divided into controllable and noise factors. Noise factors can not be controlled directly. It is also impractical and most of the time impossible to eliminate the noise factors (Dehnad,1989). Taguchi experimental design will help to reduce the effect of noise factors.

Taguchi adopted the concept of signal to noise ratio to reduce the effect of noise factors in the experiment. The desired value or mean response value is represented by signal (S).The undesirable value or standard deviation is denoted by noise (N). The variation present in the response variable or the component of noise factor is represented by S/N ratio. Objective functions are classified into 3 types for design of experiment applications by Taguchi. They are "smaller is better," the "larger is better," and "the nominal is best." Since in the algorithms developed the objectif is to minimize the flow time. The "smaller is better" type is used, his corresponding S/N ratio is given by the following equation :

$$\frac{S}{N} = -10 \log_{10}(k)^2, \tag{23}$$

where k is the objective function value.

The parameters of the genetic algorithm are : population size (PS), maximum number of generations (MNG), crossover probability (Pc), mutation probability (Pm), number of elite individuals (NEI) and the number of individuals on which local research is applied (NILS). The parameters of the iterated local search algorithm are : maximum number of iterations of sequence operator (MISO), maximum number of iterations of affectation operator (MIAO), maximum number of perturbations of perturbation operator (MPPO) and maximum number of perturbations of the iterated local search algorithm (MPILS). Taguchi design which is based on fractional factorial experiments is used to set the parameters of the genetic algorithm and those of the iterated local search method.

Minitab 19 is used to employ the Taguchi method. Under the menu options of Minitab, Stat-DOE-Taguchi Design Create Design is selected. For 5 levels of 6 factors, L25 orthogonal array is used. This orthogonal array lists the different combinations of factors at different levels at which the response values of experiments have to be determined.

For different combinations of the factor levels, each example is solved five times and the mean response was used in the analysis. Figures 25, 27, 29 and 31 show the main effect plot of S/N ration for different parameter levels of the proposed algorithms. Based on "smaller is better" definition for the signal to noise ratio, optimum values of the genetic algorithm and iterated local search algorithm parameters are shown in Tables 18-21.

Level	PS	MNG	NEI	NILS	Pc	Pm
1	100	500	1	1	0.1	0.1
2	200	600	50	50	0.2	0.2
3	500	800	100	100	0.7	0.3
4	800	1000	500	500	0.8	0.8
5	1000	1200	1000	1000	0.9	0.9

Table 8: Parameters of genetic algorithm 1.

Level	PS	MNG	NEI	NILS	Pc	Pm
1	100	500	1	1	0.1	0.1
2	200	600	80	80	0.2	0.2
3	500	800	100	100	0.7	0.3
4	800	1000	500	500	0.8	0.8
5	1000	1200	1000	1000	0.9	0.9

Table 9: Parameters of genetic algorithm 2.

Table 10: Genetic algorithm parameters and levels of Taguchi design.

Level	MPILS	MISO	MIAO	MPPO
1	500	100	100	50
2	600	200	200	100
3	800	500	500	200
4	1000	800	800	500
5	1200	1000	1000	800

Table 11: Parameters of iterated local search 1.

Level	MPILS	MISO	MIAO	MPPO
1	500	100	100	50
2	600	200	200	100
3	800	500	500	200
4	1000	800	800	500
5	1200	1000	1000	800

Table 12: Parameters of iterated local search 2.

Table 13: Iterated local search parameters and levels of Taguchi design.

PS	MNG	NEI	NILS	Pc	Pm	Fitness	S/N ratio
100	500	1	1	0.9	0.1	990.2	-59.914
100	600	100	1	0.8	0.2	988.9	-59.903
100	800	100	100	0.7	0.3	987.7	-59.892
100	1000	50	100	0.8	0.1	987.2	-59.888
100	1200	100	50	0.9	0.2	987.4	-59.889
200	500	50	100	0.8	0.9	997.7	-59.979
200	600	100	50	0.9	0.1	989.8	-59.910
200	800	50	100	0.1	0.2	1000.1	-60.000
200	1000	100	1	0.2	0.7	997.5	-59.978
200	1200	1	50	0.7	0.8	998.3	-59.985
500	500	100	500	0.2	0.8	1004.9	-60.042
500	600	500	1	0.7	0.9	998.9	-59.990
500	800	500	100	0.9	0.1	987.1	-59.887
500	1000	50	50	0.8	0.2	986.8	-59.884
500	1200	50	500	0.1	0.7	999.2	-59.993
800	500	500	50	0.9	0.7	997.2	-59.975
800	600	500	100	0.1	0.8	1005.8	-60.050
800	800	1	500	0.2	0.9	999.1	-59.992
800	1000	50	500	0.7	0.1	987.3	-59.888
800	1200	50	50	0.8	0.2	987.1	-59.887
1000	500	1000	500	0.9	0.2	987.4	-59.889
1000	600	1	1000	0.8	0.7	994.6	-59.952
1000	800	50	1	0.9	0.8	995.8	-59.963
1000	1000	100	50	0.1	0.2	1004.6	-60.039
1000	1200	500	100	0.2	0.1	1003.5	-60.030

Table 14: Experimental results of Taguchi design GA 1.

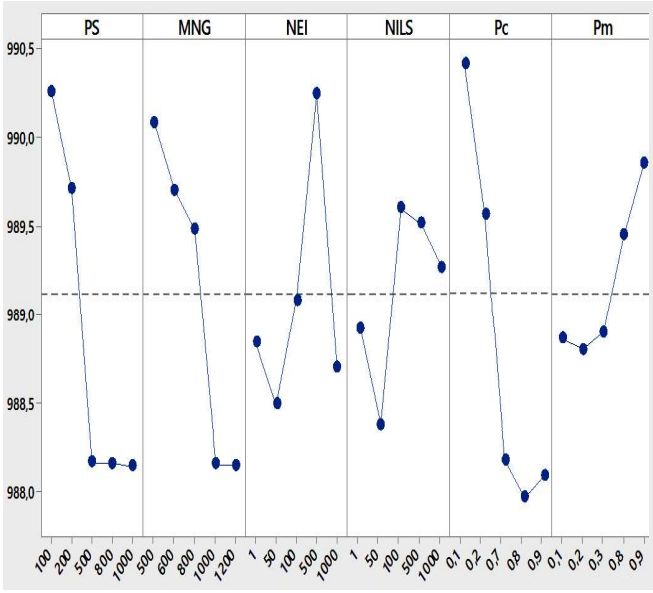

Figure 24: Main effect plot of means GA 1.

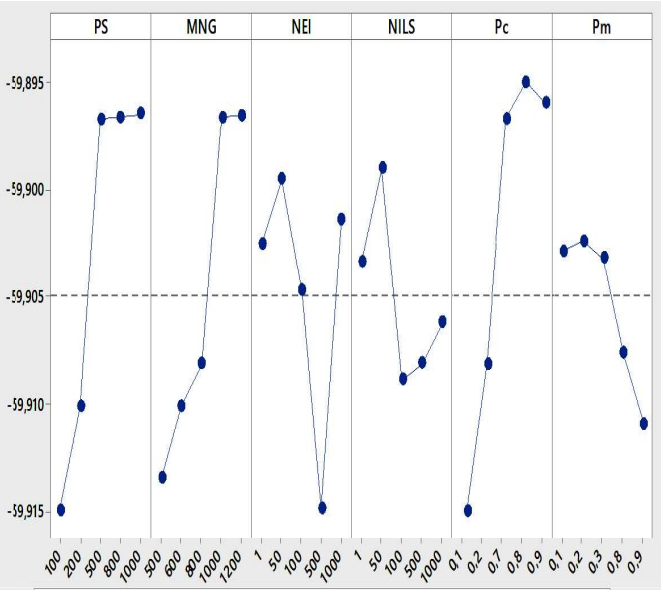

Figure 25: Main effect plot of S/N ratios GA 1.

PS	MNG	NEI	NILS	Pc	Pm	Fitness	S/N ratio
100	500	1	1	0.9	0.1	989.2	-59.905
100	600	100	1	0.8	0.2	987.5	-59.890
100	800	100	100	0.7	0.3	986.9	-59.885
100	1000	80	100	0.8	0.1	986.7	-59.883
100	1200	100	80	0.9	0.2	986.8	-59.884
200	500	80	100	0.8	0.9	996.4	-59.968
200	600	100	80	0.9	0.1	988.6	-59.900
200	800	80	100	0.1	0.2	999.7	-59.997
200	1000	100	1	0.2	0.7	996.2	-59.966
200	1200	1	80	0.7	0.8	997.4	-59.977
500	500	100	500	0.2	0.8	1003.5	-60.030
500	600	500	1	0.7	0.9	997.3	-59.976
500	800	500	100	0.9	0.1	986.7	-59.883
500	1000	80	80	0.8	0.2	986.5	-59.881
500	1200	80	500	0.1	0.7	999.2	-59.993
800	500	500	80	0.9	0.7	996.7	-59.971
800	600	500	100	0.1	0.8	1004.6	-60.039
800	800	1	500	0.2	0.9	998.8	-59.989
800	1000	80	500	0.7	0.1	986.5	-59.881
800	1200	80	80	0.8	0.2	986.3	-59.880
1000	500	1000	500	0.7	0.2	986.7	-59.883
1000	600	1	1000	0.8	0.7	993.8	-59.945
1000	800	80	1	0.9	0.8	994.6	-59.952
1000	1000	100	80	0.1	0.9	1003.4	-60.029
1000	1200	500	100	0.2	0.1	1002.2	-60.019

Table 15: Experimental results of Taguchi design GA 2.

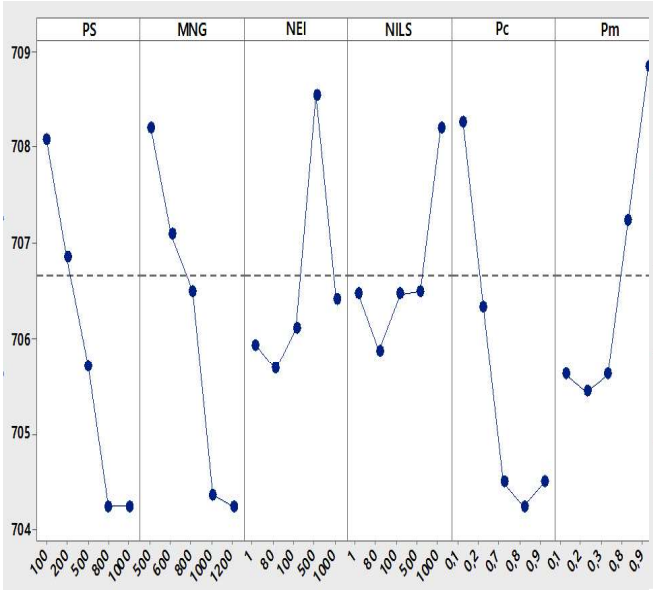

Figure 26: Main effect plot of means GA 2.

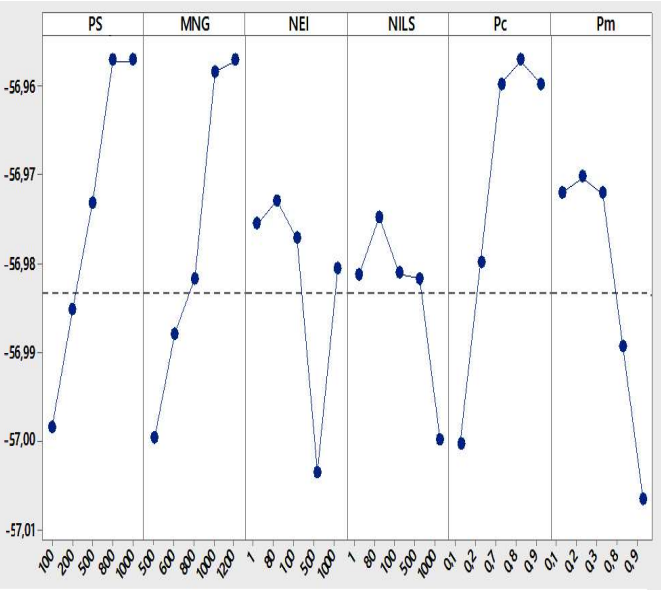

Figure 27: Main effect plot of S/N ratios GA 2.

