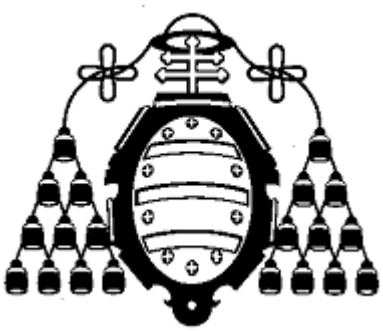
UNIVERSIDAD DE OVIEDO



MASTER IN SOFT COMPUTING AND INTELLIGENT DATA ANALYSIS

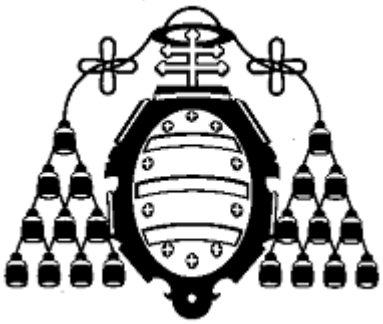
PROYECTO FIN DE MASTER
MASTER PROJECT

NEIGHBORHOOD STRUCTURES FOR SCHEDULING PROBLEMS WITH ADDITIONAL RESOURCE TYPES

RAÚL MENCÍA CASCALLANA

JULY 2012

UNIVERSIDAD DE OVIEDO



MASTER IN SOFT COMPUTING AND INTELLIGENT DATA ANALYSIS

PROYECTO FIN DE MASTER
MASTER PROJECT

NEIGHBORHOOD STRUCTURES FOR SCHEDULING PROBLEMS WITH ADDITIONAL RESOURCE TYPES

RAÚL MENCÍA CASCALLANA

JULY 2012

TUTOR / ADVISOR:
RAMIRO VARELA ARIAS
Mª CAMINO RODRÍGUEZ VELA

Abstract

The job shop scheduling is a challenging problem that has interested to researchers in the fields of Artificial Intelligence and Metaheuristics over the last decades. In this project, we face the job shop scheduling problem with an additional resource type (operators). This is a variant of the problem, which has been proposed recently in the literature. We start from a genetic algorithm that has been proposed previously to solve this problem and improve it in two different ways. Firstly, we introduce a modification in the schedule generation scheme in order to control the time of inactivity of the machines. Secondly we define a number of neighbourhood structures that are then incorporated in a memetic algorithm. In order to evaluate the proposed strategies, we have conducted an experimental study across a benchmark derived from a set of hard instances of the classic job shop problem.

Key words

metaheuristics, genetic algorithms, memetic algorithms, local search, job shop scheduling problem with operators, makespan, total flow time.

Table of contents

1. Introduction

There are many combinatorial optimization problems for which exact methods are not efficient in their resolution. That is why in recent years have proliferated evolutionary and metaheuristic techniques, which calculate approximate solutions with remarkable success [Talbi 2009]. Examples of these techniques are genetic algorithms, local search algorithms and memetic algorithms, which combine the two.

In this master project we have developed two algorithms for one of those problems: the Job Shop Scheduling Problem with Operators, in which it is required to order the execution of a set of tasks that share resources and that are assisted by human operators so that an objective function is optimized. This problem is a generalization of the Job Shop Scheduling problem, which is of great importance in Artificial Intelligence.

The two algorithms are extensions of a genetic algorithm presented in the final project [Mencía 2010] and the conference paper [Mencía et al. 2011]. The first proposal changes the fitness function used by the genetic algorithm. The second one consists in combining it with a local search algorithm; in particular, we try to extend the neighborhood structure, termed N1 in [Matfeld1995], for the classic job shop scheduling problem.

The remainder of this document is organized as follows. In section 2 we clarify the objectives of this work. In section 3, the Job shop Scheduling problem with Operators is formulated. Section 4 describes the main components of the genetic algorithm used to solve this problem. Section 5 introduces the modifications introduced in the schedule generation scheme in order to control the time of inactivity of the machines, which give rise to the so called Hybrid OG&T algorithm. Section 6 is devoted to present the neighborhood structures. In section 7, we introduce the memetic algorithms created by combining the genetic algorithm with a local search that uses these neighborhood structures. The design and results of the experimental study are given in section 8. In section 9 we summarize the main conclusions and propose some ideas for future research. Finally, we include a appendix containing detailed results.

2. Objectives

The main goal of this master thesis is to improve the genetic algorithm proposed in [Mencía et al. 2011]. In particular we propose the following specific objectives:

Modify the schedule generation schema, termed OG&T, adapted to decode chromosomes in the genetic algorithm, so as the maximum time of inactivity of the machines can be limited. These modifications give rise to the so called Hybrid OG&T algorithm. In this way we can reduce the search to a subset of solutions that are expected to be better in average than the solutions of the whole search space.

Define neighbourhood structures for the job shop scheduling problem with operators and makespan minimization. In particular we propose extending the well-know structure, termed N1, defined for the classic job shop scheduling problem. These structures will then be incorporated in the genetic algorithm and it is expected that the resulting memetic algorithm reaches better solutions than the original genetic algorithm.

Evaluate the proposed strategies by means of an experimental study. In particular, we try to establish the conditions in which the proposed extensions may outperform the original genetic algorithm.

3. The Job Shop Scheduling Problem with Operators

3.1. Problem formulation

Formally the job-shop scheduling problem with operators can be defined as follows. We are given a set of n jobs $\{J_1, \dots, J_n\}$ a set of m resources or machines $\{R_1, \dots, R_m\}$ and a set of p operators $\{O_1, \dots, O_p\}$. Each job J_i consists of a sequence of v_i operations or tasks $(\theta_{i1}, \dots, \theta_{iv_i})$. Each task θ_{il} has a single resource requirement $R_{\theta_{il}}$, an integer duration $p_{\theta_{il}}$ and a start time $st_{\theta_{il}}$ and an assisting operator $O_{\theta_{il}}$ to be determined.

A feasible schedule is a complete assignment of starting times and operators to operations that satisfies the following constraints:

- The operations of each job are sequentially scheduled.
- Each machine can process at most one operation at any time.
- No preemption is allowed.
- Each operation is assisted by one operator and one operator cannot assist more than one operation at the same time.

The objective is finding a feasible schedule that minimizes some objective function. This problem was first defined in [Agnetis et al. 2011] for makespan minimization and is denoted as $JSO(n,p)$. In this work, we consider two objective functions: the makespan, which is the completion time of all the operations, and the total flow time, obtained by summing up the completion time of all the jobs. The second may be more interesting from a practical point of view in a number of domains, as manufacturing, and usually makes the problem harder to solve [González et al. 2010] [Brucker and Knust 2006].

The significant cases of this problem are those with $p < min(n,m)$, otherwise the problem is a standard job-shop problem.

Figure 1 shows a Gantt chart for a feasible schedule to an instance with 3 jobs, 3 machines and 2 operators. The operations are labeled with the required resource and the assisting operator. It can be easily seen that all the constraints of the problem are satisfied and, as there are only two operators available, no more than two operations are ever processed in parallel. The makespan of the schedule is 14, and its total flow time is 14+13+10=37.

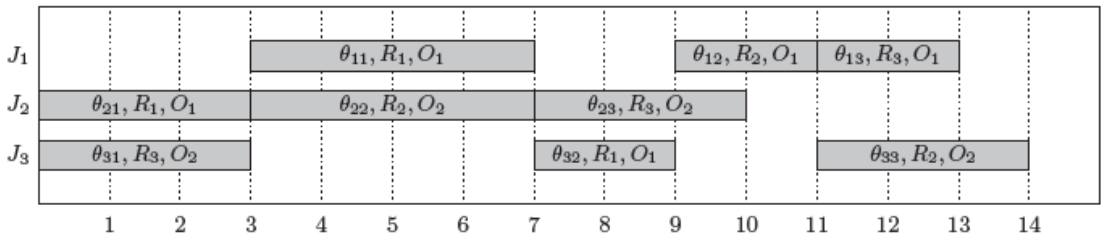


Figure 1: A Gantt chart for a feasible schedule to a problem with 3 jobs, 3 machines and 2 operators.

3.2. A disjunctive graph model

Scheduling problems are usually represented by means of a disjunctive model, which was first proposed in [Roy and Sussman 1964]. This kind of modeling allows to solve the problem by deciding about the relative order among operations that share the same resources, instead of considering for every operation all its possible starting times. We propose here to use the following model for the $JSO(n,p)$ that is similar to that used in [Agnetis et al. 2011]. A problem instance is represented by a directed graph $G = (V, C \cup D \cup I \cup O)$. Each node in the set V represents either an actual operation, or any of the fictitious operations with null processing time introduced with the purpose of giving the graph a particular structure. These fictitious operations include starting and finishing operations for each operator i, denoted O_i^{start} and O_i^{end} respectively, and the dummy operations $start$ and end.

The arcs in C are called $conjunctive\ arcs$ and represent precedence constraints among operations of the same job. The arcs in D are called $disjunctive\ arcs$ and represent capacity constraints. D is partitioned into subsets D_i with $D = \cup_{\{i=1,\dots,M\}} D_i$. D_i includes an arc (v,w) for each pair of operations requiring the resource R_i. The set O of $operator\ arcs$ includes three types of arcs: one arc (u,v) for each pair of operations of the problem, and arcs (O_i^{start}, u) and (u, O_i^{end}) for each operator node and operation. The set I includes arcs connecting node $start$ to each node O_i^{start} and arcs connecting each node O_i^{end} to node end. The arcs are weighted with the processing time of the operation at the source node.

From this representation, building a solution can be viewed as a process of fixing disjunctive and operator arcs. A disjunctive arc between operations u and v gets fixed when one of (u,v) or (v,u) is selected and consequently the other one is discarded. An operator arc between u and v is fixed when (u,v), (v,u) or none of them is selected, and fixing the arc (O_i^{start}, u) means discarding (O_i^{start}, v) for any operation v other than u. Analogously for (u, O_i^{end}).

Therefore, a feasible schedule S is represented by an acyclic subgraph of G, of the form $G_S = (V, C \cup F \cup I \cup Q)$, where F expresses the processing order of operations on the machines and Q expresses the sequences of operations that are assisted by each operator. In other words, $F = \cup_{i=1,\dots,m} F_i$, F_i being a subset of D_i that includes the arc (u,v) iff u is processed before v in S. $Q = \cup_{i=1,\dots,p} Q_i$, where Q_i is a subset of Q that represents the order in which the k_i operations assisted by the operator O_i are processed. So, it includes the arcs (O_i^{start}, u_1^i), $(u_{k_i}^i, O_i^{end})$ and (u,v) for each pair of operations assisted by the operator O_i such that u is processed before v. For each operation u, at least two arcs of the form $(u,-)$ and $(-,u)$ are in some Q_i and if one of

these arcs is in Q_i none of the remaining arcs involving u belongs to Q_j, $i \neq j$. The makespan is the cost of a $critical\ path$ in G_S. A critical path is a longest cost path from node $start$ to node end. Analogously, the total flow time is computed as the sum of the longest cost paths in G_S through the last operation of each job.

Figure 2 shows a solution graph for the schedule represented in Figure 1. Discontinuous arrows represent operator arcs. So, the sequences of operations assisted by operators O_1 and O_2 are $(\theta_{21}, \theta_{11}, \theta_{32}, \theta_{12}, \theta_{13})$ and $(\theta_{31}, \theta_{22}, \theta_{23}, \theta_{33})$ respectively. In order to simplify the picture, only disjunctive and operator arcs between consecutive operations in a machine or an operator respectively are shown. Continuous arrows represent conjunctive arcs and doted arrows represent disjunctive arcs; in these cases only arcs not overlapping with operator arcs are drawn. In this example, a critical path is given by the sequence $(\theta_{21}, \theta_{11}, \theta_{32}, \theta_{12}, \theta_{33})$, so the makespan is the cost of this path, that is, 14.

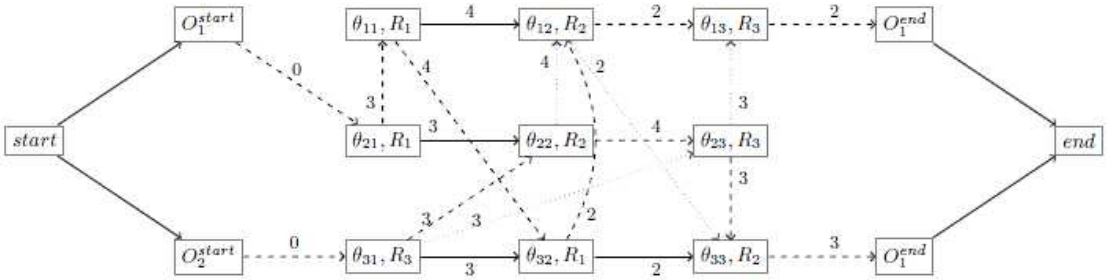


Figure 2: A disjunctive graph representing the schedule of Figure 1.

In order to simplify expressions, we define the following notation for a feasible schedule. The head r_v of an operation v is the cost of the longest path from node $start$ to node v, i.e. it is the value of st_v. The tail q_v is defined so as the value $q_v + p_v$ is the cost of the longest path from v to end. Hence, $r_v + p_v + q_v$ is the makespan if v is in a critical path, otherwise, it is a lower bound. PM_v and SM_v denote the immediate predecessor and successor of v respectively on the machine sequence, PJ_v and SJ_v denote the immediate predecessor and successor operations of v respectively on the job sequence and PO_v and SO_v denote the immediate predecessor and successor operations of v respectively on the operator of v.

A partial schedule is given by a subgraph of G where some of the disjunctive and operator arcs are not fixed yet. In such a schedule, heads and tails can be estimated as

$$r_v = max\left\{\max_{J\subseteq P(v)}\left\{\min_{j\in J} r_j + \sum_{j\in J} p_j\right\}, \max_{J\subseteq PO(v)}\left\{\min_{j\in J} r_j + \sum_{j\in J} p_j\right\}, r_{PJ_v} + p_{PJ_v}\right\}$$

$$q_v = max\left\{\max_{J\subseteq S(v)}\left\{\sum_{j\in J} p_j + \min_{j\in J} q_j\right\}, \max_{J\subseteq SO(v)}\left\{\sum_{j\in J} p_j + \min_{j\in J} q_j\right\}, p_{SJ_v} + q_{SJ_v}\right\}$$

with $r_{start} = q_{end} = r_{O_i^{start}} = q_{O_i^{end}} = 0$ and where $P(v)$ denotes the disjunctive predecessors of v, so as for all $w \in P(v)$, $R_w = R_v$ and the disjunctive arc (w, v) is already fixed (analogously, $S(v)$ denotes the disjunctive successors of v). $PO(v)$ denotes the operator predecessors of v, i.e $w \in PO(v)$ if it is already established that $O_w = O_v$ and w is processed before v, so as the operator arc (w, v) is fixed (analogously, $SO(v)$ are the operator successors of v).

4. Genetic Algorithm for the $JSO(n,p)$

The GA used here is taken from [Mencía et al. 2011]. The encoding schema is based on permutations with repetition as it was proposed in [Bierwirth 1992]. A chromosome is a permutation of the set of operations that represents a tentative ordering to schedule them, each one being represented by its job number. For example, the sequence (2 1 1 3 2 3 1 2 3) is a valid chromosome for a problem with 3 jobs and 3 machines. As it was demonstrated in [Mattfeld 1995], this encoding has a number of interesting characteristics for the classic job shop scheduling problem; for example, it tends to represent orders of operations as they appear in good solutions. So, it is expected that these characteristics are to be good for the $JSO(n,p)$ as well.