MPILS	MISO	MIAO	MPPO	Flow time	S/N ratio
500	100	100	50	922.6	-59.301
500	200	500	50	921.1	-59.286
500	500	500	200	919.8	-59.273
500	800	200	200	919.2	-59.268
500	1000	500	100	919.4	-59.270
600	100	200	200	931.3	-59.381
600	200	500	100	922.2	-59.296
600	500	200	200	934.0	-59.407
600	800	500	50	931.0	-59.379
600	1000	100	100	932.0	-59.388
800	100	500	500	939.5	-59.458
800	200	800	50	932.6	-59.394
800	500	800	200	919.1	-59.267
800	800	200	100	932.9	-59.396
800	1000	200	800	933.0	-59.397
1000	100	800	100	930.7	-59.376
1000	200	1000	200	940.6	-59.468
1000	100	100	50	918.2	-59.258
1000	800	200	500	919.3	-59.269
1000	1000	200	100	919.1	-59.267
1200	100	1000	500	919.4	-59.270
1200	200	100	800	927.7	-59.348
1200	200	200	100	929.1	-59.361
1200	800	500	50	939.2	-59.455
1200	1000	1000	200	937.9	-59.443

Table 16: Experimental results of Taguchi design ILS 1.

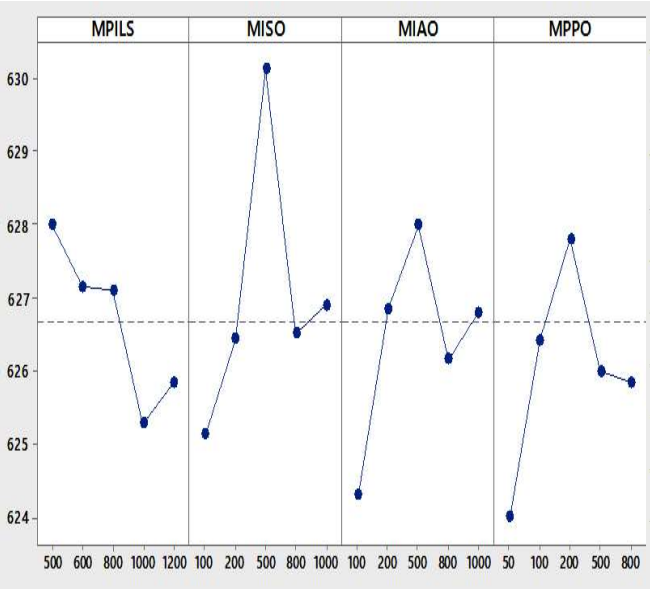

Figure 28: Main effect plot of means ILS 1.

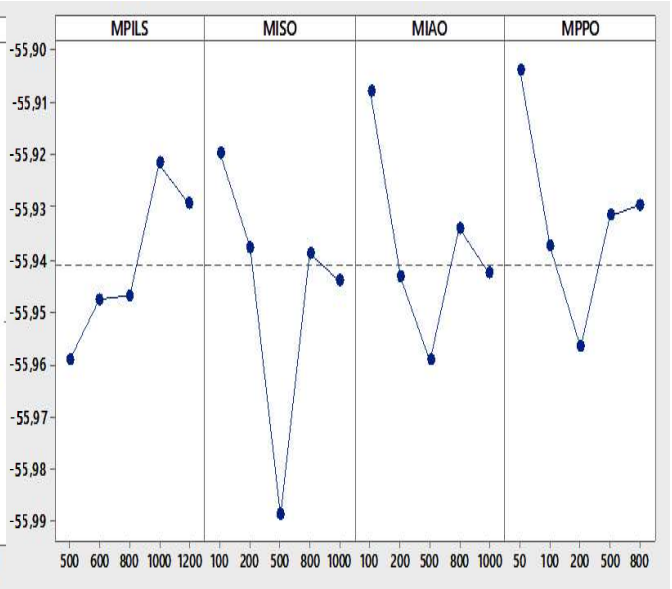

Figure 29: Main effect plot of S/N ratios ILS 1.

MPILS	MISO	MIAO	MPPO	Flow time	S/N ratio
500	100	100	50	908.4	-59.166
500	200	500	50	906.5	-59.147
500	500	500	200	906.9	-59.152
500	800	200	200	906.7	-59.149
500	1000	500	100	906.8	-59.150
600	100	200	200	916.7	-59.245
600	200	500	100	907.7	-59.159
600	500	200	200	920.5	-59.280
600	800	500	50	916.5	-59.242
600	1000	100	100	917.9	-59.255
800	100	500	500	924.9	-59.321
800	200	800	50	917.7	-59.254
800	500	800	200	906.5	-59.147
800	800	200	100	919.9	-59.275
800	1000	200	800	917.0	-59.248
1000	100	800	100	926.1	-59.333
1000	200	1000	200	919.5	-59.271
1000	100	100	50	906.5	-59.147
1000	800	200	500	906.3	-59.145
1000	1000	200	100	906.7	-59.149
1200	100	1000	500	913.7	-59.216
1200	200	100	800	914.6	-59.225
1200	200	200	100	905.5	-59.137
1200	800	500	50	924.7	-59.320
1200	1000	1000	200	923.4	-59.307

Table 17: Experimental results of Taguchi design ILS 2.

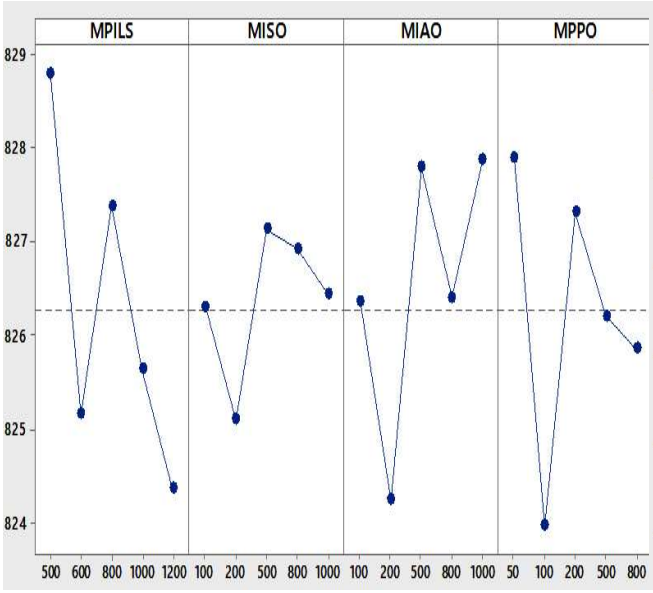

Figure 30: Main effect plot of means ILS 2.

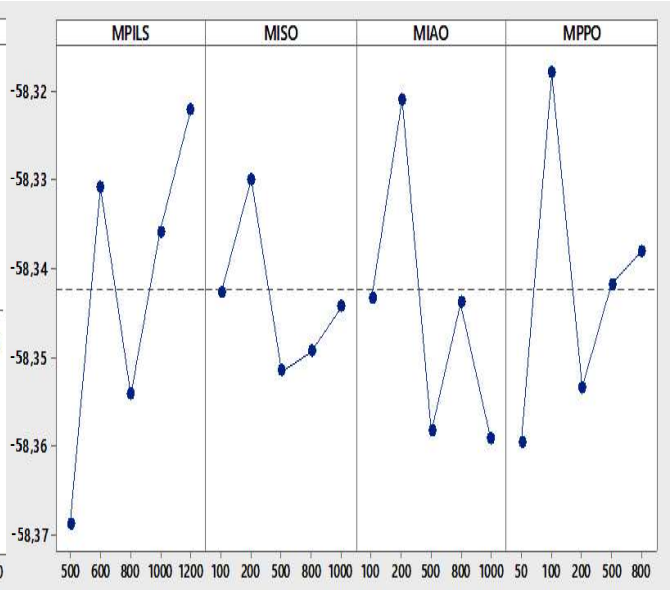

Figure 31: Main effect plot of S/N ratios ILS 2.

PS	MNG	NEI	NILS	Pc	Pm
500	1000	50	50	0.8	0.2

Table 18: Optimal parameters GA 1.

PS	MNG	NEI	NILS	Pc	Pm
800	1200	80	80	0.8	0.2

Table 19: Optimal parameters GA 2.

MPILS	MISO	MIAO	MPPO
1000	100	100	50

Table 20: Optimal parameters ILS 1.

MPILS	MISO	MIAO	MPPO
1200	200	200	100

Table 21: Optimal parameters ILS 2.

8 Performance of developed operators

This subsection presents the results of the improvement percentages of crossover operators of genetic algorithms, and affectation and sequence operators of iterative local search algorithms developed. The goal is to evaluate the performance of these operators for each algorithm. The improvement percentages of the operators were evaluated for the algorithms with the same initial solutions and the same parameters to compare the performance of algorithm operators. For each algorithm, the operator improvement percentages were calculated on more than 100 instances. These instances are of several types : real instances of HCT, randomly generated instances of type HCT and randomly generated instances. For each instance, the minimum, average and the maximum of improvement percentages for each operator are evaluated, and the improvement percentages at the start, center and at the end of algorithms and for each operator are also calculated, as well as the number of times where each operator is applied in the iterations of each algorithm. The figures below present the results of the improvement percentages of the crossover operators of the two genetic algorithms and the improvement percentages of affectation and sequence operators of iterative local search algorithms on an instance. The operator improvement percentages differ from one instance to another. For the genetic algorithms, the improvement percentages of local search algorithms were also evaluated. The goal is to show the evolution of the local search improvement percentages with the operators of two algorithms.

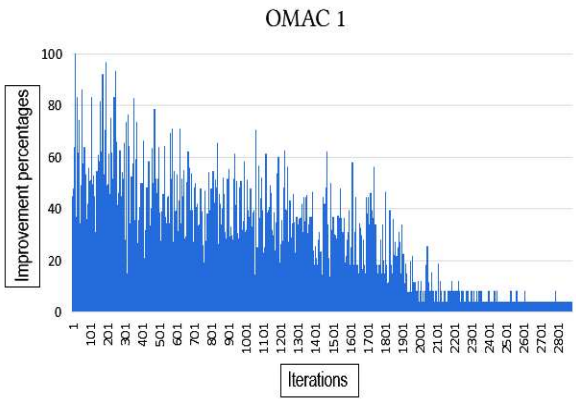

Figure 32: OMAC 1 crossover operator.

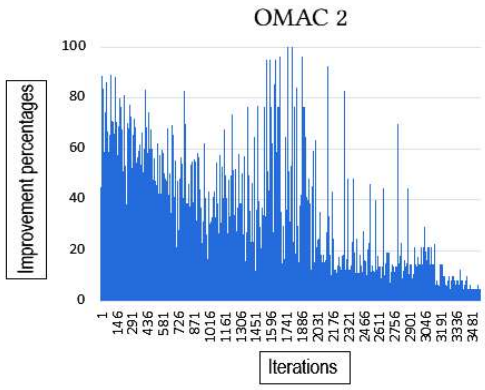

Figure 33: OMAC 2 crossover operator.