For chromosome mating, the GA uses the Job Order Crossover (JOX) described in [Bierwirth 1992]. Given two parents, JOX selects a random subset of jobs and copies their genes to the offspring in the same positions as they are in the first parent, then the remaining genes are taken from the second parent so as they maintain their relative ordering. We clarify how JOX works by means of an example. Let us consider the following two parents

Parent1 (**2** 1 1 3 **2** 3 1 **2** 3) Parent2 (3 3 1 2 1 3 2 2 1)

If the selected subset of jobs from the first parent just includes the job 2, the generated offspring is

Offspring (**2** 3 3 1 **2** 1 3 **2** 1).

Hence, operator JOX maintains for each machine a subsequence of operations in the same order as they are in parent 1 and the remaining in the same order as they are in parent 2.

To evaluate chromosomes, the GA uses the algorithm $OG\&T$ described in [Sierra et al. 2011], [Mencía 2010] and [Mencía et al. 2011], which is an extension of the original *G&T* proposed in [Giffler and Thompson 1960], so that the non-deterministic choice is done by looking at the chromosome: the operation in B which is in the leftmost position in the chromosome sequence is selected to be scheduled next. We explain this algorithm in Section 5, together with an extension to it.

The remaining elements of GA are rather conventional. To create a new generation, all chromosomes from the current one are organized into couples which are mated and then mutated to produce two offspring in accordance with the crossover and mutation probabilities respectively. Finally, tournament replacement among every couple of parents and their offspring is done to obtain the next generation.

5. Hybrid OG&T

In this work we extend an algorithm to generate schedules for the $JSO(n,p)$ problem. In this section, we explain the algorithm and its new extension.

5.1. *OG&T* schedule generation scheme

We present a schedule generation scheme termed $OG\&T$ that was proposed by [Sierra et al 2011]. This is a schedule generation scheme for the $JSO(n,p)$ which is an extension of the well-known $G\&T$ algorithm proposed in [Giffler and Thompson 1960] for the classic job shop scheduling problem. The operations are scheduled one at a time following a sequence of non deterministic choices. When an operation u is scheduled, its preceding operation in the job sequence, denoted PJ_u, was already scheduled if this operation exists. At this time, u is assigned a starting time st_u and an operator $O_i, 1 \leq i \leq p$.

In order to select the candidate operations to be scheduled next, we start considering a naive strategy to establish an initial set of candidates which is subsequently restricted by means of some local pruning rules. Let SC be the set of scheduled operations at an arbitrary time. Then, the next non deterministic choice may be any operation of the set A defined as

$$A = \{v \notin SC, \nexists PJ_v \vee (PJ_v \in SC)\}$$

i.e., the set that includes the first unscheduled operation of each job that has at least one unscheduled operation. If the operation u in A is selected, the starting time of u is given by its head r_u which is calculated as

$$r_u = \max\{r_{PJ_u} + r_{PJ_u}, r_v + p_v, \min_{1 \leq i \leq p} t_i\}$$

where $t_i, 1 \leq i \leq p$, is the time at which the operator O_i is available and v denotes the last operation scheduled having $R_v = R_u$. At the same time, the operator O_i that is available at the latest time before r_u, i.e.

$$i = \arg\max\{t_j; t_j \leq r_u; 1 \leq j \leq p\}$$

is assigned to assist the operation u. Let v^* be the operation in A having the earliest completion time if it were scheduled next, i.e.

$$v^* = \arg\min\{r_u + p_u; u \in A\}.$$

The set of non-deterministic choices may be reduced to the subset $A' \subseteq A$

$$A' = \{u \in A; r_u < r_{v^*} + p_{v^*}\}$$

Moreover, the set of choices can be further restricted in the following way. Let $\tau_0 < \cdots < \tau_k$ be the sequence of all times along the interval $[\min\{r_u; u \in A'\}, r_{v^*} + p_{v^*})$, where each τ_i is given by the head of some operation in A' or the time at which some operator becomes available. Let p_i'

be the number of operators available in the subinterval $[\tau_i, \tau_{i+1})$ and let $m_i^{'}$ be the number of different machines that are required by the operations in $A^{'}$ which may be processed along this subinterval. Then, $A^{'}$ may be reduced as long as the following operations are maintained:

i. The operations requiring the same machine as v^*.
ii. For each interval $[\tau_i, \tau_{i+1})$ with $m_i^{'} > p_i^{'}$, the operations required by at least $m_i^{'} - p_i^{'}$ machines.

The set of operations obtained in this way is termed B and it is clear that $|B| \leq |A^{'}| \leq |A|$. An important property of this schedule generation scheme is that if the number of operators is large enough, in particular if $p \geq min(n, m)$ so as $JSO(n, p)$ becomes $J||\sum C_i$, it is equivalent to the $G\&T$ algorithm. Sierra *et al.* (2011b) give a full description of $OG\&T$ together with a formal proof of its dominance property, i.e. the search space contains at least one optimal solution.

So, in accordance with the above, a state is a partial schedule. In the initial state all operations are unscheduled and the remaining states correspond to each one of the situations that may be generated by the algorithm $OG\&T$ after one of the operations of the set B is scheduled. Figure 3 shows a partial schedule that represents a feasible state to a problem with 5 jobs, 5 machines and 3 operators. In accordance with the $OG\&T$, in this situation $A = \{\theta_{13}, \theta_{22}, \theta_{32}, \theta_{42}, \theta_{51}\}$, $A^{'} = \{\theta_{22}, \theta_{32}, \theta_{42}, \theta_{51}\}$ and $B = \{\theta_{32}, \theta_{42}, \theta_{51}\}$ (if R_2 and R_1 are chosen from the interval $[\tau_1, C)$). Therefore, this state has 3 successors as a result of scheduling the operations θ_{32}, θ_{42} and θ_{51} respectively. For a state n, $g(n)$ is the total flow time of the partial schedule, i.e. the summation of the completion times of the last operation scheduled in each job, and consequently the cost $c(n, n^{'})$ from n to a successor $n^{'}$ is the variation of this value from n to $n^{'}$. So, for the state n in Figure 3, $g(n) = 19$ and if $n^{'}$ is the result of scheduling θ_{32} from n, then $c(n, n^{'}) = 7$. The goals are those states having all operations scheduled.

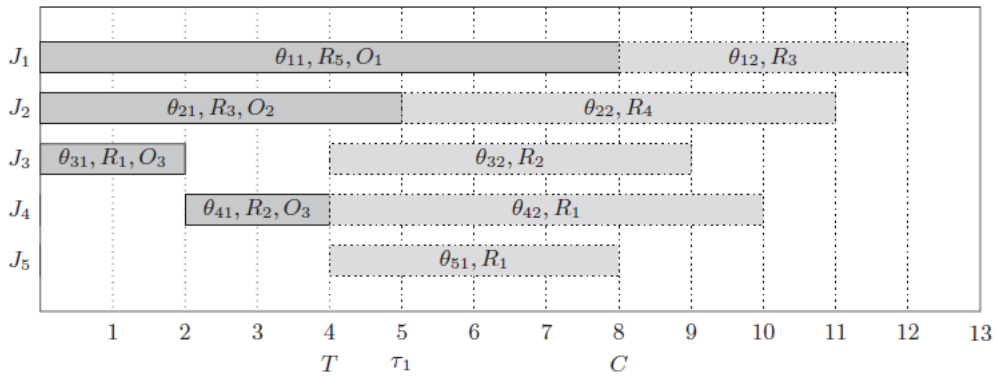


Figure 3. A partial schedule to a problem with 5 jobs, 5 machines and 3 operators.

5.2. Extending the $OG\&T$ algorithm

In its original formulation, the $G\&T$ algorithm was shown to be able to generate all the possible active schedules for the classic job shop problem. In these schedules, at least one operation has to be delayed in order to start the processing of another operation earlier, i.e., there are no machines

idle along a whole period where an operation could be completely processed. Active schedules constitute a subset of the feasible schedules that contains at least one optimal solution.

The concept of active schedule is not yet formalized for the $JSO(n,p)$ problem, and it is not trivial due to the limited number of assisting operators as a new resource type. Nevertheless, the reduction made by the $OG\&T$ algorithm from the set A to A', taking only the operations that can start before the earliest possible completion time of v^*, guarantees that no operation can be processed earlier in a schedule without delaying at least other operation.

Although the set of schedules defined by the $OG\&T$ algorithm is smaller than the set of the feasible schedules, it may be still very large for large instances. This way, we propose extending this algorithm in order to achieve further reductions, yet loosing the possibility of finding an optimal solution.

For doing so, we introduce a real parameter, termed delta, which varies from 0 to 1. Then, we reduce B to B', by defining

$$B' = \{u \in B; r_u < \min\{r_w; w \in B\} + delta * ((r_{v^*} + p_{v^*}) - (\min\{r_w; w \in B\}))\}$$

So, depending on the value of the delta parameter, the considered search space would be different.

- If delta is 1, the interval of release times of the operations considered does not change, which means that the search space is the same.
- In the rest of the cases, this interval changes by moving its superior limit to the left, so limiting the maximum time machines can be idle while there is some operation that can start its processing. Hence, the search space is reduced and it might eventually not contain an optimal solution to the problem. The greatest reduction of the search space is achieved with delta = 0, where no idle times for the machines are allowed. These schedules are termed *dense* or *non-delay* in the classic job shop scheduling problem.

It is expected that giving delta a value lower than 1 may be worthwhile if the problem instance is large. It is our hypothesis that the larger the instance, the lower the delta parameter should be.

6. Neighborhood structures for the JSO

This section describes a neighborhood structure for the problem JSO based on one of the structures used in the classic Job Shop Scheduling problem called N_1. What we do here is to extend the structures proposed in [Mencía 2012] so as the operators, and not only the machines, are also taken into account.

To improve clarity, firstly we explain the structure and then how we adapt it to the JSO.

6.1. Neighborhood structure N_1 for $JSS(n)$

To understand the neighborhood structure N_1 it is necessary to introduce the concept of critical block, which is defined as a maximal sequence of operations that need the same resource in a critical path of a schedule.

Given a schedule for the problem, the structure N_1 is based in reversing the order of operations that belong to the same critical block, trying to avoid moves that won't yield an improvement.

The following lemmas establish the base of this neighborhood structure. Their proofs can be found in [Mattfeld 1995].

Lemma 3.1.1. Reversing one critical arc in G_S cannot lead to a cycle and therefore cannot result in an infeasible solution.

Lemma 3.1.2. If the reversal of a non-critical arc in S leads to a feasible solution S', then $f(S') \geq f(S)$ holds.

Lemma 3.1.3. The reversal of a critical arc (v, w) can only lead to an improvement if at least one of PM_v and SM_w is non-critical.

Lemma 3.1.4. Let v and w be the first two successive operations of the first block. Reversing the critical arc (v, w) cannot lead to a makespan improvement. Analogous let v and w be the last two successive operations of the last block. Again, no improvement can be gained by reversing v, w.

Definition (N_1). Given S, the neighborhood $N_1(S)$ consists of all schedules derived from S by reversing one arc (v, w) of a critical path in G_S. At least one of v and w is either the first or the last member of a block. For the first block only v and w at the end of the block are considered whereas for the last block only v and w at the beginning of the block must be checked.

6.2. Adaptation of N_1 for $JSO(n,p)$

In this work we design neighborhood structures based in the reversal of one arc in the critical path.

In order to do that, we first explain the types of arcs that there are in the solution graph and we give a number of lemmas on which the neighborhood structures are based.

6.2.1. Types of arcs

(v,w) is a conjunctive arc or an arc of type J if $J_v = J_w$, $O_v \neq O_w$, $M_v \neq M_w$.

(v,w) is an arc of machine or an arc of type M if $M_v = M_w$, $O_v \neq O_w$, $J_v \neq J_w$.

(v,w) is an arc or operator or an arc of type O if $O_v = O_w$, $J_v \neq J_w$, $M_v \neq M_w$.

(v,w) is an arc of job and operator or an arc of type JO if $J_v = J_w$, $O_v = O_w$, $M_v \neq M_w$.

(v,w) is an arc of machines and operator or an arc of type MO if $M_v = M_w$, $O_v = O_w$, $J_v \neq J_w$.

The arcs that can be reversed are those of type MO, M, and O. Arcs of type J, and JO cannot be reversed because that would imply changing the order of two operations in a jobs which would lead to an infeasible solution.

In the following section we explain how to reverse the arcs in a schedule.

Reversal of arc of type MO

In Figure 4 we can see a part of a schedule were (v,w) is of type MO. In the graphs, an arc of type MO is represented as a pair of arcs, one of type M and the other of type O. The arcs of type J are represented as black continuous arrows, the arcs of type M are represented as blue discontinuous arrows and the arcs of type O are represented as red dotted arrows.

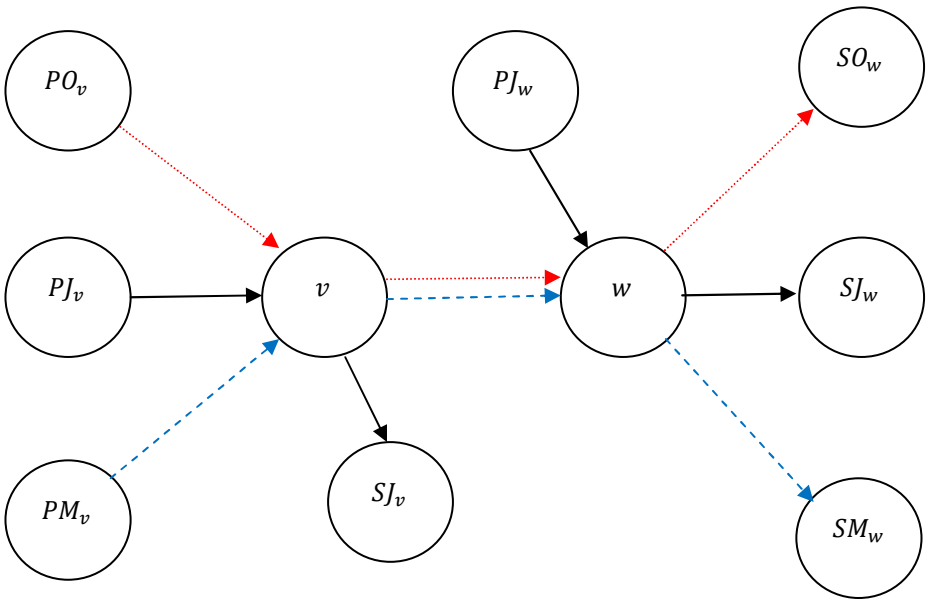


Figure 4. Graph before the arc of type MO reversal.

As can be seen in Figure 5, apart from reversing the MO arc, we have to exchange the remaining predecessors and successors of v and w. For example, the predecessor of v in the operator before the reversal is the predecessor of w after the reversal.