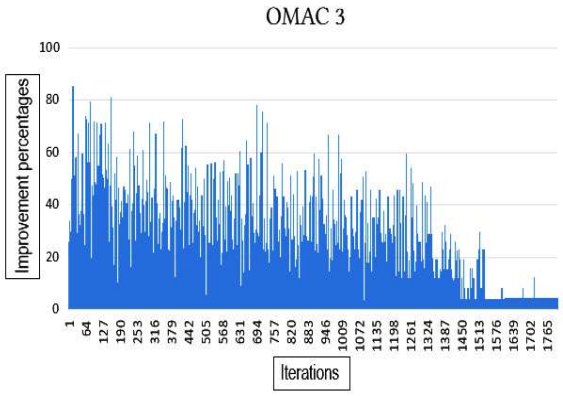

Figure 34: OMAC 3 crossover operator.

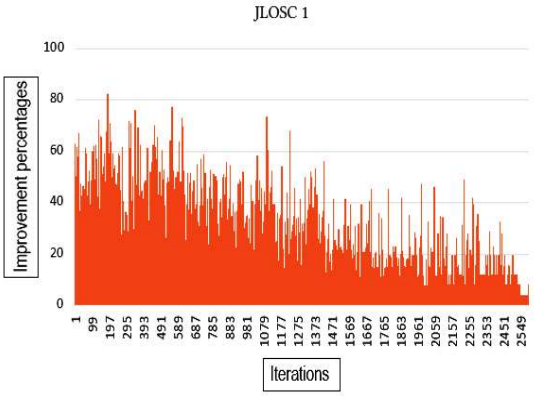

Figure 35: JLOSC 1 crossover operator.

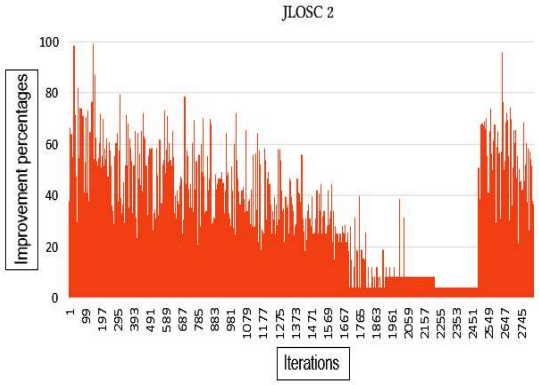

Figure 36: JLOSC 2 crossover operator.

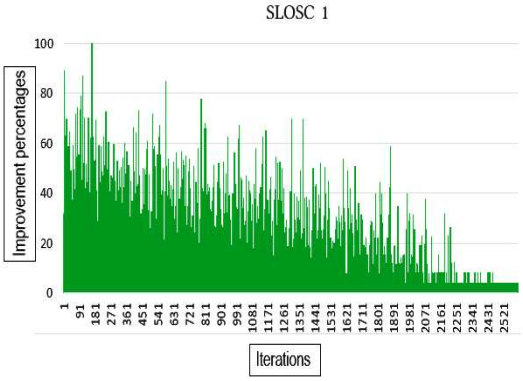

Figure 37: SLOSC 1 crossover operator.

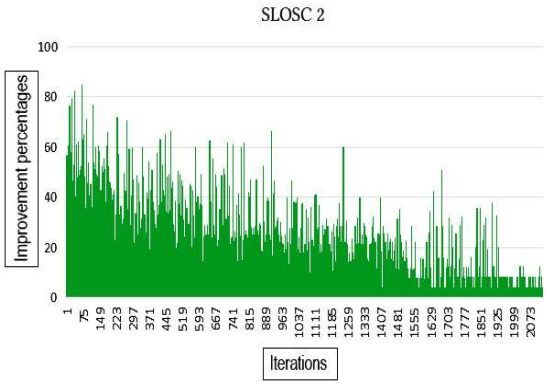

Figure 38: SLOSC 2 crossover operator.

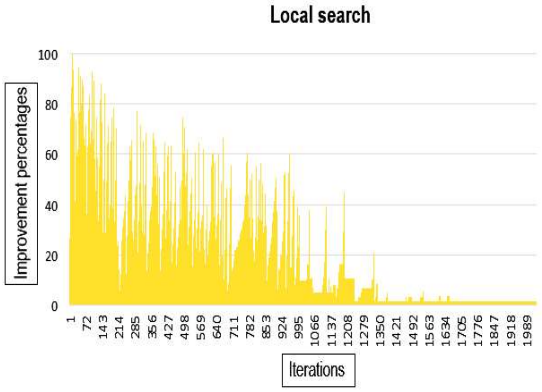

Figure 39: Local search with operators OMAC 1, JLOSC 1 and SLOSC 1.

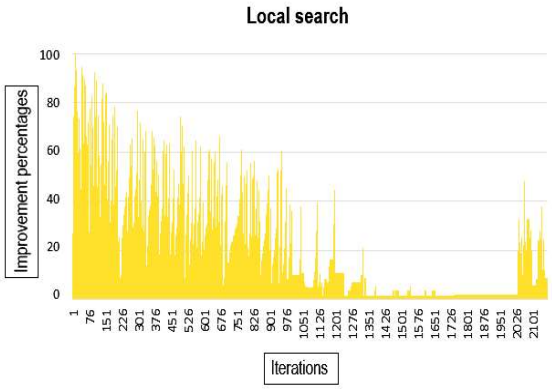

Figure 40: Local search with operators OMAC 2, OMAC 3, JLOSC 2 and SLOSC 2.

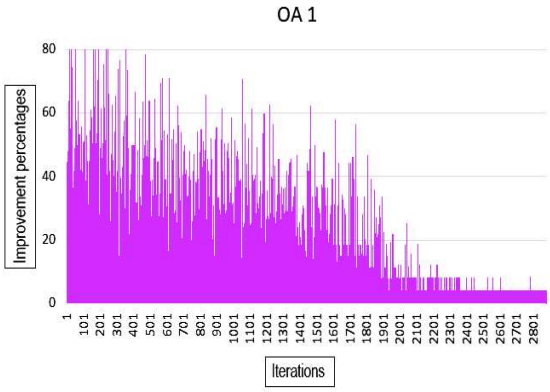

Figure 41: Opérateur d'affectation AO 1.

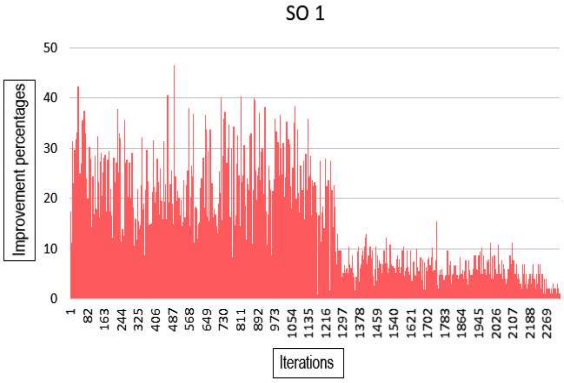

Figure 42: Opérateur de séquencement SO 1.

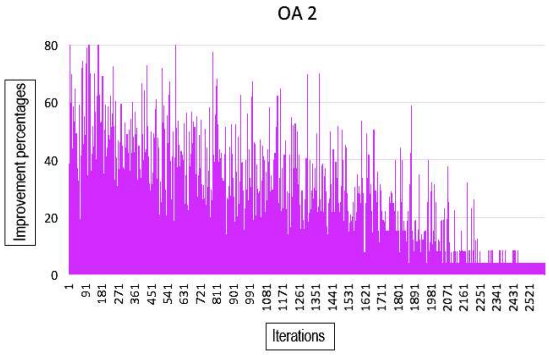

Figure 43: Opérateur d'affectation AO 2.

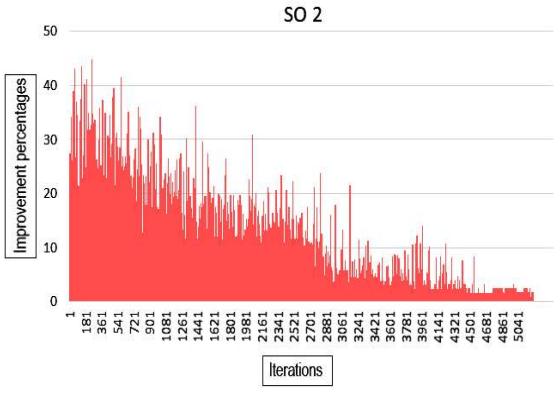

Figure 44: Opérateur de séquencement SO 2.

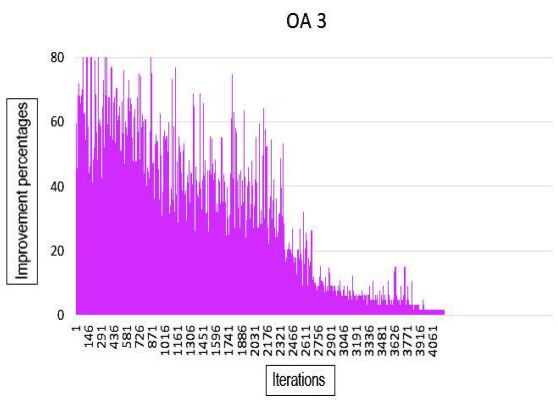

Figure 45: Opérateur d'affectation AO 3.

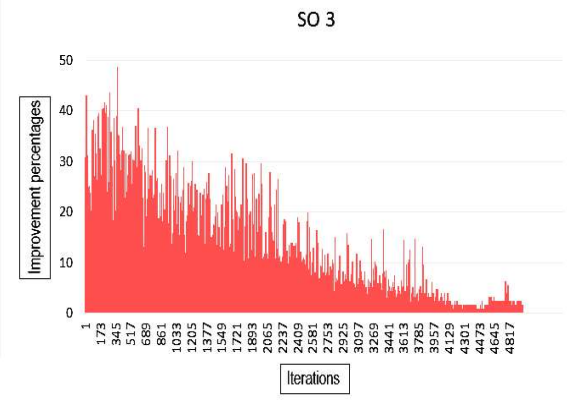

Figure 46: Opérateur de séquencement SO 3.

9 Computational results of developed algorithms

The genetic algorithms presented previously have been implemented using the JAVA programming language and have been tested on more than 100 instances of different types : adapted instances of literature (Behnkel et Geiger (2012), Azzouz *et al.* (2017), Buddala et Mahapatra (2018), Shen *et al.*(2018)), randomly generated instances, randomly generated instances of type HCT and real instances of HCT. The optimization criteria are based on three sub-criteria : quality of solutions, rapidity and stability of algorithms. The quality and efficiency of the solutions obtained has been proven by comparing the solutions obtained with the algorithms and the optimal solutions of the mathematical model. The quality of solutions has also been proven by evaluating the degrees of similarity between solutions of metaheuristics and also between solutions obtained by metaheuristics and real solutions of some examples of production days. The degrees of similarity between solutions are evaluated based on the distances of Manhattan and Hamming which are represented in Table 22. By comparing the real solution of an example of a production day (Table 23) with the solution obtained with the genetic algorithm, it was found that the gap between the two solutions is $-12,565\%$, which shows the quality of the algorithms developed. Regarding the rapidity of algorithms, the computation times of the algorithms were compared with those of the mathematical model. The stability of the algorithms was measured by launching them 10 runs for each instance tested. Based on the criteria : quality, rapidity and stability, the performance of the algorithms developed has been proven by testing them on several types of instances of different sizes. The results are presented in detail below.