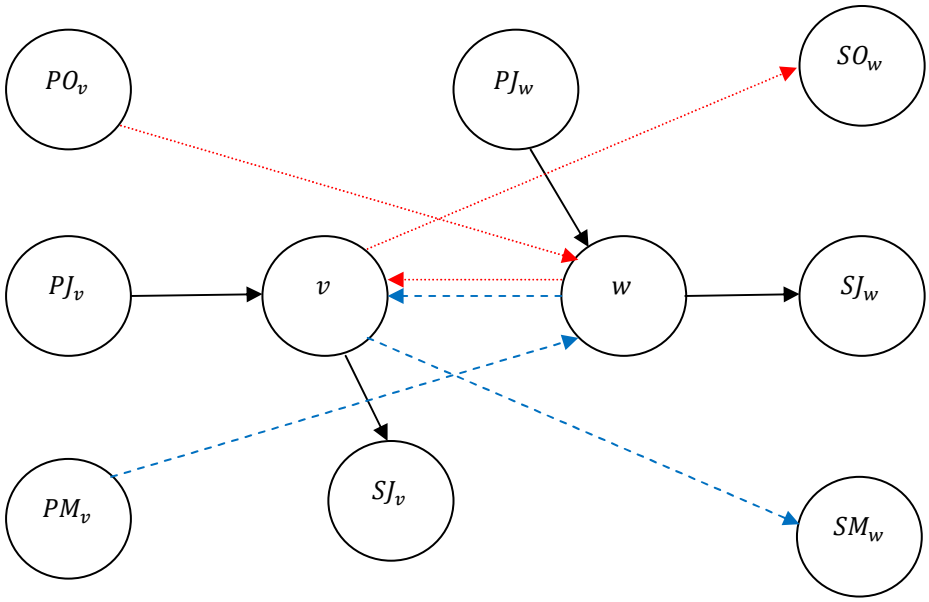


Figure 5. Graph after the arc of type MO reversal.

Reversal of arc of type M

In Figure 6 we can see a part of an schedule were (v, w) is of type M.

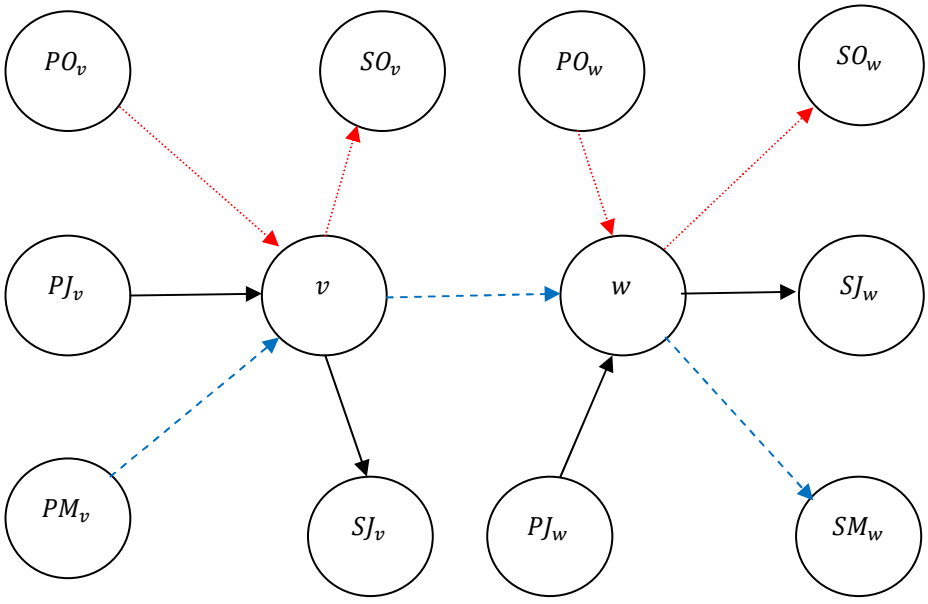


Figure 6. Graph before the arc of type M reversal.

Figure 7 shows the result of reversing (v, w).

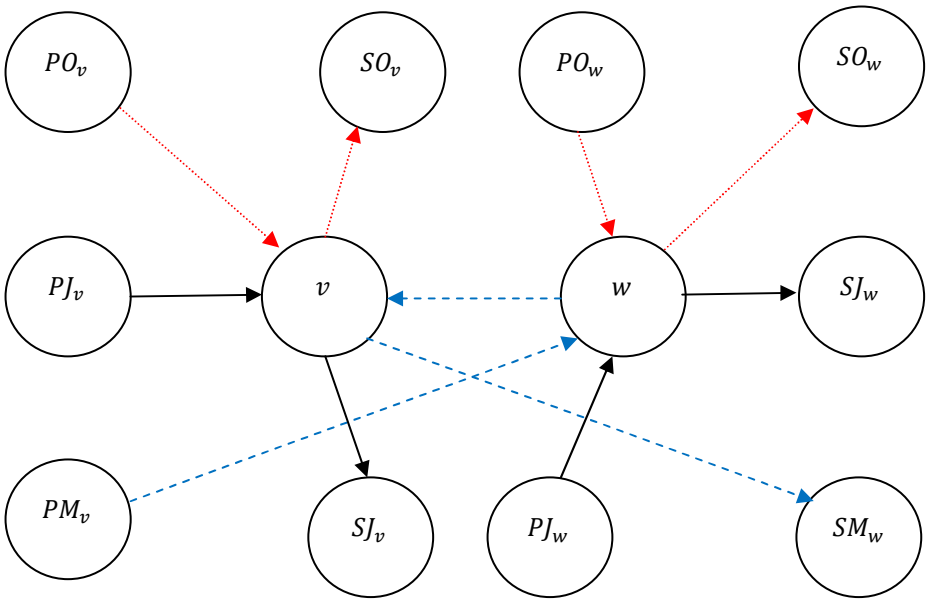


Figure 7. Graph after the arc of type M reversal.

Reversal of arc of type O

In Figure 8 we can see a part of an schedule were (v, w) is of type O.

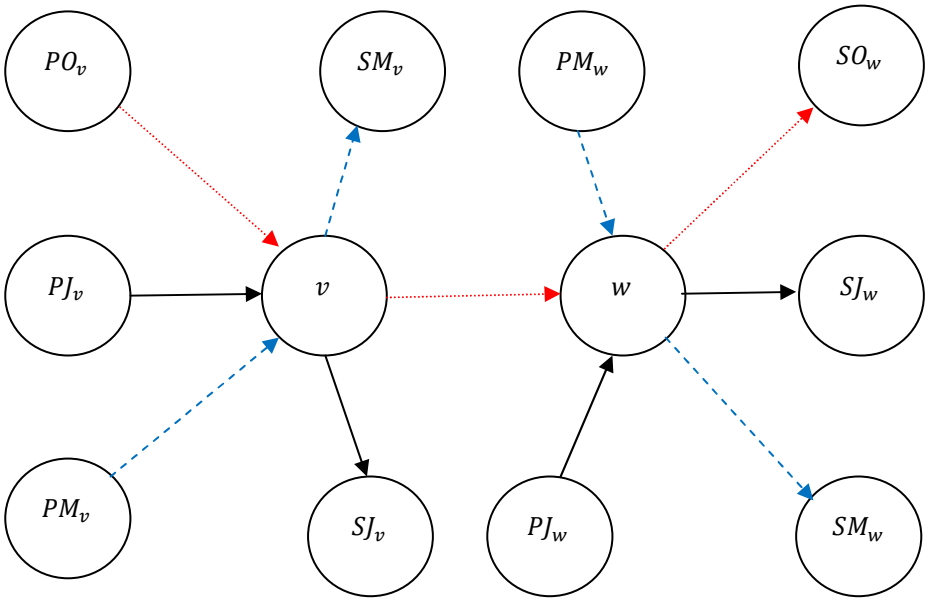


Figure 8. Graph before the arc of type O reversal.

Figure 9 shows the result of reversing (v, w).

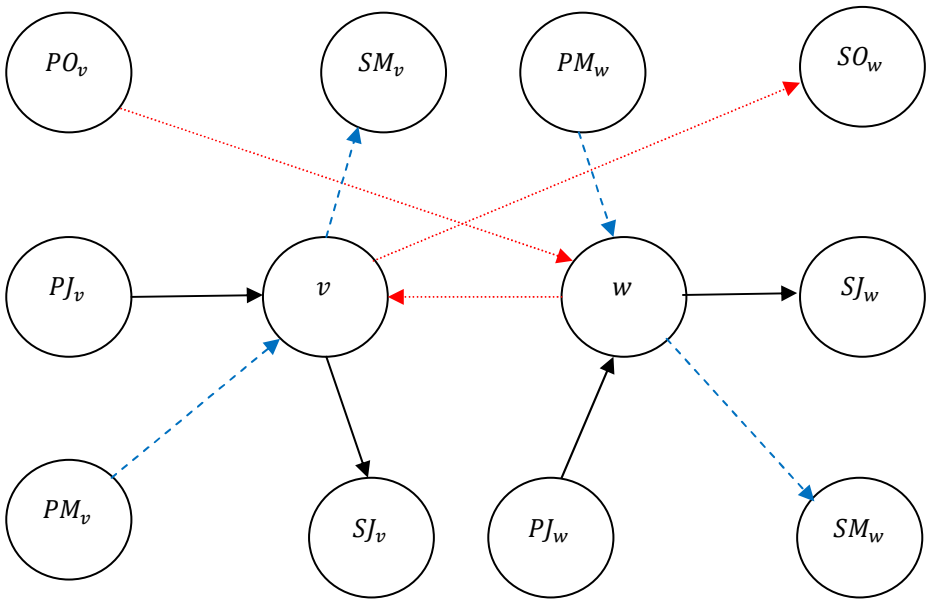


Figure 9. Graph after the arc of type O reversal.

6.2.2. Important concepts

In this section we start introducing some notions which are required to understand the proposed neighborhood structures.

Definition (Critical path): A maximum cost path from node start to node end.

Definition (Critical block): Maximal subsequence of operations of a critical path that use

- the same machines (machines critical block), or
- the same operator (operator's critical block).

Note that the arcs of a machines critical block are of type M or MO, and the arcs of an operator's critical block are of type O, MO or JO. Arcs of type JO cannot be reversed.

In order to clarify how these structures work, two illustrative examples are added.

Example of machines critical blocks:

Figure 10 shows an abstraction of a critical path of a schedule. Each circle represents an operation and it's indicated in which resource (machine) is processed.

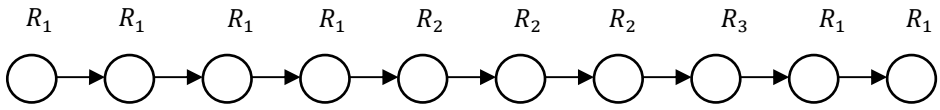


Figure 10: Critical path of a schedule.

As we can see, in the critical path there are 3 resources (R_1, R_2 and R_3).

To generate neighboring schedules, at first we need to find the existing critical blocks in a critical path. So, as it has been explained before, we take the maximal subsequences of operations that use the same resource. In Figure 11 we can see a critical path with four critical blocks.

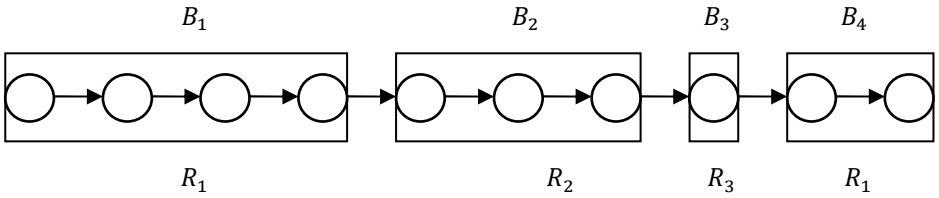


Figure 11: Machines critical blocks.

The first block consists of 4 operations that use the resource R_1. The second block consists of 3 tasks that use the resource R_2. The third block consists of 1 operation that uses the resource R_3. And, the fourth block consists of 2 operations that use the resource R_1.

Example of operator's critical blocks:

Figure 12 shows an abstraction of a critical path of a schedule. Each circle represents an operation and it is indicated which operator processes it.

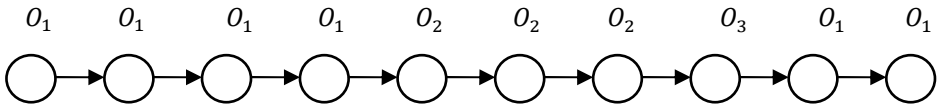


Figure 12: Critical path of a schedule.

As we can see, in the critical path there are 3 operators (O_1, O_2 and O_3).

To generate neighboring schedules, at first we need to find the existing critical blocks in a critical path. So, as it has been explained before, we take the maximal subsequences of operations that use the same operator. In Figure 13 we can see a critical path with four operator's critical blocks.

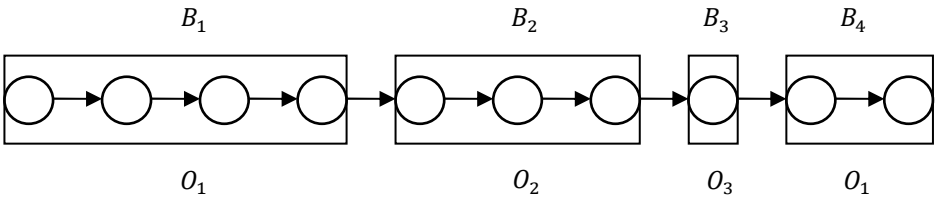


Figure 13: Operator's critical blocks.

The first block consists of 4 operations that use the operator O_1. The second block consists of 3 tasks that use the operator O_2. The third block consists of 1 operation that uses the operator O_3. And, the fourth block consists of 2 operations that use the operator O_1.

Definition (Critical arc): An arc that is in the critical path. It can be of any type (J, JO, M, O, MO).

Lemma: If an arc (v, w) of G_S is critical, then the only path from v to w in G_S is the arc (v, w).

Proof: Any other path from v to w has at least two arcs (v, x) and (y, w). The arc (v, x) has the same cost as the arc (v, w), thus the cost of this path would be larger than the cost of the arc (v, w) and in that case (v, w) would not be in the critical path.

Corollary: If (v, w) of type M, O, or MO is critical in G_S, then the $G_S{}'$ which is obtained from G_S by reversing (v, w) does not have cycles and thus represents a feasible solution.

Lemma: If an arc (v, w) that is not critical in G_S is reversed, then the resulting graph $G_S{}'$, in the cases where it is not acyclic, represents a solution which is not better than that of G_S.

Proof: The critical path of G_S remains in $G_S{}'$ after reversing the non-critical arc.

Lemma: If a critical arc (v, w) is reversed in G_S, and (v, w) is internal to a mahines critical block and internal to an operator's critical block, the resulting solution $G_S{}'$ is not better than G_S.

Proof: If (PMO_v, v, w, SMO_w) are successive operations on a critical path, reversing (v, w) does not change the starting time r_{SMO_w} as $r_{PMO_v} + p_v + p_w = r_{SMO_w}$. Therefore this reversal cannot lead to an improvement.

Lemma: If (v, w) is the first arc of the first critical block, or the last arc of the last critical block, then the reversal of (v, w) cannot produce any improvement (unless the block has only two operations).

Proof: What makes this case different to the other is that the predecessor of the operation v is the node start, which has null duration. This means that the head of the successor of w would stay the same $(p_v + p_w)$ and the critical path would be the same since the successor of w being in the critical path means that there is no other path linking w and that successor of higher cost, thus there is not any improvement.

6.2.3. Definition of neighborhood structures

According to the results explained before, the only options to improve, with a simple reversal of an arc, is to reverse arcs (O, M, or MO) that are in the borders of a critical block, with the

exception of the first arc of the first block and the last arc of the last block in case that his block have only two tasks.

Firstly, we will define three neighborhood structures, N_1^1, N_1^2 and N_1^3. After that we will define the structure N_1^O which is the union of them.

Definition (N_1^1). Given an schedule S, the neighborhood $N_1^1(S)$ is defined as the set of schedules obtained from G_S by reversing an arc (v, w) such that

i. (v, w) is critical
ii. (v, w) is of type MO
iii. (v, w) is neither in the interior of a machine critical block nor in the interior of an operator's critical block
iv. (v, w) is not the first arc of the critical path being the length of the critical block it belongs to larger than 2
v. (v, w) is not the last arc of the critical path being the length of the critical block it belongs to larger than 2.

Definition (N_1^2): Given an schedule S, the neighborhood $N_1^2(S)$ is defined as the set of schedules obtained from G_S by reversing an arc (v, w) such that

i. (v, w) is critical
ii. (v, w) is of type O or MO
iii. If (v, w) is of type O, it is not in the interior of an operator's critical block
iv. If (v, w) is of type MO, it is not in the interior of an operator's critical block, and it is in the interior of a machines critical block
v. (v, w) is not in the interior of an operator's critical block
vi. (v, w) is not the first arc of the critical path being the length of the first operator's critical block larger than 2
vii. (v, w) is not the last arc of the critical path being the length of the last operator's critical block larger than 2.

Definition (N_1^3): Given an schedule S, the neighborhood $N_1^3(S)$ is defined as the set of schedules obtained from G_S by reversing an arc (v, w) such that

i. (v, w) is critical
ii. (v, w) is of type M or MO
iii. If (v, w) is of type M, it is not in the interior of an machines critical block
iv. If (v, w) is of type MO, it is not in the interior of an machines critical block, and it is in the interior of a operator's critical block
v. (v, w) is not the first arc of the critical path being the length of the first machines critical block to larger than 2
vi. (v, w) is not the last arc of the critical path being the length of the last machines critical block to larger than 2.

Definition (N_1^O): Given an schedule S, the neighborhood $N_1^O(S)$ is defined as the set of schedules obtained from G_S by reversing an arc (v, w) such that

i. (v, w) is critical
ii. (v, w) is of type O, M or MO
iii. (v, w) is not at the same time in the interior of a machine critical block and in the interior of an operator's critical block
iv. (v, w) is not the first arc of the critical path being the length first critical block that it belongs to larger than 2
v. (v, w) is not the last arc of the critical path being the length of the critical block it belongs to larger than 2.

Notice that $N_1^O = N_1^1 \cup N_1^2 \cup N_1^3$

Note that the neighborhood structures N_1^2 and N_1^3 could be divided into two separate neighborhood structures, one for each type of arc. Firstly, the neighborhood structure N_1^2, it can be divided into one structure for the arcs of type O, and the other one for the arcs of type MO that it contains. Then, the neighborhood structure N_1^3 can be divided into one structure for the arcs of type M and another structure for the arcs of type MO that it contains.

We clarify by means of an example how these neighborhood structures work.

Figure 14 shows an abstraction of a critical path of a schedule. Each circle represents an operation and it is indicated which operator processes it and on which resource (machine) it is processed.

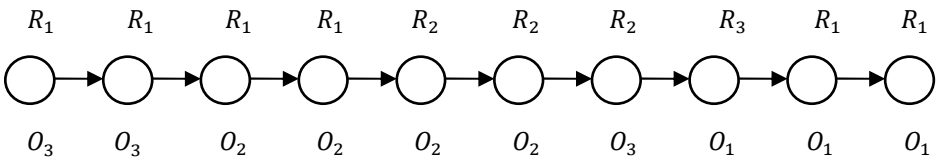


Figure 14. Critical path of a schedule.

As we can see, in the critical path there are 3 machines (R_1, R_2 and R_3) and 3 operators (O_1, O_2 and O_3).

To generate neighboring schedules, we have to indentify all the existing critical blocks in a critical path. In Figure 15 we can see a critical path with four machines critical blocks and another four operator's critical blocks. The machines critical blocks are represented with blue boxes and the operator's critical blocks are represented with red boxes.

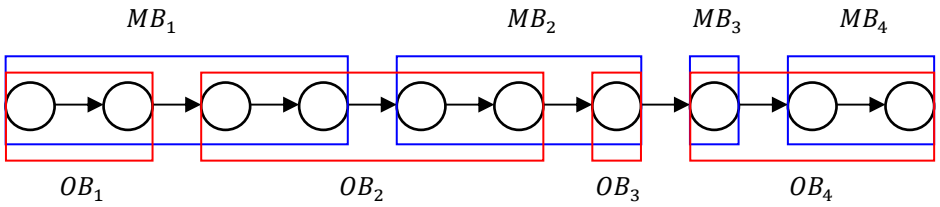


Figure 15. Critical blocks.

The first machines block consists of 4 operations that use the machine R_1. The second machines block consists of 3 tasks that use the machine R_2. The third machines block consists of 1 operation that uses the machine R_3. And, the fourth machines block consists of 2 operations that use the machine R_1.

The first operators block consists of 2 operations that use the operator O_1. The second operators block consists of 4 tasks that use the operator O_2. The third operators block consists of 1 operation that uses the operator O_3. And, the fourth operators block consists of 3 operations that use the operator O_1.

As explained before, N_1^1 generates neighbors reversing arcs that are both at the border of a machines critical blocks and at the border of an operator's critical blocks (with the exception of the first arc of the first block and the last arc of the last block in the case in which these blocks have more than 2 operations). N_1^2 generates neighbors reversing arcs that are at the border of an operator's critical block and are not at the border of a machines critical block (with the exception of the first arc of the first block and the last arc of the last block in the case in which these blocks have more than 2 operations). N_1^3 generates neighbors reversing arcs that are at the border of a machines critical block and are not at the border of an operator's critical block (with the exception of the first arc of the first block and the last arc of the last block in the case in which these blocks have more than 2 operations). N_1^O is the union of all the previous structures.

Figure 16 shows the arcs painted in colors depending which neighborhood structure would reverse them. Also, it shows the type of the candidate arcs to be reversed. Note that the arcs of type O could be of type JO as well, but let us imagine that they are of type O.

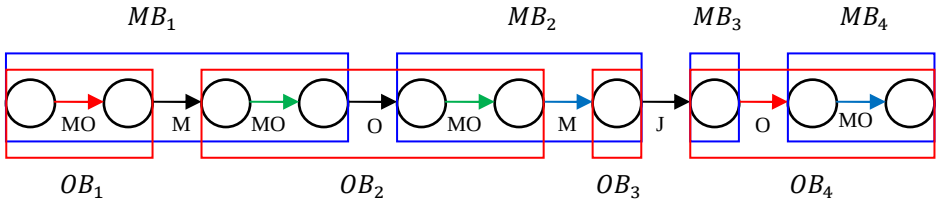


Figure 16. Critical arcs to reverse.

Figure 16 shows the arcs that would be reversed by each structure.

The green arcs would be reversed in the neighborhood structure N_1^1.

The red arcs would be reversed in the neighborhood structure N_1^2.

The blue arcs would be reversed in the neighborhood structure N_1^3.

The neighborhood structure N_1^O would reverse the green, the red and the blue arcs, as it is defined as the union of the other three structures ($N_1^O = N_1^1 \cup N_1^2 \cup N_1^3$).

So, in this example,

- N_1^1 would have create 2 neighboring solutions by reversing 2 arcs of type MO.
- N_1^2 would have create 2 neighboring solutions by reversing an arc of type MO and an arc of type O.
- N_1^3 would have create 2 neighboring solutions by reversing an arc of type MO and an arc of type M.
- N_1^O would have create 6 neighboring solutions by reversing 4 arcs of type MO, 1 arc of type M and 1 arc of type O.

6.2.4. Makespan estimation

Makespan estimation is a very important concept in this work as it can help to save a lot of computation time. The idea is not to generate solutions that we know before-hand that are not better than the current one.

The makespan estimation is based on heads and tails. Let r'_v, r'_w, q'_v and q'_w be the heads and tails of the operations v and w after the reversal of (v, w). The makespan of the neighboring solution can be efficiently estimated as:

$$C'_{max} = max(r'_w + p_w + q'_w, r'_v + p_v + q'_v)$$

which is a lower bound of the actual makespan after reversing (v, w).

Heads and tails after a MO reversal

In order to make it simpler to understand we are using Figure 17. Remember that the arcs of type J are represented as black continuous arrows, the arcs of type M are represented as blue discontinuous arrows and the arcs of type O are represented as red dotted arrows.

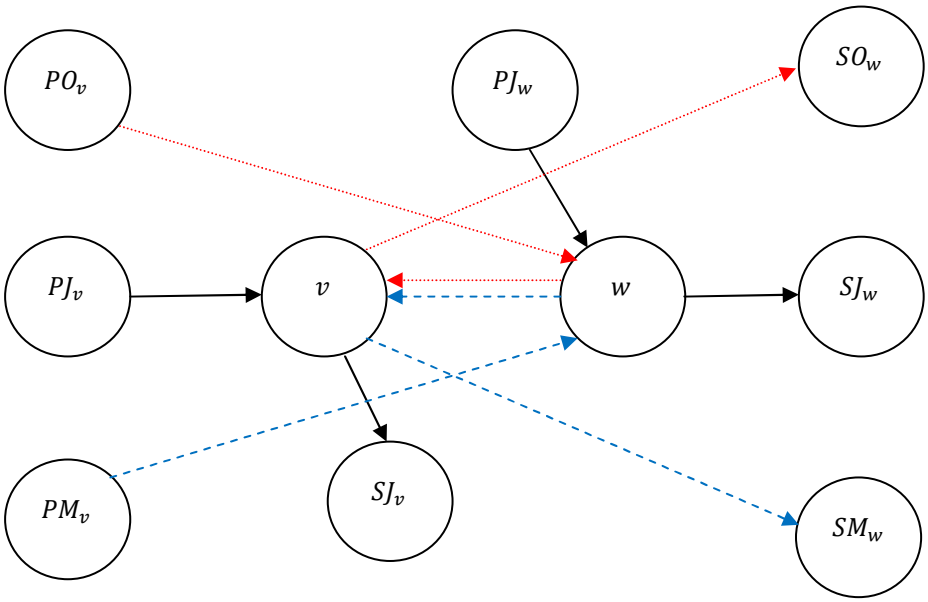


Figure 17. Graph after the arc of type MO reversal.

Let us start with the heads. In order to calculate de heads of a node we need to know the heads of its predecessors and their durations. Knowing all of them, we take the maximum due to the definition of o head.

As we can see in Figure 17, w has three predecessors, PM_v, PJ_w and PO_v. Then, its head would be calculated as $r'_w = max(r_{PM_v} + p_{PM_v}, r_{PJ_w} + p_{PJ_w}, r_{PO_v} + p_{PO_v})$. On the other hand, v has two predecessors, w (whose head has already been calculated) and PJ_v. So, its head after the reversal can be calculated as $r'_v = max(r'_w + p_w, r_{PJ_v} + p_{PJ_v})$.

Now lets us deal with the tails. In order to calculate de tail of a node we need to know the tails of its successors and their durations. Knowing all of them, we take the maximum due to the definition of tail.

As we can see in Figure 17, v has three successors, SM_w, SJ_v and SO_w. Therefore, its new tail is calculated as $q'_v = max(q_{SM_w} + p_{SM_w}, q_{SJ_v} + p_{SJ_v}, q_{SO_w} + p_{SO_w})$. On the other hand, w has two successors, v (whose tail has already been calculated), and SJ_w. So, its tail can be calculated as $q'_w = max(q'_v + p_v, q_{SJ_w} + p_{SJ_w}, q_{SO_v} + p_{SO_v})$.

To sum up, the values r'_v, r'_w, q'_v and q'_w after reversing the arc (v, w) of the type MO are

$$r'_w = max(r_{PM_v} + p_{PM_v}, r_{PJ_w} + p_{PJ_w}, r_{PO_v} + p_{PO_v})$$

$$r'_v = max(r'_w + p_w, r_{PJ_v} + p_{PJ_v})$$

$$q'_v = max(q_{SM_w} + p_{SM_w}, q_{SJ_v} + p_{SJ_v}, q_{SO_w} + p_{SO_w})$$

$$q'_w = max(q'_v + p_v, q_{SJ_w} + p_{SJ_w})$$

Heads and tails after a M reversal

In order to make it simpler to understand we are using Figure 18.

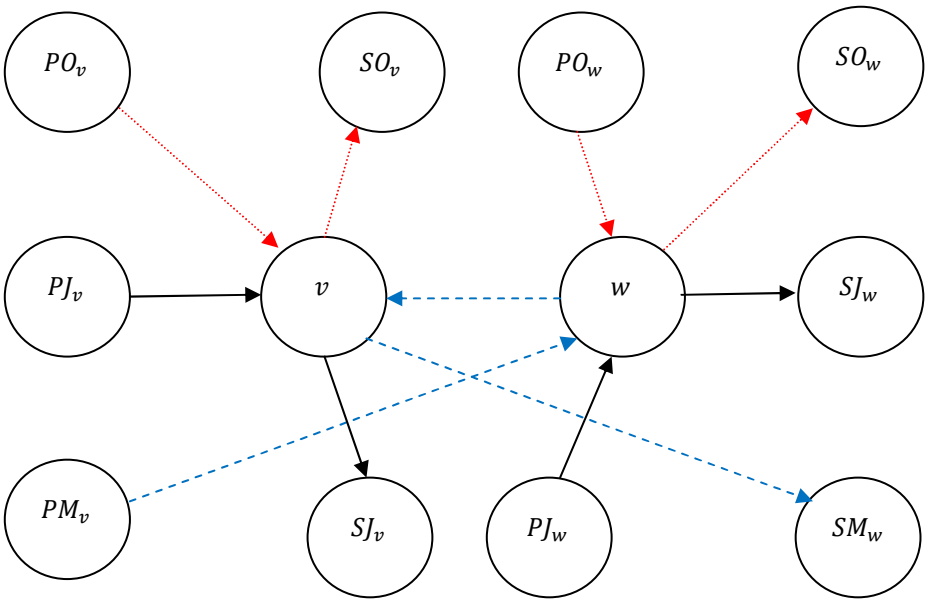


Figure 18. Graph after the arc of type M reversal.

Let us start with the heads. In order to calculate de heads of a node we need to know the heads of its predecessors and their durations. Knowing all of them, we take the maximum due to the definition of head.

As we can see in Figure 18, w has three predecessors, PM_v, PJ_w and PO_w. Then, its new head would be calculated as $r'_w = max(r_{PM_v} + p_{PM_v}, r_{PJ_w} + p_{PJ_w}, r_{PO_w} + p_{PO_w})$. On the other hand, v has another three predecessors, w (whose head has already been calculated), PO_v and PJ_v. So, its head can be calculated as $r'_v = max(r'_w + p_w, r_{PJ_v} + p_{PJ_v}, r_{PO_v} + p_{PO_v})$.

Now lets us deal with the tails. In order to calculate de tail of a node we need to know the tails of its successors and their durations. Knowing all of them, we take the maximum due to the definition of tail.

As we can see in Figure 18, v has three successors, SM_w, SJ_v and SO_v. Therefore, its new tail would be calculated as $q'_v = max(q_{SM_w} + p_{SM_w}, q_{SJ_v} + p_{SJ_v}, q_{SO_v} + p_{SO_v})$. On the other hand, w has three successors as well, v (whose tail has already been calculated), SO_w and SJ_w. So, its tail can be calculated as $q'_w = max(q'_v + p_v, q_{SJ_w} + p_{SJ_w}, q_{SO_w} + p_{SO_w})$.

To sum up, the values r'_v, r'_w, q'_v and q'_w of after reversing the arc (v, w) of the type MO are

$$r'_w = max(r_{PM_v} + p_{PM_v}, r_{PJ_w} + p_{PJ_w}, r_{PO_w} + p_{PO_w})$$

$$r'_v = max(r'_w + p_w, r_{PJ_v} + p_{PJ_v}, r_{PO_v} + p_{PO_v})$$

$$q'_v = max(q_{SM_w} + p_{SM_w}, q_{SJ_v} + p_{SJ_v}, q_{SO_v} + p_{SO_v})$$

$$q'_w = max(q'_v + p_v, q_{SJ_w} + p_{SJ_w}, q_{SO_w} + p_{SO_w})$$

Heads and tails after a O reversal

In order to make it simpler to understand we are using Figure 19.