Distance	Mathematical formulas
Gap	$D_1 = \lvert 100 * \frac{\sum_i^n C_i^1 - \sum_i^n C_i^2}{\sum_i^n C_i^2} \rvert$
Manhattan	$D_2 = \frac{\sum_{p=1}^{P} \sum_{r=1}^{3} \lvert Sol_{pr}^1 - Sol_{pr}^2 \rvert}{P}$
Manhattan	$D_3 = \frac{\sum_{i=1}^{n} \lvert S_i^1 - S_i^2 \rvert}{P}$
Manhattan	$D_4 = \frac{\sum_{i=1}^{n} \lvert C_i^1 - C_i^2 \rvert}{P}$
Hamming	$D_5 = d = \frac{\sum_{i=1}^{n} X_i}{P}, X = (X_1, ..., X_i, ...X_n)$ $X_i = \begin{cases} 1, & \text{if starting dates of job } i \text{ in the compared solutions are different} \\ 0, & \text{otherwise} \end{cases}$
Hamming	$D_6 = \frac{\sum_{i=1}^{n} X_i}{P}, X = (X_1, ..., X_i, ...X_n)$ $X_i = \begin{cases} 1, & \text{if completion dates of job } i \text{ in the compared solutions are different} \\ 0, & \text{otherwise} \end{cases}$
Hamming	$D_7 = D_6^d$
Hamming	$D_8 = D_5 * D_6$
Hamming	$D_9 = D_5 + D_6$
Hamming	$D_{10} = \frac{\sum_{p=1}^{P} X_p}{P}, X = (X_1, ..., X_p, ...X_n)$ $X_p = \begin{cases} 1, & \text{if columns } p \text{ in the compared solution vectors are not identical} \\ 0, & \text{otherwise} \end{cases}$
Hamming	$D_{11} = b = \frac{\sum_{p=1}^{P} X_p}{P}, X = (X_1, ..., X_p, ...X_n)$ $X_p = \begin{cases} 1, & \text{if the operation in columns } p \text{ of solution vectors is not assigned to the same machine} \\ 0, & \text{otherwise} \end{cases}$
Hamming	$D_{12} = D_{10}^b$
Hamming	$D_{13} = D_{10} * D_{11}$
Hamming	$D_{14} = D_{10} + D_{11}$
Hamming	$D_{15} = \frac{\sum_{i=1}^{n} X_i}{P}, X = (X_1, ..., X_i, ...X_n)$ $X_i = \begin{cases} 1, & \text{if the sub-lots of jobs } i \text{ are not treated similarly in the compared solutions} \\ 0, & \text{otherwise} \end{cases}$

$\sum_i^n C_i^1, \sum_i^n C_i^2$: flow times of compared solutions, P : size of vector solution, Sol^1, Sol^2 : two compared solutions, S_i^1, S_i^2 : starting times of job i in compared solutions, C_i^1, C_i^2 : completion times of job i in compared solutions.

Table 22: Metrics used to measure the degrees of similarity between solutions.

- Instance data	
- Number of dishes	**82**
- Total number of sub-lots of dishes	**92**
- Total number of operations	**370**
- Number of material resources	**29**
- Average number of meals produced	**5 000**
- Real solution	**901,97 h**
- Genetic algorithm solution	**788,64 h**
- Gap between real and genetic algorithm solutions	**- 12,565 %**

Table 23: Comparison between real and genetic algorithm solutions on an example of a production day.

D_1	D_2	D_3	D_4	D_5	D_6	D_7	D_8	D_9	D_{10}	D_{11}	D_{12}	D_{13}	D_{14}	D_{15}
14.37	38.80	0.81	0.93	1.12	2.25	2.48	2.52	3.38	1.87	0.48	1.35	0.89	2.36	1.08

Table 24: Degrees of similarity between real and genetic algorithm solutions.

9.1 Results of genetic algorithm 1

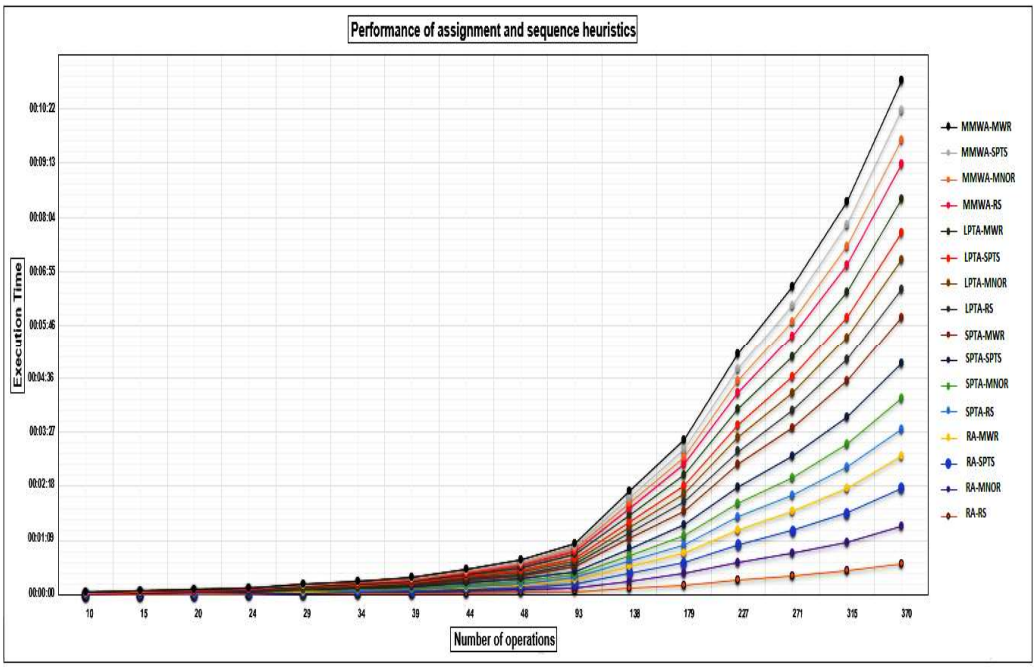

Figure 47: Performance of GA 1 in terms of rapidity with the different combinations of heuristics for the generation of initial population.

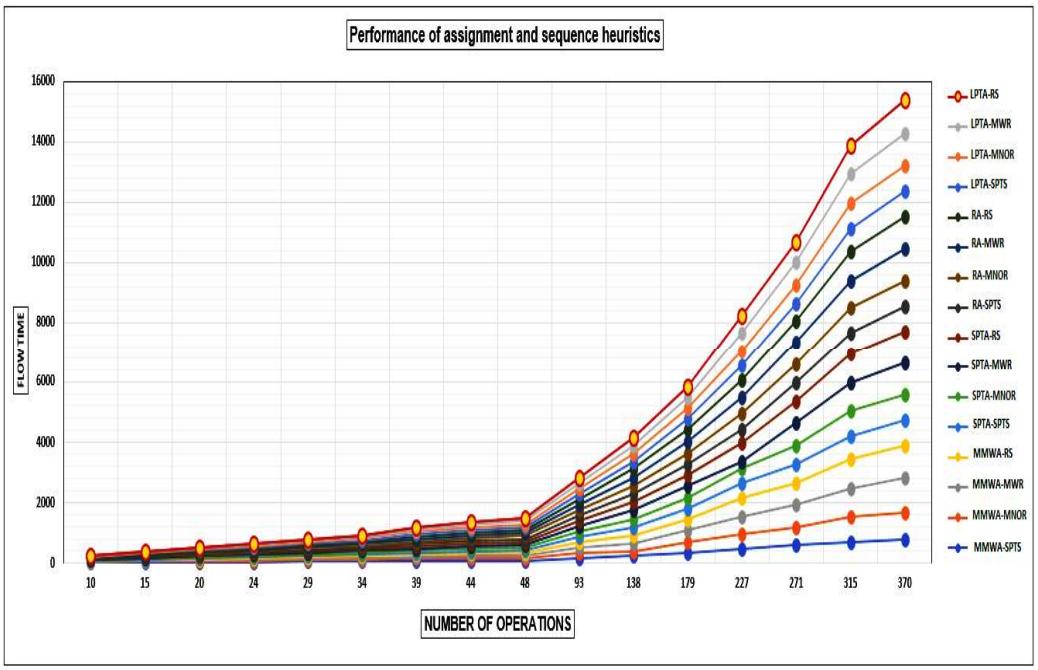

Figure 48: Performance of GA 1 in terms of quality with the different combinations of heuristics for the generation of initial population.

Job	Sub-lot	Operation	Machine	$\sum C_i$	Time	Gap 1	Gap 2	Stability
2	2	10	29	15.3 h	0.1 s	0.0 %	0.0 %	100 %
3	3	15	29	24.4 h	0.2 s	0.0 %	0.0 %	100 %
4	4	20	29	32.8 h	0.3 s	0.0 %	0.0 %	100 %
5	5	24	29	39.9 h	0.4 s	0.0 %	0.0 %	100 %
6	6	29	29	49.2 h	0.6 s	0.0 %	0.0 %	100 %
7	7	34	29	57.6 h	0.8 s	0.0 %	0.0 %	100 %
8	10	39	29	69.4 h	1 s	0.0 %	0.0 %	100 %
9	11	44	29	77.7 h	1.5 s	-	0.0 %	100 %
10	12	48	29	85.0 h	2 s	-	0.0 %	100 %
20	22	93	29	162.4 h	30 s	-	0.0 %	100 %
30	32	138	29	252.9 h	60 s	-	0.0 %	100 %
40	42	179	29	339.7 h	1.5 mn	-	0.0 %	100 %
50	58	227	29	471.6 h	2.3 mn	-	- 1.46 %	100 %
60	68	271	29	590.4 h	3 mn	-	- 0.59 %	100 %
70	78	315	29	682.8 h	3.8 mn	-	0.77 %	100 %
82	92	370	29	798.3 h	5 mn	-	1.23 %	100 %

Gap 1 : gap between optimal solutions and those of genetic algorithm 1, Gap 2 : gap between solutions of genetic algorithms 1 and 2.

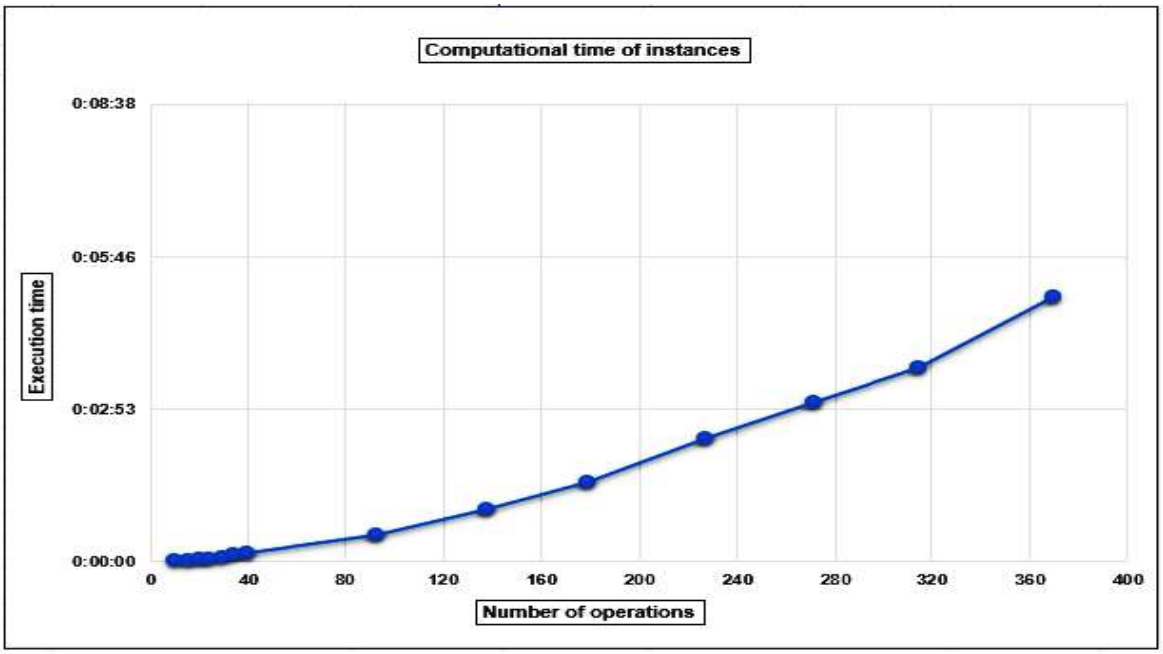

Figure 49: Computational results of genetic algorithm 1 on real instances of HCT.