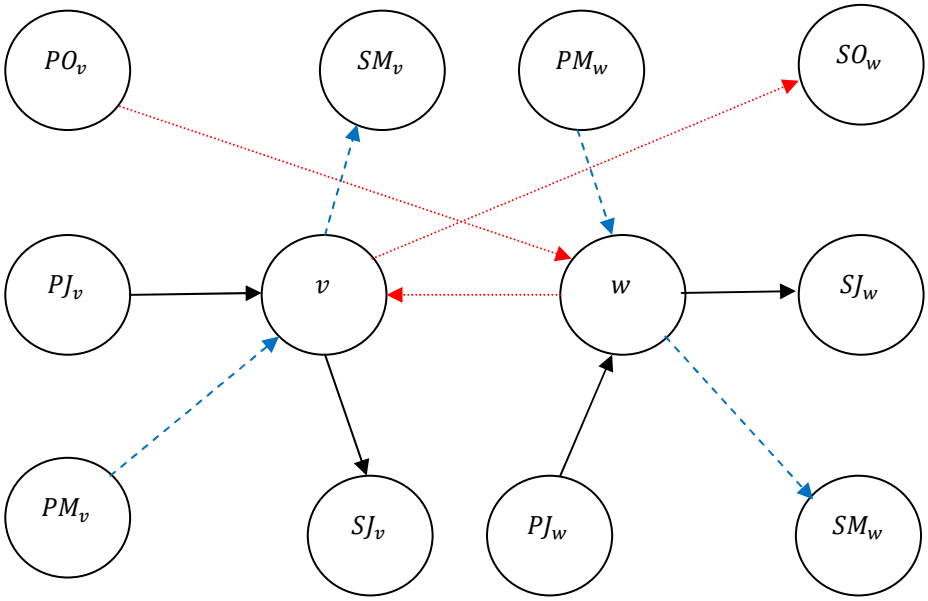


Figure 19. Graph after the arc of type O reversal.

Let us start with the heads. In order to calculate de heads of a node we need to know the heads of its predecessors and their durations. Knowing all of them, we take the maximum due to the definition of head.

As you can see in Figure 19, w has three predecessors, PM_w, PJ_w and PO_v. Then, its new head would be calculated as $r'_w = max(r_{PM_w} + p_{PM_w}, r_{PJ_w} + p_{PJ_w}, r_{PO_v} + p_{PO_v})$. On the other hand, v has another three predecessors, w (whose head has already been calculated), PM_v and PJ_v. So, its head can be calculated as $r'_v = max(r'_w + p_w, r_{PJ_v} + p_{PJ_v}, r_{PM_v} + p_{PM_v})$.

Now lets us deal with the tails. In order to calculate de tail of a node we need to know the tails of its successors and their durations. Knowing all of them, we take the maximum due to the definition of tail.

As you can see in Figure 19, v has three successors, SM_v, SJ_v and SO_w. Therefore, its new tail would be calculated as $q'_v = max(q_{SM_v} + p_{SM_v}, q_{SJ_v} + p_{SJ_v}, q_{SO_w} + p_{SO_w})$. On the other hand, w has three successors as well, v (whose tail has already been calculated), SM_w and SJ_w. So, its tail can be calculated as $q'_w = max(q'_v + p_v, q_{SJ_w} + p_{SJ_w}, q_{SM_w} + p_{SM_w})$.

To sum up, the values r'_v, r'_w, q'_v and q'_w of after reversing the arc (v, w) of the type MO are

$$r'_w = max(r_{PM_w} + p_{PM_w}, r_{PJ_w} + p_{PJ_w}, r_{PO_v} + p_{PO_v})$$

$$r'_v = max(r'_w + p_w, r_{PJ_v} + p_{PJ_v}, r_{PM_v} + p_{PM_v})$$

$$q'_v = max(q_{SM_v} + p_{SM_v}, q_{SJ_v} + p_{SJ_v}, q_{SO_w} + p_{SO_w})$$

$$q'_w = max(q'_v + p_v, q_{SJ_w} + p_{SJ_w}, q_{SM_w} + p_{SM_w})$$

6.3. Example

Here is an example with a real schedule in which we can see more clearly how to apply the proposed neighborhood structure and how to improve the schedule with a slight transformation.

It starts with a schedule for a JSO problem instance of 3 jobs, 3 machines and 2 operators.

Figure 20 shows the corresponding disjunctive graph. In it, conjunctive arcs are represented with continuous lines, discontinuous lines represent the disjunctive arcs and dotted lines represent the operator arcs. For the sake of clarity, we only show the arcs that connect adjacent operations in the schedule.

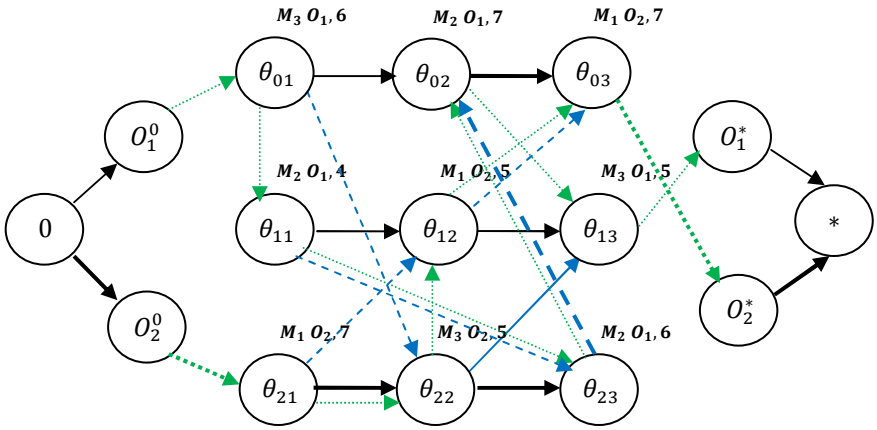


Figure 20: Disjunctive graph of the starting schedule.

Also, thick arcs represent a critical path in the graph, formed by the nodes (0, O_2^0, θ_{21}, θ_{22}, θ_{23}, θ_{02}, θ_{03}, O_2^*, $*$), whose cost is 32. Thus, the schedule has a makespan that is equal to 32.

In order to facilitate understanding, the following Gantt chart of the previous schedule is given in Figure 21. Shaded operations are the critical path.

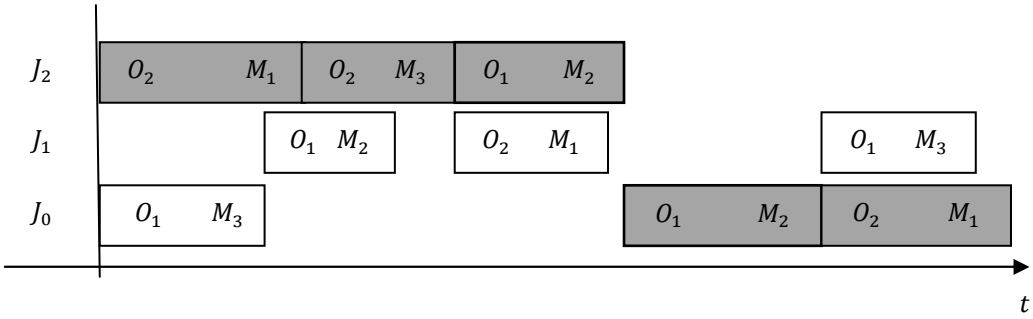


Figure 21: Gantt chart of the starting schedule.

Based on a critical path of the schedule, the neighborhood structure proposed calculates firstly the critical blocks. Figure 22 shows the critical path including the machines and the operators that each task uses.

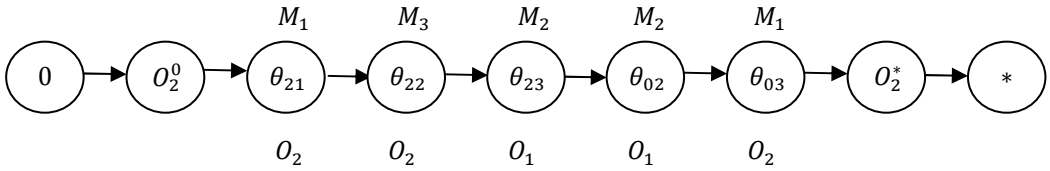


Figure 22. Critical path of the schedule.

As it can be observed, between operations θ_{21} and θ_{22} there are a conjunctive arc and an arc of operators. Similarly, operations θ_{23} and θ_{02} are connected by two arcs, one for operators and one disjunctive.

Machines critical blocks are maximal sequences of operations that share the machine. In the critical path depicted above 4 machines critical blocks can be identified. However, there is only one containing more than one operation and so it is the only one interesting for generating neighbors. This is shown in Figure 23.

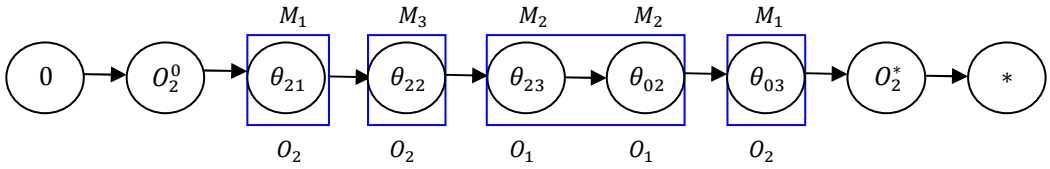


Figure 23. Machines critical blocks.

Operator's critical blocks are maximal sequences of operations that share the same operator. In the critical path depicted above 3 operator's critical blocks can be identified. However, there is only one containing more than one operation, so it is the only one interesting for generating neighbors. This is shown in Figure 24.

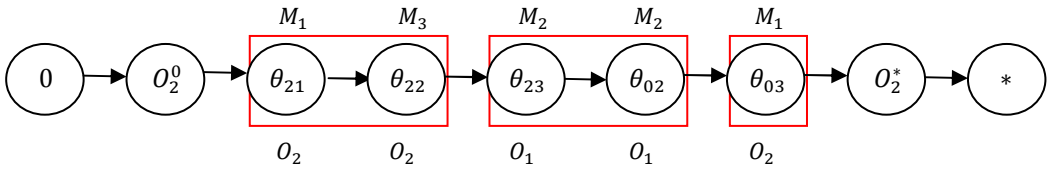


Figure 24. Operator's critical blocks.

Figure 25 shows the machines critical blocks marked in blue and the operator's critical blocks marked in red. The types of the arcs are added as well.

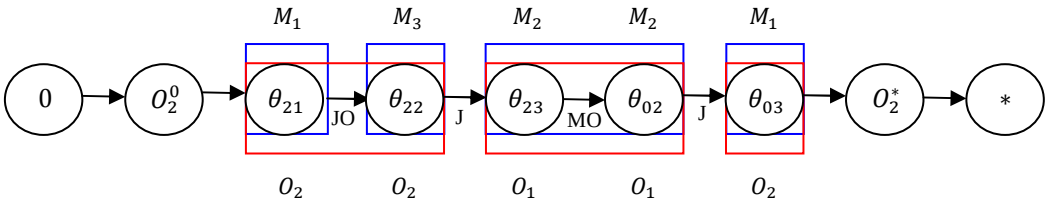


Figure 25. Critical blocks.

As explained before, N_1^1 generates neighbors reversing arcs that are both at the border of a machines critical blocks and at the border of an operator's critical blocks (with the exception of the first arc of the first block and the last arc of the last block in the case in which these blocks have more than 2 operations). N_1^2 generates neighbors reversing arcs that are at the border of an operator's critical block and are not at the border of a machines critical block (with the exception of the first arc of the first block and the last arc of the last block in the case in which these blocks have more than 2 operations). N_1^3 generates neighbors reversing arcs that are at the border of a machines critical block and are not at the border of an operator's critical block (with the exception of the first arc of the first block and the last arc of the last block in the case in which these blocks have more than 2 operations). N_1^O generates all the neighbors that the previous 3 neighborhood structures generates.

So, looking at Figure 25,

- The neighborhood structure N_1^1 would generate one neighbor reversing the arc (θ_{23}, θ_{02}) of type MO.
- The neighborhood structure N_1^2 would not generate any neighbor because the arc that it has is (θ_{21}, θ_{22}), but it is of type JO so it cannot be reversed.
- The neighborhood structure N_1^3 would not generate any neighbor because it does not have any arc.
- The neighborhood structure N_1^O would generate one neighbor reversing the arc (θ_{23}, θ_{02}) of type MO.

The block has two marked critical operations, θ_{23} and θ_{02}, which require the machine M_2 and are assisted by the operator op_1. Therefore, to generate a neighbor schedule we only need to exchange the order of the operations (θ_{02} is processed before θ_{21}). This requires reversing both disjunctive and operator arcs, as well as reorganizing the new predecessor of θ_{21} and the new successor of θ_{02}.

Before generating definitively this neighbor, in accordance with the proposed method, we have to consider its estimated makespan, as it is indicated in Section [**Makespan estimation**]. If this value were larger or equal than the actual makespan of the current solution, the neighbor would be discarded as it is not an improving schedule. This procedure may discard efficiently many non improving solutions. The makespan estimation is based on heads and tails. The values r'_v, r'_w, q'_v and q'_w after reversing the arc (v, w) are different depending on the type of arc we are reversing as explained in the previous section. In this case, v is θ_{23} and w is θ_{02}. As said before, the arc we are reversing is of type MO.

The heads and tails are calculated as follows.

$$r'_w = max\left(r_{PM_v} + p_{PM_v}, r_{PJ_w} + p_{PJ_w}, r_{PO_v} + p_{PO_v}\right)$$
$$= max\left(r_{\theta_{11}} + p_{\theta_{11}}, r_{\theta_{01}} + p_{\theta_{01}}, r_{\theta_{11}} + p_{\theta_{11}}\right) = max(6+4, 0+6, 6+4) = 10$$

$$r'_v = max\left(r'_w + p_w, r_{PJ_v} + p_{PJ_v}\right) = max\left(r'_w + p_w, r_{\theta_{22}} + p_{\theta_{22}}\right) = max(10+7, 7+5)$$
$$= 17$$

$$q'_v = max\left(q_{SM_w} + p_{SM_w}, q_{SJ_v} + p_{SJ_v}, q_{SO_w} + p_{SO_w}\right) = max\left(q_* + p_*, q_* + p_*, q_{\theta_{13}} + p_{\theta_{13}}\right)$$
$$= max(0+0, 0+0, 0+5) = 5$$

$$q'_w = max\left(q'_v + p_v, q_{SJ_w} + p_{SJ_w}\right) = max\left(q'_v + p_v, q_{\theta_{03}} + p_{\theta_{03}}\right) = max(5+6, 0+7) = 11$$

Using them we estimate the makespan as

$$C'_{max} = max\left(r'_w + p_w + q'_w, r'_v + p_v + q'_v\right) = max(10+7+11, 17+6+5) = 28$$

The estimated makespan (28) which is a lower bound of the makespan of the schedule that we would generate by reversing the arc, is lower than the current makespan (32), which means that the resulting solution could improve the current one, so we will generate it.

Figure 26 shows the disjunctive graph corresponding to the generated neighboring solution. As before, only the operations connecting adjacent arcs in the schedule are depicted and the critical path is marked with thick arcs.