Job	Sub-lot	Operation	Machine	$\sum C_i$	Time	Gap 1	Gap 2	Stability
100	100	342	29	704.7 h	4 mn	-	- 0.07 %	99.97 %
110	395	402	29	986.7 h	8 mn	-	0.04 %	99.96 %
90	90	324	29	696.5 h	4.5 mn	-	- 0.01 %	99.98 %
80	277	286	29	525.6 h	2.3 mn	-	0.02 %	100 %
60	208	200	29	248.8 h	40 s	-	0.12 %	100 %
50	50	173	29	318.9 h	1 mn	-	0.19 %	100 %
20	20	74	29	158.6 h	25 s	-	0.25 %	100 %
15	15	59	29	421.5 h	2 mn	-	0.09 %	100 %
10	10	31	29	215.4 h	40 s	-	0.14 %	100 %
5	5	19	29	368.8 h	2.5 mn	-	0.16 %	100 %

Gap 1 : gap between optimal solutions and those of genetic algorithm 1, Gap 2 : gap between solutions of genetic algorithms 1 and 2.

Table 25: Computational results of genetic algorithm 1 on randomly generated instances of type HCT.

Job	Sub-lot	Operation	Machine	$\sum C_i$	Time	Gap 1	Gap 2	Stability
3	12	11	6	117	3 s	0.0 %	100%	100%
3	8	13	6	85	5 s	0.0 %	100%	100 %
3	10	8	6	45	1.5 s	0.0 %	100%	100%
3	11	11	6	55	4 s	0.0 %	100%	100 %
3	12	12	6	117	24 s	0.0 %	100%	100 %
3	11	13	6	78	6 s	0.0 %	100%	100 %
3	10	12	6	88	42 s	0.0 %	100%	100 %
3	10	13	6	103	56 s	0.0 %	100%	100 %
3	10	10	6	76	6 s	0.0 %	100%	100 %
3	8	9	6	44	5 s	0.0 %	100%	100%

Gap 1 : gap between optimal solutions and those of genetic algorithm 1, Gap 2 : gap between solutions of genetic algorithms 1 and 2.

Table 26: Computational results of genetic algorithm 1 on randomly generated instances.

9.2 Results of genetic algorithm 2

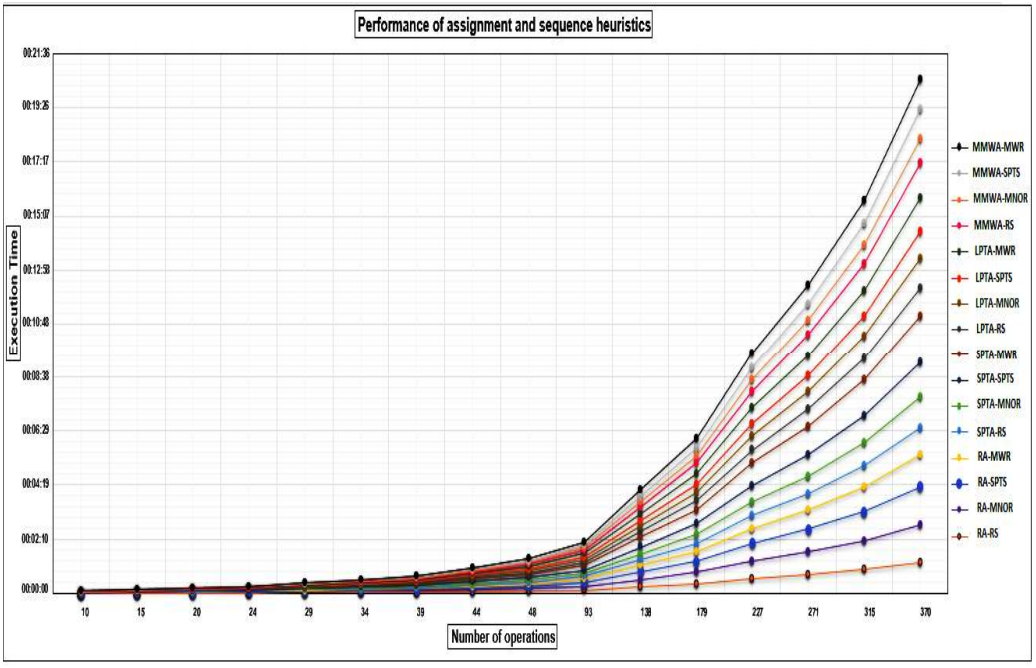

Figure 50: Performance of GA 2 in terms of rapidity with the different combinations of heuristics for the generation of initial population.

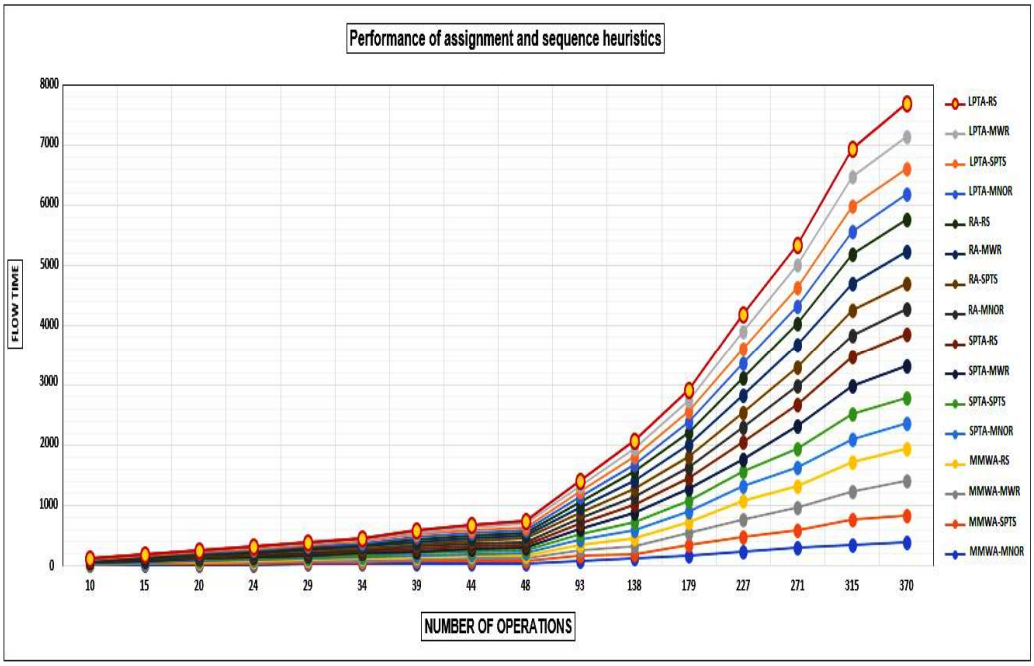

Figure 51: Performance of GA 2 in terms of quality with the different combinations of heuristics for the generation of initial population.

Job	Sub-lot	Operation	Machine	$\sum C_i$	Time	Gap 1	Gap 2	Stability
2	2	10	29	15.3 h	0.3 s	0.0 %	0.0 %	100 %
3	3	15	29	24.4 h	0.5 s	0.0 %	0.0 %	100 %
4	4	20	29	32.8 h	0.8 s	0.0 %	0.0 %	100 %
5	5	24	29	39.9 h	1 s	0.0 %	0.0 %	100 %
6	6	29	29	49.2 h	1.4 s	0.0 %	0.0 %	100 %
7	7	34	29	57.6 h	1.8 s	0.0 %	0.0 %	100 %
8	10	39	29	69.4 h	2 s	0.0 %	0.0 %	100 %
9	11	44	29	77.7 h	3 s	-	0.0 %	100 %
10	12	48	29	85.0 h	4 s	-	0.0 %	100 %
20	22	93	29	162.4 h	1 mn	-	0.0 %	100 %
30	32	138	29	252.9 h	2 mn	-	0.0 %	100 %
40	42	179	29	339.7 h	3 mn	-	0.0 %	100 %
50	58	227	29	478.6 h	4.6 mn	-	1.48%	100 %
60	68	271	29	593.9 h	6 mn	-	0.59 %	100 %
70	78	315	29	677.6 h	7.8 mn	-	- 0.77 %	100 %
82	92	370	29	788.6 h	10 mn	-	- 1.22 %	100 %

Gap 1 : gap between optimal solutions and those of genetic algorithm 2, Gap 2 : gap between solutions of genetic algorithms 1 and 2.

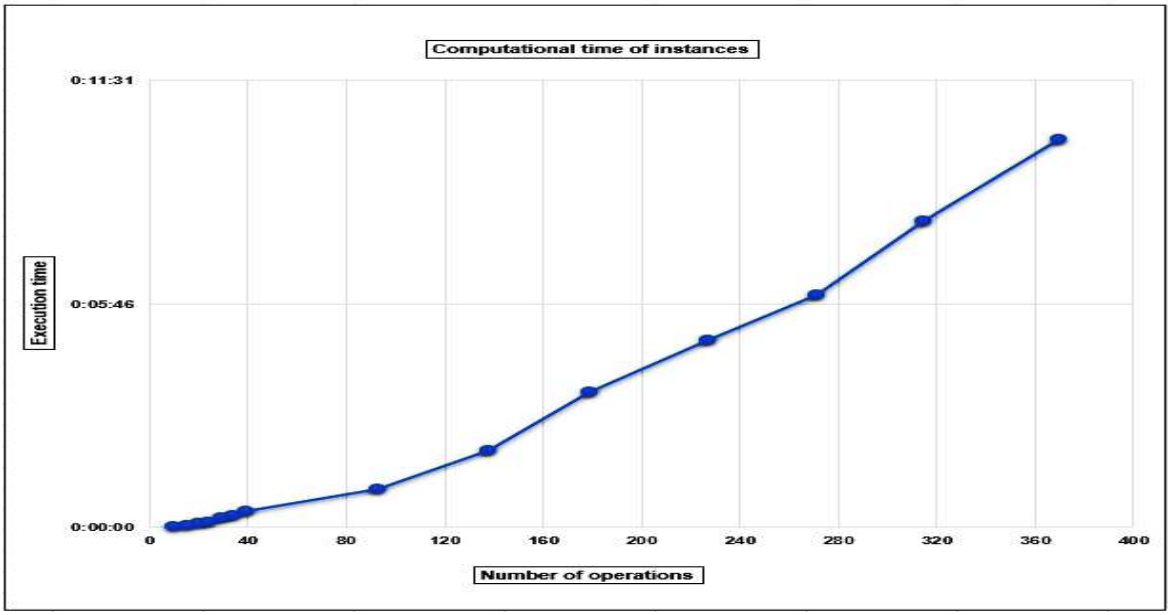

Figure 52: Computational results of genetic algorithm 2 on real instances of HCT.