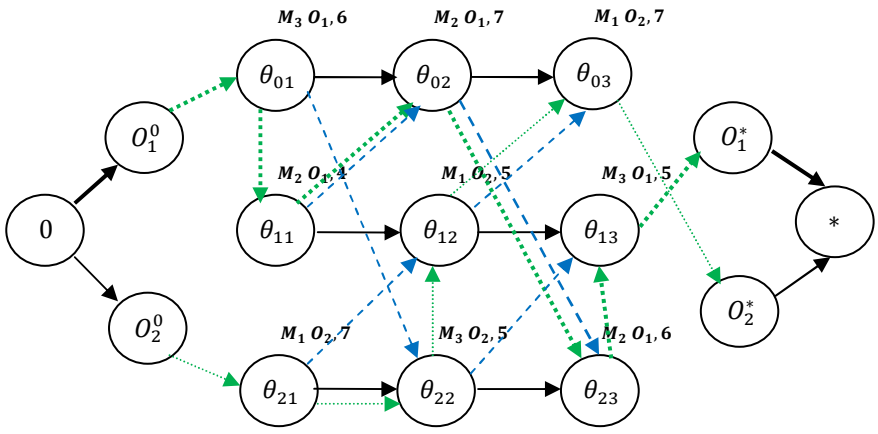


Figure 26: Disjunctive graph of the neighboring schedule.

As it can be seen in the neighboring solution a critical path is (0, O_1^0, θ_{01}, θ_{11}, θ_{02}, θ_{23}, θ_{13}, O_1^*, $*$), whose makespan is 28. Therefore, the neighboring solution is better than the previous one.

Figure 27 is the Gantt chart associated with this new schedule.

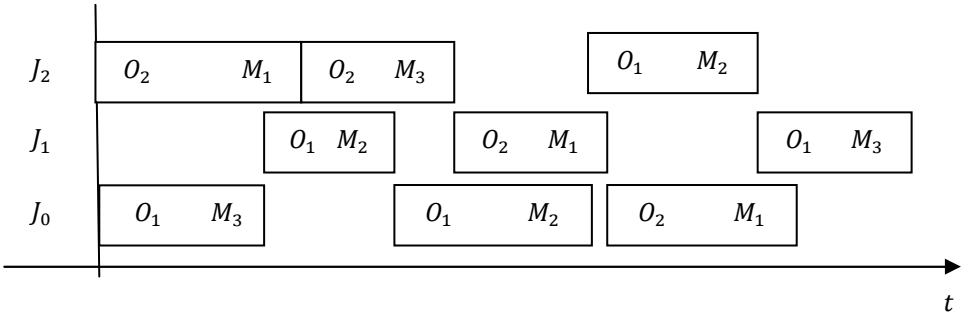


Figure 27: Gantt chart of the neighboring schedule.

In this example, there is only one neighboring schedule which has a makespan lower than that of the original schedule, the local search algorithm will repeat the process from the new schedule until there were no improvements or any other criterion fulfills.

7. Memetic Algorithm for the JSO

A memetic algorithm is defined from the combination of a genetic algorithm with local search. This kind of algorithms has been successful over the last years for a variety of combinatorial optimization problems [Talbi 2009], including scheduling problems [Vela et al. 2010].

The memetic algorithms proposed combine the genetic algorithm described in Section 4. with a Local Search Algorithm that uses the Neighborhood structures described in Section 6.

As a result, we have four memetic algorithms, that we name, MA-N_1^O, MA-N_1^1, MA-N_1^2 and MA-N_1^3.

- MA-N_1^O uses the neighborhood structure N_1^O.
- MA-N_1^1 uses the neighborhood structure N_1^1.
- MA-N_1^2 uses the neighborhood structure N_1^2.
- MA-N_1^3 uses the neighborhood structure N_1^3.

The proposed local search algorithms use maximum gradient hill-climbing as selection/acceptation criteria. So, each time the local search is issued, all neighbors are calculated. If any of them is better than the current solution, then the best neighbor is taken as the new current solution, otherwise the algorithm finishes and returns the current chromosome.

As we will see in the experimentation, in the first place we had opted to apply the local search algorithm to all the individuals generated by the genetic algorithm, instead of a portion of them. After that, we did the experiments issuing a local search with a probability of 0.2 for each individual, in order to gain diversity.

The evolution model used is Lamarckian, i.e., the improved individuals substitute the original chromosomes after the local search when there are improvements, unlike in the Baldwinian model where only the fitness values are updated.

8. Experimental Study

The purpose of this experimental study is to assess the efficiency and effectiveness of the methods proposed in this project.

For doing so, we have experimented across a set of problem instances with different sizes and characteristics taken from the OR-Library [Beasley 1990] for the classical job shop scheduling problem. Concretely, we have used a relevant benchmark proposed in [Applegate and Cook 1991] that consists of the following instances:

- 10x10: FT10
- 20x5: FT20
- 15x10: LA21, LA24, LA25
- 20x10: LA27, LA29
- 15x15: LA38, LA40
- 20x15: ABZ7, ABZ8, ABZ9

From each of them we have built new JSO(n, p) instances by ranging the number of operators available (p), from 1 to min{n,m}. Note that instances with p = min{n, m} are also instances of the JSS problem. So, we have 140 instances in all.

For each instance, all the algorithms were run several times, giving the cost of the best solution found, the average cost of the solutions returned, the standard deviation and the average computation time taken. The first two values measure the effectiveness of the algorithm, while the fourth is a measure of the efficiency of the method. Then, for evaluating and comparing them, we have used the following metrics for each instance:

- Best solution error (%), calculated as

$$100 \times \frac{UB - Best\ solution}{UB}$$

- Solution's average error (%), calculated as

$$100 \times \frac{UB - Solution's\ average}{UB}$$

- Pearson's variation coefficient (%), calculated as

$$CV = 100 \times \frac{Solution's\ standart\ deviantion}{Solution's\ average}$$

where *UB* is the cost of the best solution found across the whole experimental study for the problem instance.

The algorithms were coded in C++ and the target machine was Scientific Linux (64-bit) on Intel Xeon (2.26 GHz), 24 GB RAM.

Below, we first present the results from the Hybrid OG&T algorithm, presented in Section 5 for the JSO(n,p) problem with total flow time minimization. Then, we report the results from the memetic algorithms introduced in Section 7, for the JSO(n,p) problem with makespan minimization.

The complete results can be shown in the Appendix.

8.1. Hybrid OG&T algorithm for total flow time minimization

The Hybrid OG&T algorithm was used as decoding method in the genetic algorithm presented in [Mencía et al. 2011]. In this case, the objective function considered is total flow time minimization.

The impossibility of performing all combinations that exist among the configuration parameters of the genetic algorithm has led to the determination of using the following configuration, which provides good results for the problem:

- **Crossover probability:** 0.7
- **Mutation probability:** 0.2
- **Number of runs:** 30
- **Selection type:** Tournament
- **Crossover type:** JOX
- **Mutation type:** Simple1.
- **Fitness type:** OG&T - TOTAL FLOW TIME
- **Population size:** 100
- **Number of generations:** 120

For evaluating the effect of the Hybrid OG&T algorithm in the performance of the genetic algorithm we considered 5 different values for the parameter delta: 0, 0.25, 0.5, 0.75 and 1.

Table 1 reports, for each value of delta, the error in percentage of both the best and the average solutions reached, averaged for instances with the same number of jobs and machines. Overall, the best value for delta is 0.75. Also, the effectiveness of the algorithm decreases as delta gets smaller.

	Delta = 0,00		**Delta = 0,25**		**Delta = 0,50**		**Delta = 0,75**		**Delta = 1,00**	
Instances size	**E.Best (%)**	**E.Avg (%)**	**E.Best (%)**	**E.Avg (%)**	**E.Best (%)**	**E.Avg (%)**	**E.Best (%)**	**E.Avg (%)**	**E.Best (%)**	**E.Avg (%)**
10x10	16.73	18.58	10.49	12.36	4.24	6.89	0.74	3.73	0.53	4.63
20x5	19.18	24.4	8.93	14.14	4.54	9.72	1.59	7.14	0.53	6.66
15x10	7.53	9.81	4.84	7.71	2.84	5.86	1.11	4.39	0.80	4.92
15x15	5.29	6.58	1.95	3.97	1.08	3.29	0.52	3.19	1.47	4.38
20x10	6.89	9.00	4.61	6.97	2.38	5.24	0.79	4.03	0.82	4.73
20x15	2.55	4.15	1.50	3.43	0.88	3.01	0.45	3.09	1.80	4.62
Avg.	9.70	12.09	5.39	8.10	2.66	5.67	0.87	4.26	0.99	4.99

Table 1. Results from the genetic algorithm using the Hybrid OG&T with different values of delta.

Figures 28-33 shows the error in average for each instances size and value for delta.

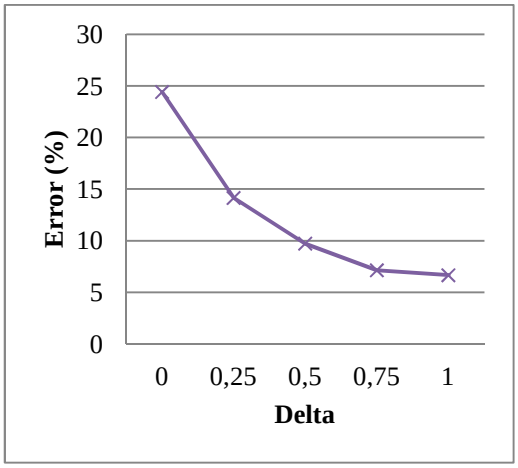


Figure 28. Average error (%) comparison between the different delta values (20x5).

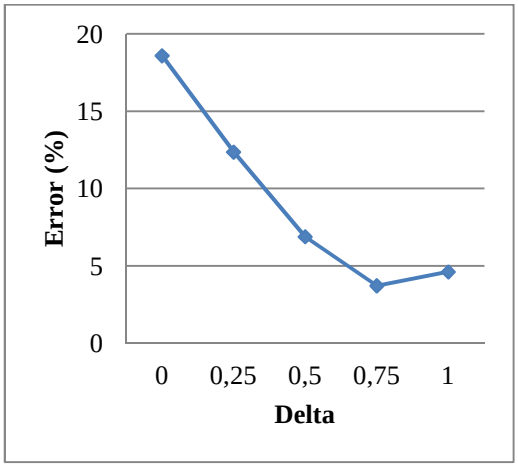


Figure 29. Average error (%) comparison between the different delta values (10x10).

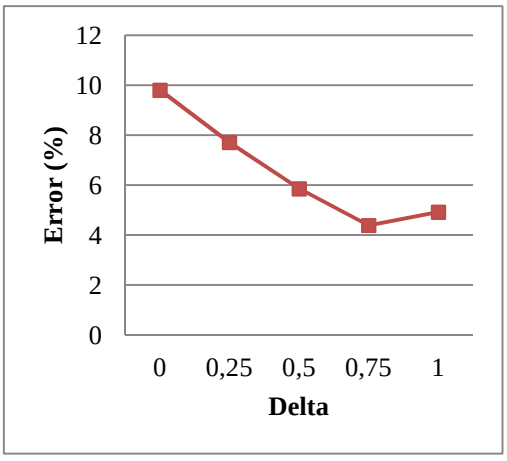


Figure 30. Average error (%) comparison between the different delta values (15x10).

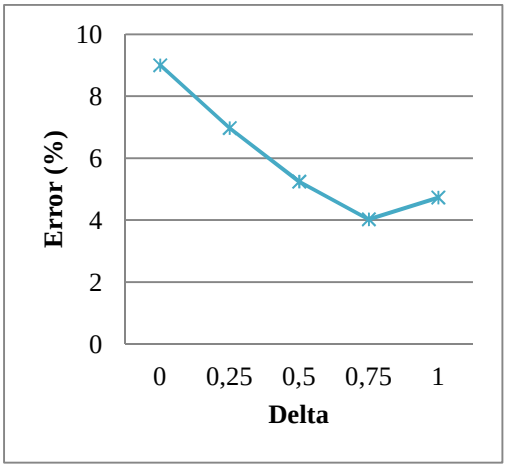


Figure 31. Average error (%) comparison between the different delta values (20x10).

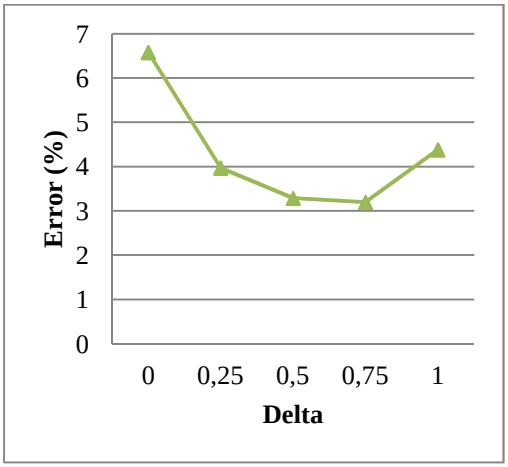


Figure 32. Average error (%) comparison between the different delta values (15x15).

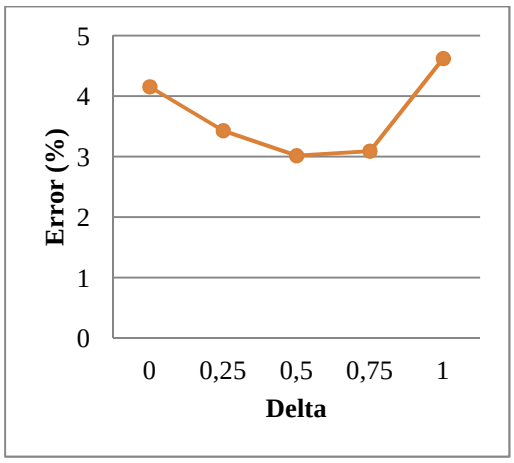


Figure 33. Average error (%) comparison between the different delta values (20x15).

As we can see, as long as the problem instances get larger, a smaller value of delta, which achieves more reduction of the search space, is the best option.

- For small instances of size 20x5, the genetic algorithm reaches the best results with a delta value of 1, and we can see that the lower the delta is the worse the results the GA reaches are.
- Regarding instances of sizes 10x10, 15x10 and 20x10, it reaches the best results with a delta value of 0.75, then with 1, and after that with 0.5, 0.25, and 0.
- Then, for instances of size 15x15, it reaches the best results with a delta of 0.75, but then it gets the next best results with deltas of 0.5 and 0.25, and then with 1 and 0.
- Finally, for the largest instances of size 20x15, we reach the best results with a delta of 0.5, then 0.75, 0.25, 0, and finally, it gets the worst results with a delta of 1.

This is quite reasonable, as for small instances, a large reduction of the search space may lose good solutions and may make the genetic algorithm to converge prematurely. At the same time, for large instances, a large search space may be hard to explore, so reducing it seems to be worthwhile.

Another interesting result we can observe from this experimentation is referred to the stability of the algorithm depending on the value of delta. As we can see in Figure 34, which shows the Pearson's variation coefficient averaged for instances with the same size, the lower delta the more stable the algorithm.

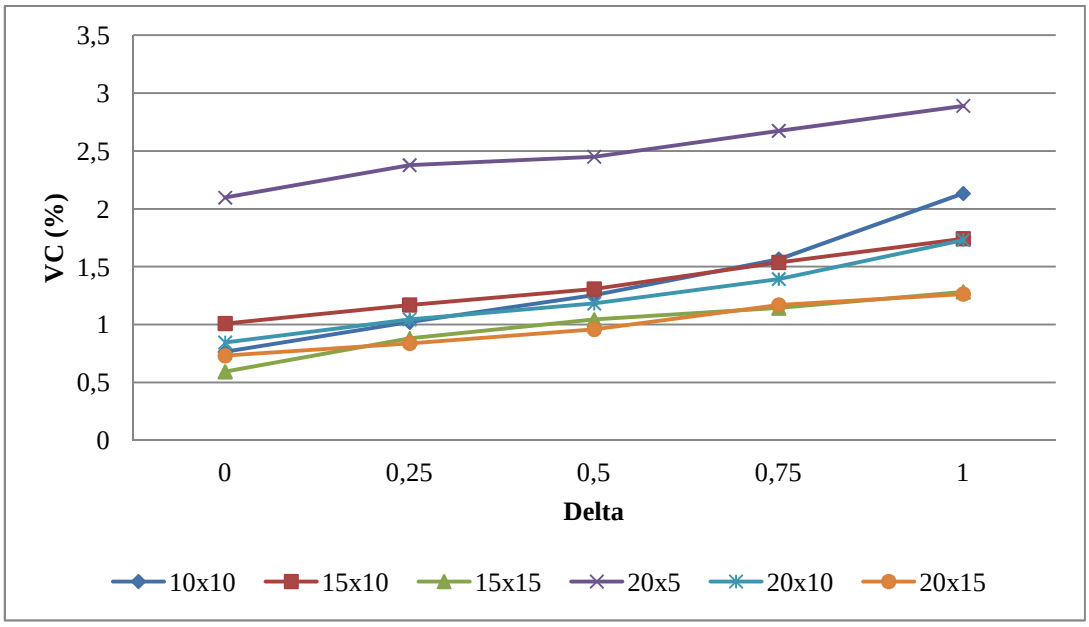


Figure 34. VC (%) comparison between the different values of delta.

Finally, regarding the computation time taken, we have not observed significant differences among the different values of delta considered.

8.2. Memetic algorithms for makespan minimization

This part of the experimental study is intended to evaluate the memetic algorithms (MAs) using each of the four neighborhood structures proposed in this project and compare them with the genetic algorithm (GA).

We first present some results about the computation time required by each local search method and then we compare the MAs with the GA giving them the same computation time.

8.2.1. Evaluation of the Neighborhood Structures

A very important characteristic of a memetic algorithm is how much computation time takes the genetic algorithm part of the MA and how much time takes the local search.

Figure 35 shows how much time takes in proportion each part in average across the 140 instances considered in the experimental study. In these experiments, the local search was issued from each individual of a population with size 200 along a number of 300 generations.

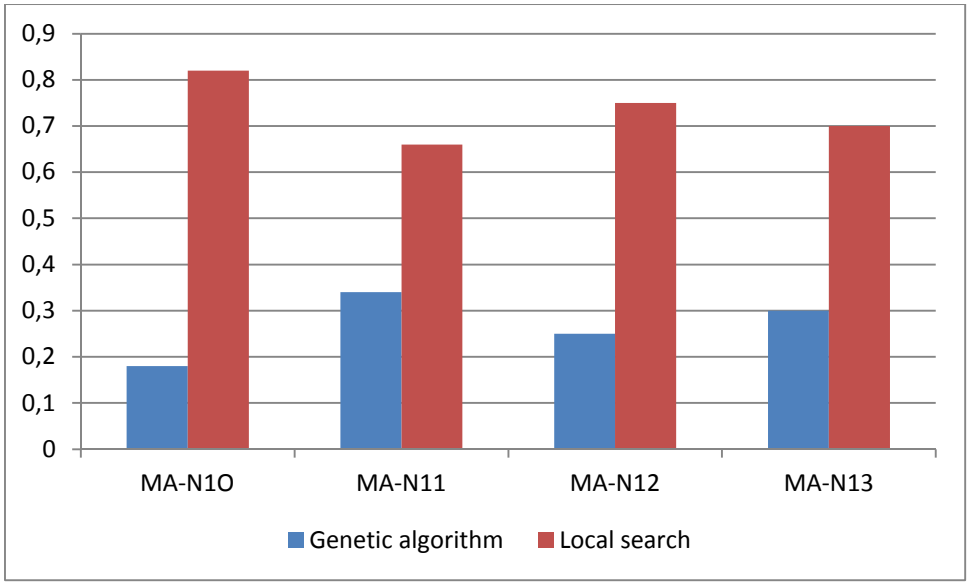


Figure 35. Weights of the GA and the local search in each MA.

It can be observed that there are only slightly differences among **MA-N_1^1**, **MA-N_1^2** and **MA-N_1^3**. Firstly, **MA-N_1^1** is the algorithm where the local search takes the least time (66%) among the four memetic algorithms. Then **MA-N_1^3** followed by **MA-N_1^2**, with 70% and 75% of the time respectively.

Finally, in **MA-N_1^O** the local search takes the largest time (82%). This was expected as the neighborhood structure N_1^O is defined as the union of the other three, so it produces the largest number of neighbors.

8.2.2. Comparison among GA and the different MAs

With the aim of assessing the effectiveness of the methods proposed in this project for the JSO(n,p) problem with makespan minimization, we made several experiments with the GA and the four MAs. The parameters of the genetic algorithm are similar than those used in the section devoted to the Hybrid OG&T algorithm, but in this case we only performed 10 runs for each instance and method as the available time was scarce.

- **Crossover probability:** 0.7
- **Mutation probability:** 0.2
- **Number of runs:** 10
- **Selection type:** Tournament
- **Crossover type:** JOX
- **Mutation type:** Simple1.
- **Fitness type:** OG&T - Makespan

We have also considered two variants for each memetic algorithm with respect to the portion of chromosomes from which a local search is issued. Firstly, we applied local search to all the population, and second, we have given each individual a probability of 0.2 of yielding a local search from it.

In all the experiments, we have given every algorithm a **population size** of 100 chromosomes and set a time limit of 60 seconds. So, each algorithm will perform a different number of generations depending on how much time it takes it to compute a generation.

8.2.2.1. MAs applying local search to all the population

Table 2 shows the error in percentage of both the best and the average solutions reached, averaged for instances with the same number of jobs and machines from GA and the four MAs. These results are also represented graphically in Figures 36 and 37.

	GA		MA-N_1^0		MA-N_1^1		MA-N_1^2		MA-N_1^3	
Instance size	**E.Best (%)**	**E.Avg (%)**	**E.Best (%)**	**E.Avg (%)**	**E.Best (%)**	**E.Avg (%)**	**E.Best (%)**	**E.Avg (%)**	**E.Best (%)**	**E.Avg (%)**
20x15	1,22	1,96	0,78	1,5	0,56	1,34	0,62	1,43	0,79	1,49
10x10	0,8	2,19	0,64	1,7	0,29	1,53	0,59	1,74	0,45	1,74
20x5	0,99	1,76	0,23	0,8	0,2	1,02	0,52	1,23	0,38	1,26
15x10	1,06	1,7	0,57	1,2	0,43	1,17	0,47	1,25	0,57	1,27
20x10	0,87	1,4	0,41	0,9	0,39	0,9	0,45	1,08	0,58	1,07
15x15	1,55	2,57	0,74	1,8	0,71	1,73	0,77	1,79	0,68	1,7

Table 2. Results from GA and MAs averaged for instances with the same size.

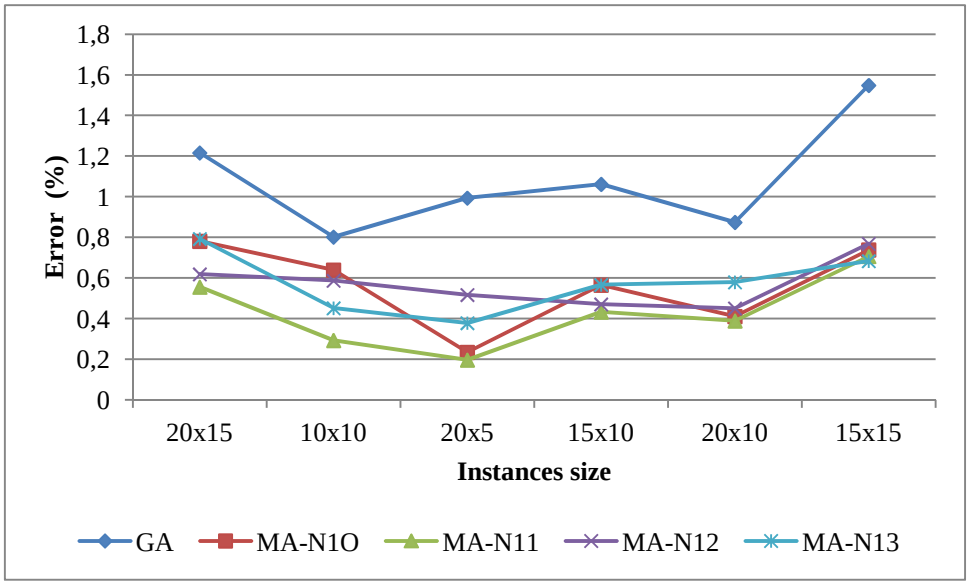


Figure 36. Avg. Error of the best solution reached for GA and MAs (instances with the same size)

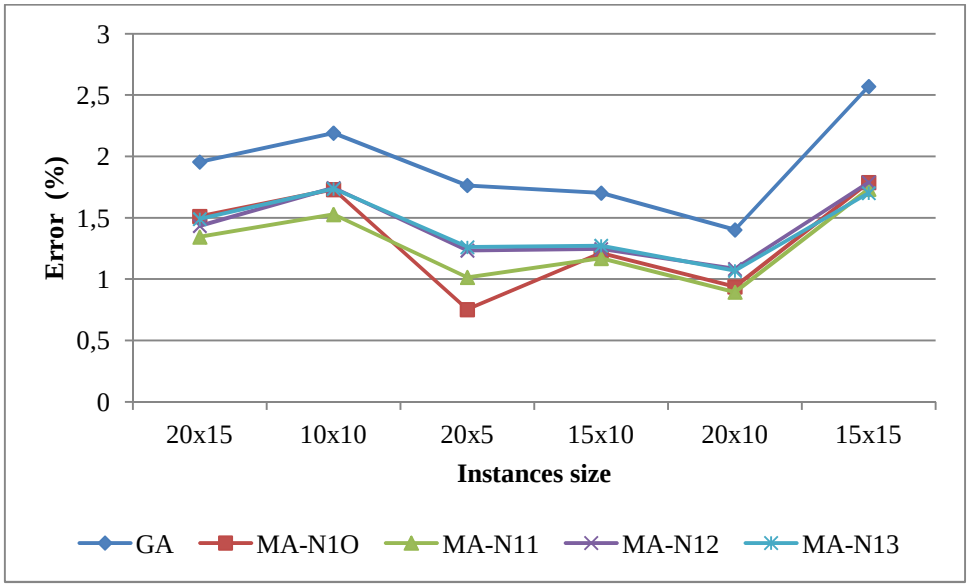


Figure 37. Avg. Error of the average solution reached for GA and MAs (instances with the same size)

We can observe that the GA achieves the worst results, for both the best and the average solutions returned, in all the subsets of instances with the same size. Regarding the memetic algorithms, it seems that it is MA-N11 which reaches the best results in almost all cases. MA-N12, MA-N13 and MA-N1O show a less stable behavior, so it is difficult to determine which of them is the best.

For reaching more solid conclusions, following [García et al. 2009] we have made a Friedman Test for multiple comparisons considering all the instances in the experimental study, so

establishing a ranking among the methods w.r.t. the average error reached. Table 3 shows the average ranking:

Algorithm	Ranking
MA-N11	2.47
GA	3.00
MA-N12	3.15
MA-N13	3.16
MA-N1O	3.22

Table 3. Average ranking.

The Friedman test produced a p-value of 2.99E-4, and the Iman and Davenport test returned a p-value of 2.61E-4. So, there are significant differences among the methods in the ranking.

We computed a poshoc analysis, concluding that Bergmann's procedure rejects the following hypotheses with a high level of confidence ($\alpha = 0.05$):

- MA-N11 vs GA
- MA-N11 vs MA-N12
- MA-N11 vs MA-N13
- MA-N11 vs MA-N1O

So, there is strong statistical evidence that MA-N11 outperforms all the other methods. At the same time, there are no significant differences among GA, MA-N12, MA-N13 and MA-N1O, although the average values reported in Table 2 are smaller for the memetic algorithms than for GA. The rejected hypotheses are the same with $\alpha = 0.10$.

Figures 38-43 show for each group of instances with the same size, the error in percentage of the average cost of the solutions, averaged for instances with the same number of operators available.

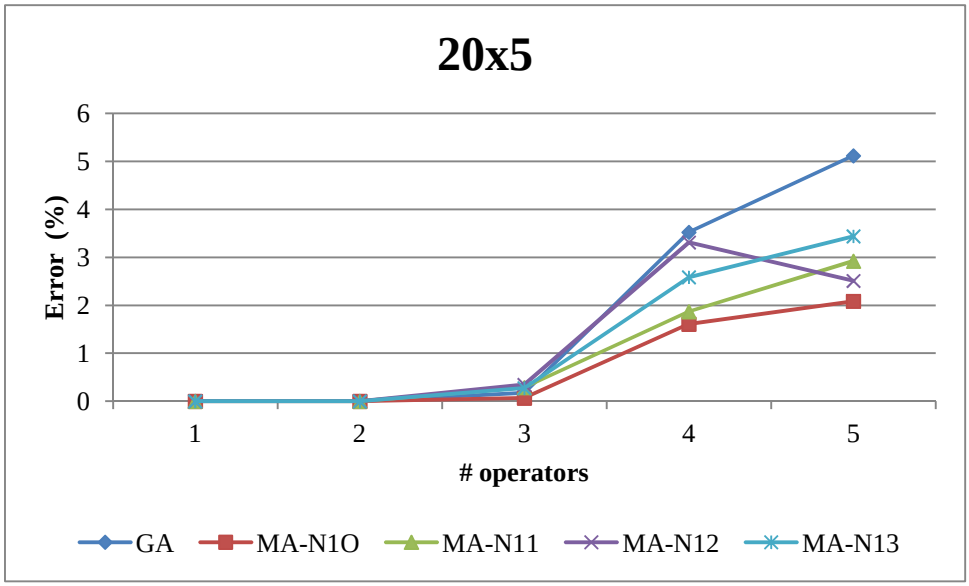


Figure 38. Comparison between the average solution of the considered algorithms (20x5).

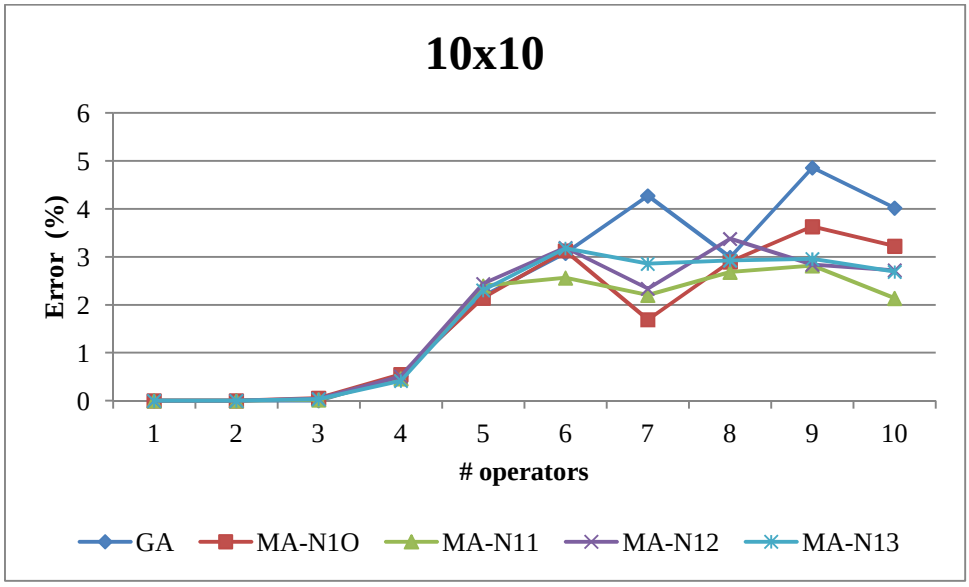


Figure 39. Comparison between the average solution of the considered algorithms (10x10).