Job	Sub-lot	Operation	Machine	$\sum C_i$	Time	Gap 1	Gap 2	Stability
100	100	342	29	705.1 h	6 mn	-	0.06 %	99.98 %
110	395	402	29	986.2 h	12 mn	-	- 0.07 %	99.97 %
90	90	324	29	696.5 h	8 mn	-	- 0.04 %	99.99 %
80	277	286	29	525.5 h	4 mn	-	- 0.04 %	100 %
60	208	200	29	248.5 h	1 mn	-	- 0.12 %	100 %
50	50	173	29	318.3 h	1.8 mn	-	- 0.19 %	100 %
20	20	74	29	158.2 h	45 s	-	- 0.25 %	100 %
15	15	59	29	421.1 h	3.8 s	-	- 0.09 %	100 %
10	10	31	29	215.1 h	75 s	-	- 0.14 %	100 %
5	5	19	29	368.1 h	4.2 mn	-	- 0.17 %	100 %

Gap 1 : gap between optimal solutions and those of genetic algorithm 2, Gap 2 : gap between solutions of genetic algorithms 1 and 2.

Table 27: Computational results of genetic algorithm 2 on randomly generated instances of type HCT.

Job	Sub-lot	Operation	Machine	$\sum C_i$	Time	Gap 1	Gap 2	Stability
3	12	11	6	117	6 s	0.0 %	0.0%	100 %
3	8	13	6	85	10 s	0.0 %	0.0%	100 %
3	10	8	6	45	3 s	0.0 %	0.0%	100 %
3	11	11	6	55	8 s	0.0 %	0.0%	100 %
3	12	12	6	117	48 s	0.0 %	0.0%	100 %
3	11	13	6	78	12 s	0.0 %	0.0%	100 %
3	10	12	6	88	82 s	0.0 %	0.0%	100 %
3	10	13	6	103	112 s	0.0 %	0.0%	100 %
3	10	10	6	76	12 s	0.0 %	0.0%	100 %
3	8	9	6	44	10 s	0.0 %	0.0%	100 %

Gap 1 : gap between optimal solutions and those of genetic algorithm 2, Gap 2 : gap between solutions of genetic algorithms 1 and 2.

Table 28: Computational results of genetic algorithm 2 on randomly generated instances.

Instance	D_1	D_2	D_3	D_4	D_5	D_6	D_7	D_8	D_9	D_{10}	D_{11}	D_{12}	D_{13}	D_{14}	D_{15}
2x29x10	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3x29x15	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4x29x20	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5x29x24	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
6x29x29	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
7x29x34	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
8x29x39	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
9x29x44	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
10x29x48	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
20x29x93	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
30x29x138	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
40x29x179	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
50x29x227	1.46	3.74	0.07	0.08	0.41	0.52	0.76	0.21	0.93	0.19	0.05	0.92	0.009	0.24	0.02
60x29x271	0.58	1.60	0.03	0.04	0.54	0.68	0.82	0.36	1.22	0.09	0.02	0.95	0.002	0.11	0.04
70x29x315	0.76	2.10	0.05	0.06	0.72	0.80	0.85	0.57	1.52	0.11	0.03	0.93	0.001	0.14	0.06
82x29x370	1.23	3.32	0.07	0.08	0.82	0.94	0.95	0.77	1.76	0.16	0.04	0.92	0.006	0.056	0.09

Table 29: Degrees of similarity between GA 1 and GA 2 solutions on real instances of HCT.

From the results presented above, we remark that the two genetic algorithms are efficient in terms of quality, rapidity and stability. The performance of these two algorithms depends on the type of instances and their sizes, and it also depends on the choice of heuristics for the generation of initial solutions. The first genetic algorithm is more efficient in terms of quality with the MMWA assignment heuristic and the SPT sequence heuristic, while the second genetic algorithm is more efficient with the MMWA and MNOR heuristics.

The results of the genetic algorithms on different types of instances show that the metaheuristics developed are efficient relative to the quality of solutions. For some instances, the two algorithms found the optimal solutions in a very short computation time compared to the mathematical model. For the instances for which the optimality is not reached, the gaps between the solutions obtained with the

algorithms and the optimal solutions are very small. For the large instances for which the mathematical model has failed to find solutions after more than three hours of execution, the genetic algorithms have found feasible solutions within reasonable computation time. By comparing the two algorithms in terms of rapidity, the first algorithm is faster than the second algorithm, while comparing them to stability both algorithms are stable for real instances of HCT and randomly generated instances, while for the randomly generated instances of type HCT, the second algorithm is more stable than the first algorithm. Comparing the two algorithms in terms of quality of solutions, for some instances the first algorithm is better than the second algorithm, while for other instances the second algorithm is more efficient than the first algorithm.

9.3 Results of iterated local search algorithm 1

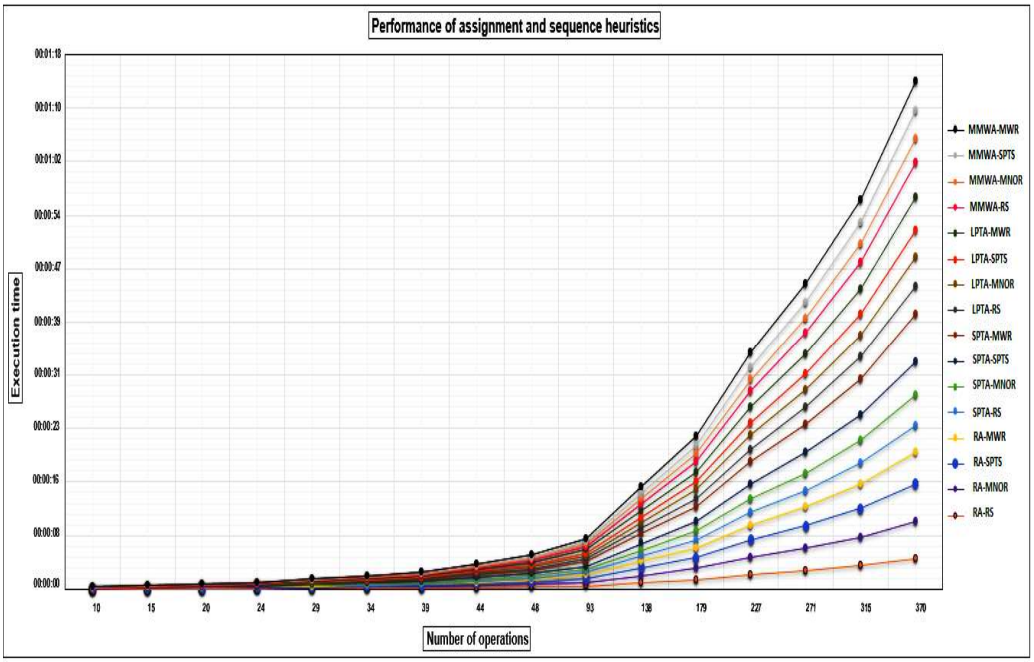

Figure 53: Performance of ILS 1 in terms of rapidity with the different combinations of heuristics for the generation of initial solutions.

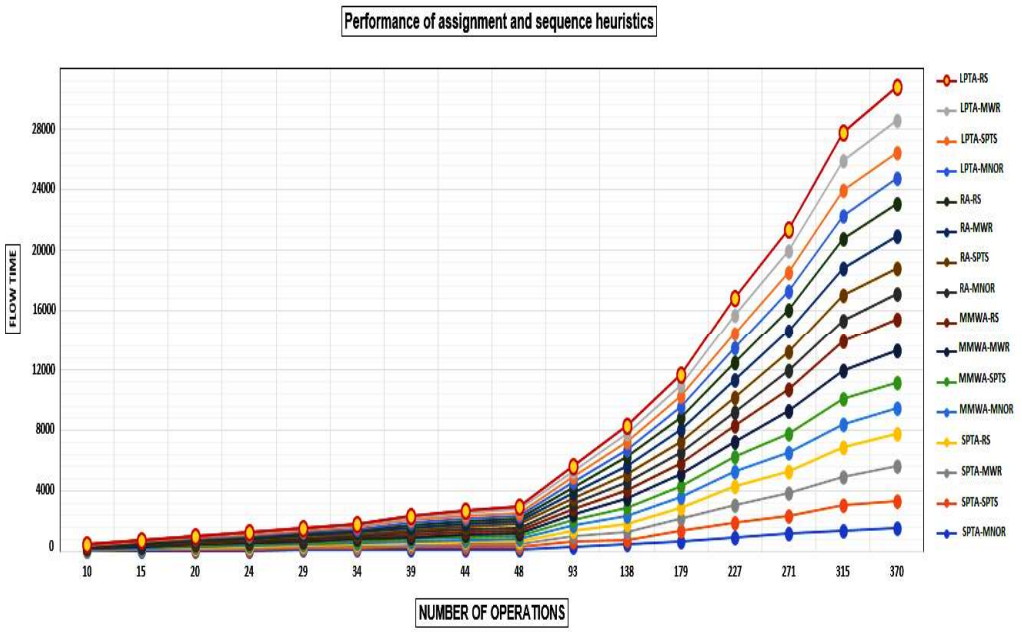

Figure 54: Performance of ILS 1 in terms of quality with the different combinations of heuristics for the generation of initial solutions.

Job	Sub-lot	Operation	Machine	$\sum C_i$	Time	Gap 1	Gap 2	Gap 3	Gap 4	Stability
2	2	10	29	15.3 h	0.05 s	0.0 %	0.0 %	0.0 %	0.0 %	100 %
3	3	15	29	24.4 h	0.08 s	0.0 %	0.0 %	0.0 %	0.0 %	100 %
4	4	20	29	32.8 h	0.1 s	0.0 %	0.0 %	0.0 %	0.0 %	100 %
5	5	24	29	39.9 h	0.14 s	0.0 %	0.0 %	0.0 %	0.0 %	100 %
6	6	29	29	49.2 h	0.2 s	0.0 %	0.0 %	0.0 %	0.0 %	100 %
7	7	34	29	57.6 h	0.3 s	0.0 %	0.0 %	0.0 %	0.0 %	100 %
8	10	39	29	69.4 h	0.5 s	0.0 %	0.0 %	0.0 %	0.0 %	100 %
9	11	44	29	77.7 h	0.75 s	-	0.0 %	0.0 %	0.0 %	100 %
10	12	48	29	86.4 h	1 s	-	1.65 %	1.65 %	0.58 %	100 %
20	22	93	29	166.5 h	10 s	-	2.52%	2.52 %	0.42 %	100 %
30	32	138	29	262.3 h	20 s	-	3.72 %	3.72 %	0.65 %	100 %
40	42	179	29	354.9 h	28 s	-	4.47 %	4.47 %	0.74 %	100 %
50	58	227	29	490.6 h	35 s	-	4.03 %	2.51 %	1.03 %	100 %
60	68	271	29	624.4 h	48 s	-	5.76 %	5.14 %	1.25 %	100 %
70	78	315	29	720.7 h	56 s	-	5.55 %	6.36 %	0.77 %	100 %
82	92	370	29	846.5 h	75 s	-	6.04 %	7.34 %	0.68 %	100 %

Gap 1 : gap between optimal solutions and those of ILS 1, Gap 2 : gap between solutions of ILS 1 and GA 1, Gap 3 : gap between solutions of ILS 1 and GA 2, Gap 4 : gap between solutions of ILS 1 and ILS 2.

Table 30: Computational results of iterated local search algorithm 1 on real instances of HCT.

Job	Sub-lot	Operation	Machine	$\sum C_i$	Time	Gap 1	Gap 2	Gap 3	Gap 4	Stability
100	100	342	29	740.2 h	45 s	-	5.02 %	4.96 %	1.05 %	99.94 %
110	395	402	29	1030.5 h	120 s	-	4.42 %	4.48 %	1.19 %	99.93%
90	90	324	29	732.3 h	52 s	-	5.09 %	5.12 %	1.22 %	99.95 %
80	277	286	29	556.7h	42 s	-	5.90 %	5.94 %	1.46 %	100 %
60	208	200	29	262.4 h	18 s	-	5.47 %	5.59 %	1.47 %	100 %
50	50	173	29	336.9 h	24 s	-	5.64 %	5.84 %	1.63 %	100 %
20	20	74	29	165.3 h	8 s	-	4.22 %	4.49 %	0.85 %	100 %
15	15	59	29	432.1 h	28 s	-	2.51 %	2.61 %	0.77 %	100 %
10	10	31	29	219.8 h	15 s	-	2.04 %	2.19 %	0.87 %	100 %
5	5	19	29	376.5 h	32 s	-	2.09 %	2.25 %	0.64 %	100 %

Gap 1 : gap between optimal solutions and those of ILS 1, Gap 2 : gap between solutions of ILS 1 and GA 1, Gap 3 : gap between solutions of ILS 1 and GA 2 and Gap 4 : gap between solutions of ILS 1 and ILS 2.

Table 31: Computational results of iterated local search algorithm 1 on randomly generated instances of type HCT.

Job	Sub-lot	Operation	Machine	$\sum C_i$	Time	Gap 1	Gap 2	Gap 3	Gap 4	Stability
3	12	11	6	123	0.5 s	5.13 %	5.13%	5.13%	0.82 %	100%
3	8	13	6	89	0.6 s	4.71 %	4.71%	4.71%	1.14 %	100 %
3	10	8	6	47	0.1 s	4.44 %	4.44%	4.44%	0.0 %	100%
3	11	11	6	58	0.6 s	5.45 %	5.45%	5.45%	1.75 %	100 %
3	12	12	6	120	2 s	2.56 %	2.56%	2.56%	0.84 %	100 %
3	11	13	6	81	0.8 s	3.85 %	3.85%	3.85%	1.25 %	100 %
3	10	12	6	92	4 s	4.55 %	4.55%	4.55%	1.10 %	100 %
3	10	13	6	106	8 s	2.91 %	2.91%	4.55%	0.95 %	100 %
3	10	10	6	80	0.9 s	5.26 %	5.26%	5.26%	1.27 %	100 %
3	8	9	6	45	0.7 s	2.27 %	2.27%	2.27%	0.0 %	100%

Gap 1 : gap between optimal solutions and those of ILS 1, Gap 2 : gap between solutions of ILS 1 and GA 1, Gap 3 : gap between solutions of ILS 1 and GA 2 and Gap 4 : gap between solutions of ILS 1 and ILS 2.

Table 32: Computational results of iterated local search algorithm 1 on randomly generated instances.

9.4 Results of iterated local search algorithm 2

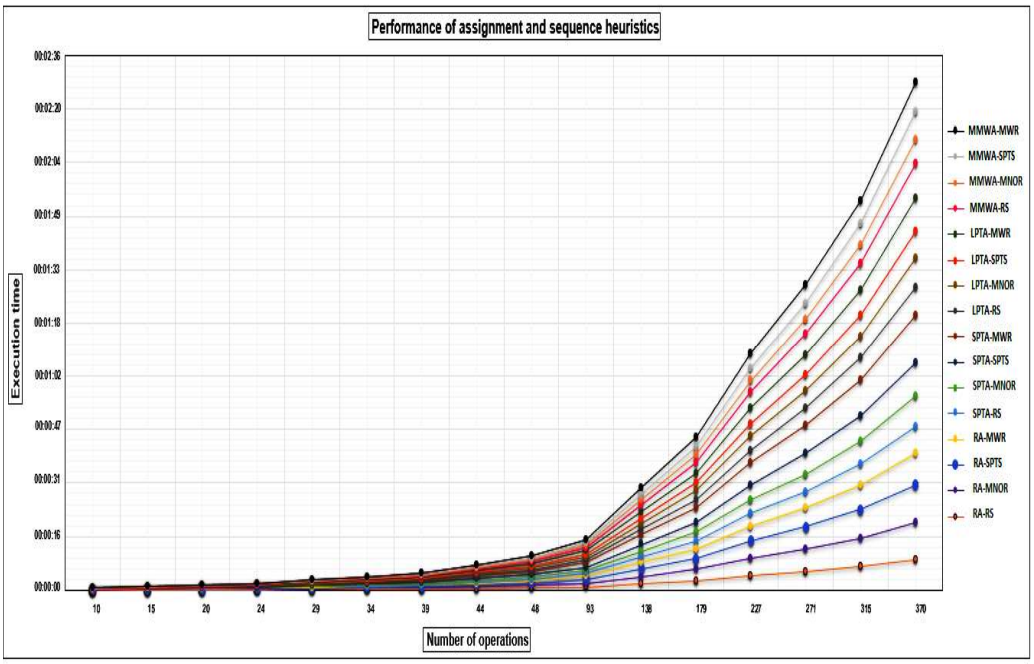

Figure 55: Performance of ILS 2 in terms of rapidity with the different combinations of heuristics for the generation of initial solutions.

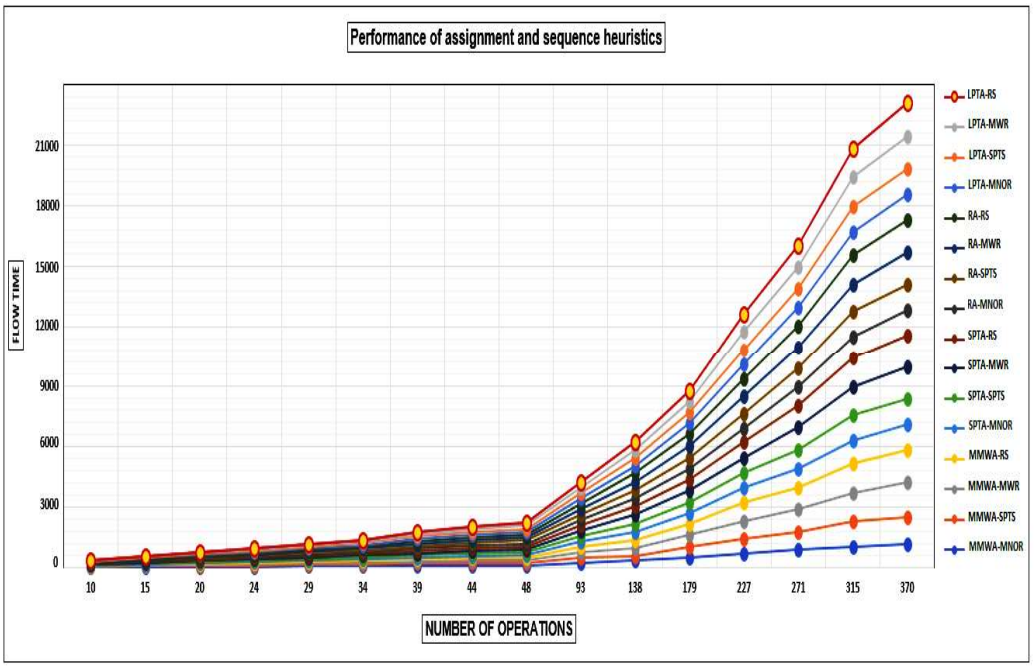

Figure 56: Performance of ILS 2 in terms of quality with the different combinations of heuristics for the generation of initial solutions.

Job	Sub-lot	Operation	Machine	$\sum C_i$	Time	Gap 1	Gap 2	Gap 3	Gap 4	Stability
2	2	10	29	15.3 h	0.12 s	0.0 %	0.0 %	0.0 %	0.0 %	100 %
3	3	15	29	24.4 h	0.18 s	0.0 %	0.0 %	0.0 %	0.0 %	100 %
4	4	20	29	32.8 h	0.24 s	0.0 %	0.0 %	0.0 %	0.0 %	100 %
5	5	24	29	39.9 h	0.32 s	0.0 %	0.0 %	0.0 %	0.0 %	100 %
6	6	29	29	49.2 h	0.44 s	0.0 %	0.0 %	0.0 %	0.0 %	100 %
7	7	34	29	57.6 h	0.58 s	0.0 %	0.0 %	0.0 %	0.0 %	100 %
8	10	39	29	69.4 h	0.7 s	0.0 %	0.0 %	0.0 %	0.0 %	100 %
9	11	44	29	77.7 h	1.6 s	-	0.0 %	0.0 %	0.0 %	100 %
10	12	48	29	85.9 h	1.8 s	-	1.06 %	1.06 %	-0.58 %	100 %
20	22	93	29	165.8 h	16 s	-	2.09 %	2.09 %	-0.42 %	100 %
30	32	138	29	260.6 h	36 s	-	3.04 %	3.04 %	-0.65 %	100 %
40	42	179	29	352.3 h	54 s	-	3.71 %	3.71 %	-0.73 %	100 %
50	58	227	29	485.6 h	65 s	-	2.97 %	1.46 %	-1.02 %	100 %
60	68	271	29	616.7 h	92 s	-	4.45 %	3.84 %	-1.23 %	100 %
70	78	315	29	715.2 h	108 s	-	4.74 %	5.55 %	-0.76 %	100 %
82	92	370	29	840.8 h	146 s	-	5.32 %	6.62 %	-0.67 %	100 %

Gap 1 : gap between optimal solutions and those of ILS 2, Gap 2 : gap between solutions of ILS 2 and GA 1, Gap 3 : gap between solutions of ILS 2 and GA 2 and Gap 4 : gap between solutions of ILS 2 and ILS 1.

Table 33: Computational results of iterated local search algorithm 2 on real instances of HCT.

Job	Sub-lot	Operation	Machine	$\sum C_i$	Time	Gap 1	Gap 2	Gap 3	Gap 4	Stability
100	100	342	29	732.5 h	82 s	-	3.93 %	3.87 %	-1.04 %	99.96 %
110	395	402	29	1018.4 h	230s	-	3.19 %	3.25 %	-1.17 %	99.95 %
90	90	324	29	723.5 h	98 s	-	3.83 %	3.86 %	-1.20 %	99.97 %
80	277	286	29	548.7 h	80s	-	4.38 %	4.41 %	-1.44 %	100 %
60	208	200	29	258.6 h	32 s	-	3.94 %	4.06 %	-1.45 %	100 %
50	50	173	29	331.5 h	44 s	-	3.95 %	4.15 %	-1.60 %	100 %
20	20	74	29	163.9 h	12 s	-	3.34 %	3.60 %	-0.85 %	100 %
15	15	59	29	428.8 h	52 s	-	1.73 %	1.83 %	-0.76 %	100 %
10	10	31	29	217.9 h	26 s	-	1.16 %	1.30 %	-0.86 %	100 %
5	5	19	29	374.1 h	62 s	-	1.44 %	1.60 %	-0.64 %	100 %

Gap 1 : gap between optimal solutions and those of ILS 2, Gap 2 : gap between solutions of ILS 2 and GA 1, Gap 3 : gap between solutions of ILS 2 and GA 2 and Gap 4 : gap between solutions of ILS 2 and ILS 1.

Table 34: Computational results of iterated local search algorithm 2 on randomly generated instances of type HCT.

Job	Sub-lot	Operation	Machine	$\sum C_i$	Time	Gap 1	Gap 2	Gap 3	Gap 4	Stability
3	12	11	6	122	0.8 s	4.27 %	4.27%	4.27%	-0.81 %	100 %
3	8	13	6	88	1.2 s	3.53 %	3.53 %	3.53 %	-1.12 %	100 %
3	10	8	6	47	0.4 s	4.44 %	4.44%	4.44%	0.00 %	100 %
3	11	11	6	57	1.4 s	3.64 %	3.64%	3.64%	-1.72 %	100 %
3	12	12	6	119	4.2 s	1.71 %	1.71%	1.71%	-0.83 %	100 %
3	11	13	6	80	1.6 s	2.56 %	2.56%	2.56%	-1.23 %	100 %
3	10	12	6	91	9.2 s	3.41 %	3.41%	3.41%	-1.09 %	100 %
3	10	13	6	105	14 s	1.94 %	1.94 %	1.94 %	-0.94 %	100 %
3	10	10	6	79	2.1 s	3.95 %	3.95%	3.95%	-1.25 %	100 %
3	8	9	6	45	1.8 s	2.27 %	2.27 %	2.27 %	0.00 %	100 %

Gap 1 : gap between optimal solutions and those of ILS 2, Gap 2 : gap between solutions of ILS 2 and GA 1, Gap 3 : gap between solutions of ILS 2 and GA 2 and Gap 4 : gap between solutions of ILS 2 and ILS 1.

Table 35: Computational results of iterated local search algorithm 2 on randomly generated instances.

To evaluate the performance of iterated local search algorithms developed, these metaheuristics were tested on the same instances as the genetic algorithms. From the previous results, we observe that the performance of the iterated local search algorithms depends on the type of instances and their sizes, and it depends also on the choice of heuristics for the generation of initial solutions such as the genetic algorithms. The first algorithm gives better solutions with the SPT assignment heuristic and the MNOR sequence heuristic, while the second algorithm finds good solutions with the MMWA and MNOR heuristics. By comparing the results obtained with genetic algorithms and iterated local search methods on all the tested instances, we find that the ILSs are less good than the GAs in terms of quality of solutions obtained. Whereas, in terms of rapidity, the ILSs methods are very fast compared to the GAs.

10 Conclusion

The present work deals with the study of a new industrial problem. Different resolution methods for scheduling production processes in hospital catering were developed. A mathematical model integrating all the constraints of the studied problem was proposed. This model is an improvement of standard flexible job shop scheduling problem with sequence-dependent setup times by integrating specific industrial constraints. An extensive model study confirming the effectiveness of the proposed model is presented. The computational results of the mathematical model on different types of instances show the limits of an exact resolution for the problem of scheduling production processes. To solve the large instances of the problem, different metaheuristics have been developed and tested on several types of instances. The computational results of these metaheuristics have proven their effectiveness and reliability for scheduling operations in the food production processes and allowed significant improvements in the performance of the studied production system. As regards future research, the present work opens the way to different perspectives such as the study of the production planning problem over several days and our future works will focus on the development of resolution methods for this problem.

References

[1] R. Buddala and S. Mahapatra (2018). An integrated approach for scheduling flexible job-shop using teaching learning-based optimization method. Journal of Industrial Engineering International.

[2] L. Shen, S. Dauzere Peres and J.S. Neufeld. (2018). Solving the flexible job shop scheduling problem with sequence-dependent setup times. Journal of Operational Research.

[3] D. Behnke1 and M. J. Geiger (2012). Test Instances for the Flexible Job Shop Scheduling Problem. with Work Centers. Helmut-Schmidt-University, Logistics Management Department, Germany.

[4] A. Azzouz, M. Ennigrou and L. Ben Said (2017). A hybrid algorithm for flexible job-shop scheduling problem with setup times. International Journal of Production Management and Engineering.

[5] Tahmassebi, T. (1996). Industrial experience with a mathematical-programming based system for factory systems planning/ scheduling. Computers Chemical Engineering.

[6] Akkerman, R., and van Donk, D. P. (2009). Analyzing scheduling in the food-processing industry: structure and tasks. Cognition, Technology Work.

[7] Smith Daniels, V. L., and Larry P. Ritzman. (1988). "A Model for Lot Sizing and Sequencing in Industries." Journal of Production Research.

[8] Nakhla, M. (1995). Production control in the food ing industry. International Journal of Operations Production Management.

[9] Arbib, C., Pacciarelli, D., and Smriglio, S. (1999). A three-dimensional matching model for perishable production scheduling. Discrete Applied Mathematics.

[10] Lütke entrup, M., Günther, H.-O., Van Beek, P., Grunow, M., Seiler, T. (2005). Mixed-Integer Linear Programming approaches to shelf-life-integrated planning and scheduling in yoghurt production. Journal of Production Research.

[11] Marinelli, F., Nenni, M. E., Sforza, A. (2007). Capacitated lot sizing and scheduling with parallel machines and shared buffers: A case study in a packaging company. Annals of Operations Research.

[12] Doganis, P., Sarimveis, H. (2007). Optimal scheduling in a yogurt production line based on mixed integer linear programming. Journal of Food Engineering.

[13] Cai, X., Chen, J., Xiao, Y., and Xu, X. (2008). Product selection, machine time allocation, and scheduling decisions for manufacturing perishable products subject to a deadline. Computers Operations Research.

[14] Ahumada, Omar, Villalobos, J.Rene, 2009. Application of planning models in the agri-food supply chain: a review. European Journal of Operational Research.

[15] Kopanos, G. M., Puigjaner, L., Georgiadis, M. C. (2012). Efficient mathematical frameworks for detailed production scheduling in food ing industries. Computers Chemical Engineering.

[16] Wauters, T., Verbeeck, K., Verstraete, P., Vanden Berghe, G., De Causmaecker, P. (2012). Real-world production scheduling for the food industry: An integrated approach. Engineering Applications of Artificial Intelligence.

[17] Sel, C., Bilgen, B., Bloemhof-Ruwaard, J. M., van der Vorst, J. G. A. J. (2015). Multi-bucket optimization for integrated planning and scheduling in the perishable dairy supply chain. Computers Chemical Engineering.

[18] Frédéric Dugardin, Hicham Chehade, Lionel Amodeo, Farouk Yalaoui and Christian Prins. (2007). Hybrid Job Shop and parallel machine scheduling problems: minimization of total tardiness criterion. University of Technology of Troyes, France.

[19] Copil, K., Wörbelauer, M., Meyr, H., Tempelmeier, H. (2016). Simultaneous lotsizing and scheduling problems: a classification and review of models. OR Spectrum.

[20] Niaki, M. K., Nonino, F., Komijan, A. R., Dehghani, M. (2017). Food production in batch manufacturing systems with multiple shared-common resources: a scheduling model and its application in the yoghurt industry. International Journal of Services and Operations Management.

[21] Chen, S., Berretta, R., Clark, A., Moscato, P. (2019). Lot Sizing and Scheduling for Perishable Food Products: A Review. Reference Module in Food Science.

[22] Wei, W., Amorim, P., Guimarães, L., Almada-Lobo, B., 2018. Tackling perishability in multi- level industries. Int. J. Prod.

[23] Sargut, F.Z., Isık, G., 2017. Dynamic economic lot size model with perishable inventory and capacity constraints. Appl. Math. Model.

[24] Stefansdottir, B., Grunow, M., Akkerman, R. (2016). Classifying and modeling setups and cleanings in lot sizing and scheduling. European Journal of Operational Research.

[25] Acevedo-Ojeda, A., Contrerasa, I., Chenb, M., 2015. Two-level lot-sizing with raw-material perishability and deterioration. Int. J. Prod. Res.

[26] Bilgen, B., Çelebi, Y., 2013. Integrated production scheduling and distribution planning in dairy supply chain by hybrid modelling. Ann. Operat. Res.

[27] Garey, M.R., Johnson, D.S., and Sethi, R., 1976. The complexity of flowshop and jobshop scheduling. Mathematics of Operations Research, 1 (2), 117–129.

[28] Kacem, I., 2003. Genetic algorithm for the flexible jobshop scheduling problem. In: Proceedings of the IEEE international conference on systems, man, and cybernetics, Washington, DC, 3464–6469.

[29] Pezzella, F., Morganti, G., and Ciaschetti, G., 2008. A genetic algorithm for the Flexible Job-shop Scheduling Problem. Computers and Operations Research, 35 (10), 3202–3212.

[30] Goldberg, D., 1989. Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley, New York.

[31] Goldberg, D. E., Korb, B., and Deb, K., 1989. Messy genetic algorithms: Motivation, analysis, and first results. Complex systems, 3 (5), 493–530.

[32] Sivanandam, S. and Deepa, S., 2007. Introduction to genetic algorithms.

[33] M. Cremadez et F. Grateau (1997). Le management stratégique hospitalier, Inter éditions, Masson. Paris.

[34] Landry S. and Beaulieu M., (2000). « Étude internationale des meilleures pratiques de logistique hospitalière », Groupe de recherche CHAINE, Montréal, rapport n° 00-05, HEC Montréal, 2000. 114P.

[35] Blouin J., Beaulieu M., Landry S., (2001). « Systèmes de réapprovisionnement des unités de soins : description et implications organisationnelles » Montréal, Groupe de recherche chaîne, 2001, Cahier de recherche 01-04, 14 p.

[36] Sampiere Teissier N., (2004). « Enjeux et limites d'une amélioration des pratiques logistiques dans les hôpitaux publics français », Logistique Management 2004.

[37] Mucchielli A. (1993). « La crise interne des hôpitaux : un outil d'évaluation de son importance », Gestions hospitalières, n° 324, Mars.

[38] Cauvin, C. (1997). « Etablissements de santé : spécificité de la gestion et particularité du management », in Encyclopédie de gestion, sous la dir. deY. Simon P. Joffre, Economica, 2ème édition, pp.1179 - 1205.

[39] Naylor, C.D. (1999). «Health Care in Canada: Incrementalism under Fiscal Duress», Health Affairs, vol. 18, n° 3, p. 9-26.

[40] Landry, S. and M. Beaulieu. (2002). « Logistique hospitalière : un remède aux maux du secteur de la santé ? », Gestion, vol. 26, n° 4, p. 34-41

[41] Fixari, D. and Tonneau, D. (1993), « La modernisation par recours aux mécanismes type marché, des effets à découvrir par une évaluation – apprentissage, le cas des nouveaux outils de gestion hospitalière », Revue Politiques et Management Public, Vol. 11, n° 2, pp. 93 – 115.

[42] Ruiz Angel. (2002). «Logistique de la distribution dans le secteur de la santé». Thèse de docforai, Université de Technologie de Compiègne, France, 94P.

[43] Bonniol Vincent, Chambaretaud Sandrine, CHANEL Olivier. (1999). "Efficacité de la dépense publique en matière de santé : cohérence des instruments de régulation : rapport scientifique", Commissariat général du Plan, Paris, La Documentation française, 1999, 526 p.

[44] Gérald, (2004). «L'hôpital poursuit son évolution logistique» Techniques hospitalières, 2004 Septembre - Octobre, N° 687, P47-50.

[45] Thevenin, J. (1999). «Logistique hospitalière en France : étude de cas», Montréal, École des HEC, 188 p.

[46] Chaudhry, I. A., Khan, A. A. (2015). A research survey: review of flexible job shop scheduling techniques. International Transactions in Operational Research, 23(3), 551–591.

[47] Lourenço, HR, OC Martin and T Stutzle (2003). Iterated local search, In Handbook of Metaheuristics, F Glover and Kochenberger (Eds.), International Series in Operations Research Management Science, Vol. 57, pp. 320–353. Norwell, MA: Kluwer Academic Publishers

[48] R. Roy, Society of Manufacturing Engineers, NY, USA, 1990.

[49] K. Dehnad, Quality Control, Robust Design, and the Taguchi Method, Springer US, Boston, MASS, USA, 1989.