For small instances, with up to 100 operations, we can observe that the GA is always outperformed by some memetic algorithm. Also, the problem gets harder as there are more operators available, as indicated by the error obtained by the methods. Regarding the memetic algorithms, MA-N11 and MA-N1O obtain the best results in almost all subsets.

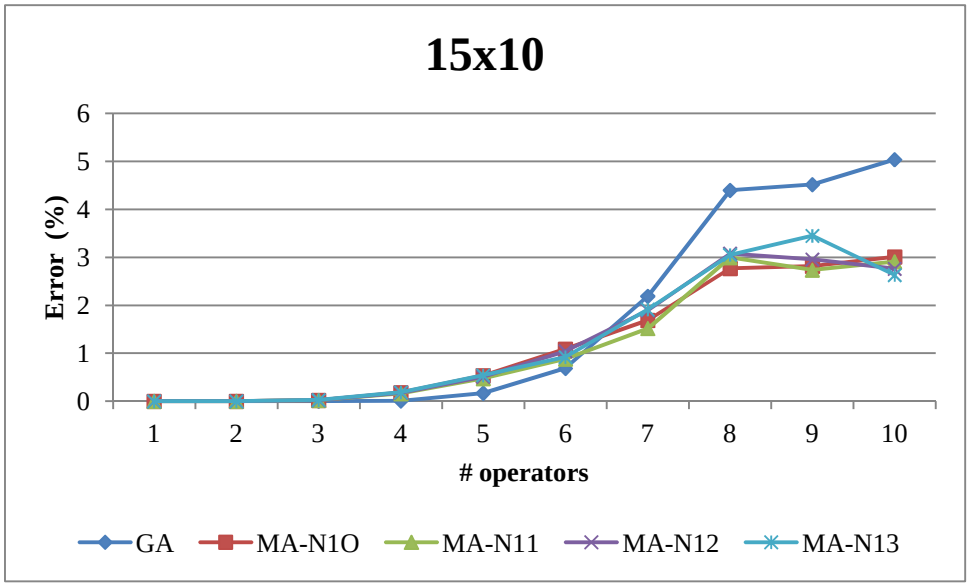


Figure 40. Comparison between the average solution of the considered algorithms (15x10).

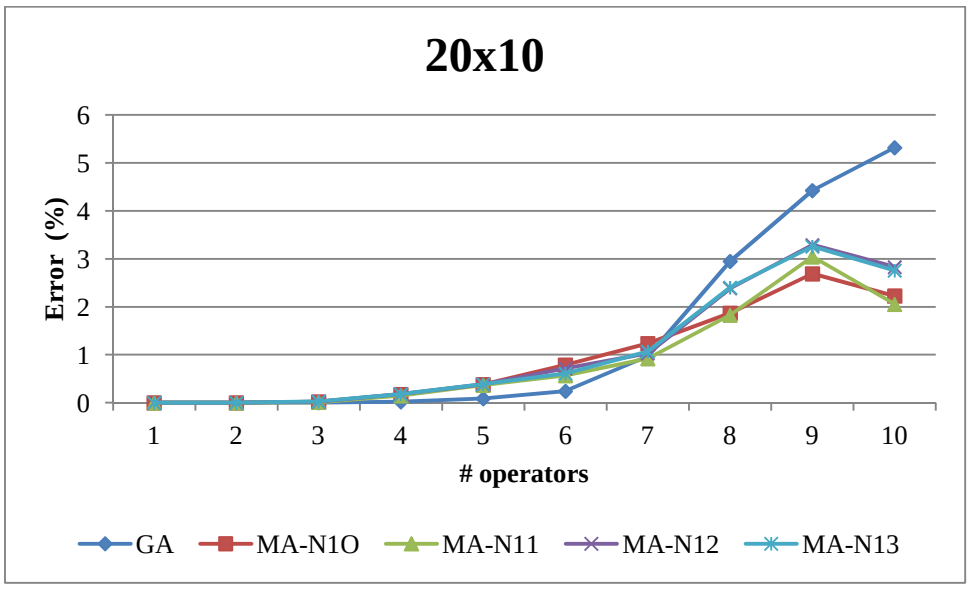


Figure 41. Comparison between the average solution of the considered algorithms (20x10).

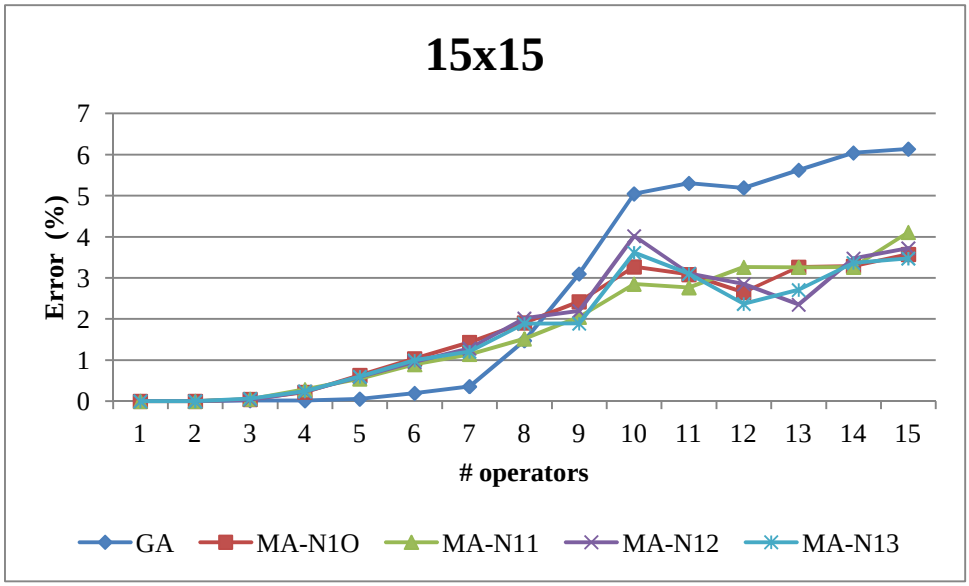


Figure 42. Comparison between the average solution of the considered algorithms (15x15).

Regarding large instances with sizes 15x10, 20x10, 15x15 and 20x15, the results of the algorithms w.r.t. the number of available operators show a different trend than for the small instances. When there are few operators available all the methods easily reach the best known solution. On the other hand, for intermediate values of the number of operators, the GA outperforms the memetic algorithms, and when there are many available operators, the memetic algorithms return the best results, especially MA-N11 and MA-N1O.

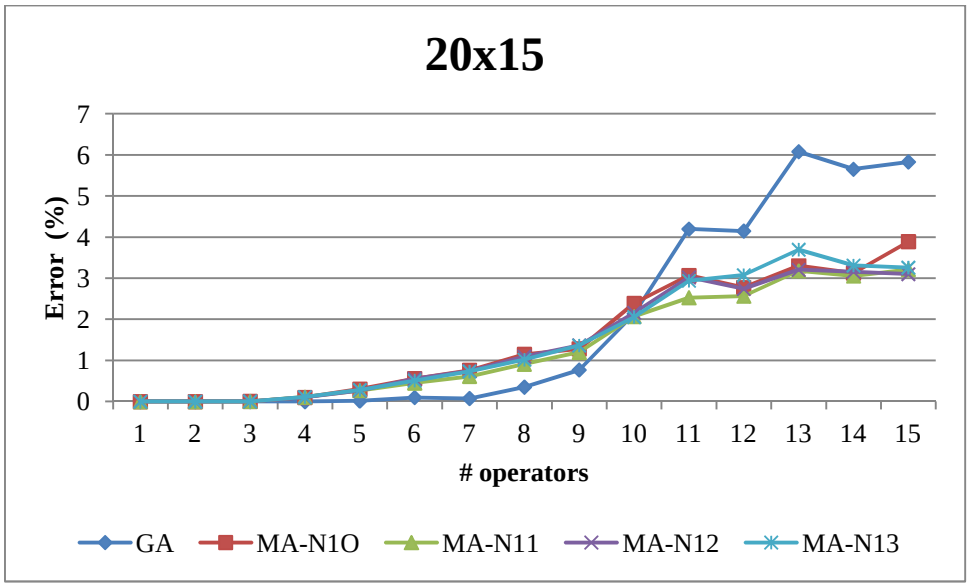


Figure 43. Comparison between the average solution of the considered algorithms (20x15).

From this results, it seems that the local search methods proposed in this project are only worth using within a memetic algorithm for a large number of operators available. We have conducted some experiments not reported in this document, giving the algorithms the same number of evaluations (population size = 200 and number of generations = 300). The results show a quite similar trend, which indicates that the local search makes the memetic algorithms converge prematurely when there is an intermediate number of operators.

8.2.2.2. MAs applying local search with a probability of 0,2

Motivated by the reasons presented above, we conducted a last series of experiments limiting the number of local searches issued during the search. The development of this part of the experimental study followed the same steps as in the previous section.

Table 4 shows the error in percentage of both the best and the average solutions reached, averaged for instances with the same number of jobs and machines from GA and the four MAs. These results are also represented graphically in Figures 44 and 45.

	GA		MA-N_1^0		MA-N_1^1		MA-N_1^2		MA-N_1^3	
Instance size	**E.Best (%)**	**E.Avg (%)**	**E.Best (%)**	**E.Avg (%)**	**E.Best (%)**	**E.Avg (%)**	**E.Best (%)**	**E.Avg (%)**	**E.Best (%)**	**E.Avg (%)**
20x5	0,99	1,76	0,28	0,9	0,48	1,12	0,1	0,84	0,27	1,16
10x10	0,8	2,19	0,22	1,4	0,29	1,34	0,3	1,54	0,24	1,5
15x10	1,06	1,7	0,22	1	0,27	0,99	0,27	0,99	0,34	1
20x10	0,87	1,4	0,37	0,8	0,13	0,76	0,3	0,82	0,28	0,79
15x15	1,55	2,57	0,56	1,5	0,58	1,47	0,46	1,52	0,35	1,4
20x15	1,22	1,96	0,35	1,1	0,38	1,07	0,27	1,05	0,37	1,16

Table 4. Results from GA and MAs averaged for instances with the same size.

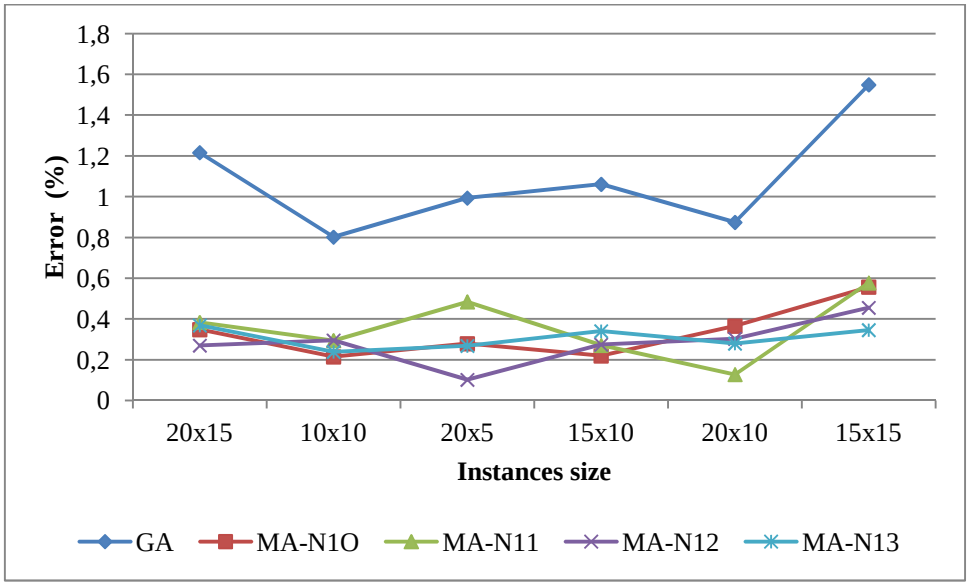


Figure 44. Avg. Error of the best solution reached for GA and MAs (instances with the same size)

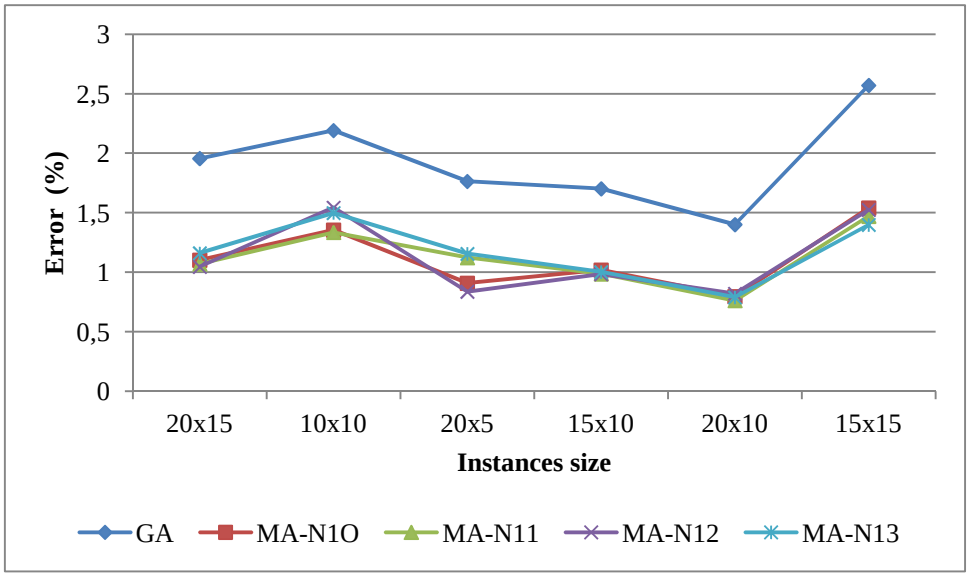


Figure 45. Avg. Error of the average solution reached for GA and MAs (instances with the same size)

In this case the results are fairly similar than before. The GA achieves the worst results in all the subsets of instances with the same size. However, the memetic algorithms reach better solutions than those applying the local search to all the population, and there seem to be less differences between their results.

We have also made a Friedman Test for multiple comparisons considering all the instances in the experimental study, so establishing a ranking among the methods w.r.t. the average error reached. Table 5 shows the average ranking:

Algorithm	Ranking
MA-N11	2.41
MA-N13	2.69
MA-N12	3.00
MA-N1O	3.25
GA	3.65

Table 5. Average ranking.

The Friedman test produced a p-value of 1.65E-10, and the Iman and Davenport test returned a p-value of 3.72E-11. So, there are significant differences among the methods in the ranking (they are much more significant than those presented in the previous section).

In this case, Bergmann's procedure rejects the following hypotheses with a high level of confidence ($\alpha = 0.05$):

- MA-N11 vs GA
- MA-N11 vs MA-N12
- MA-N11 vs MA-N1O
- MA-N12 vs GA
- MA-N13 vs GA
- MA-N13 vs MA-N1O

Therefore, there is strong statistical evidence that MA-N11 outperforms all the other methods but MA-N13. Also, MA-N12 outperforms GA and MA-N13 is better than both GA and MA-N1O.

With $\alpha = 0.10$, Bergmann's procedure rejects the hypotheses:

- MA-N13 vs MA-N12
- MA-N1O vs GA

So, with less confidence we can conclude that MA-N13 also outperforms MA-N12, and that MA-N1O is better than GA.

Figures 46-51 show for each group of instances with the same size, the error in percentage of the average cost of the solutions, averaged for instances with the same number of operators available.

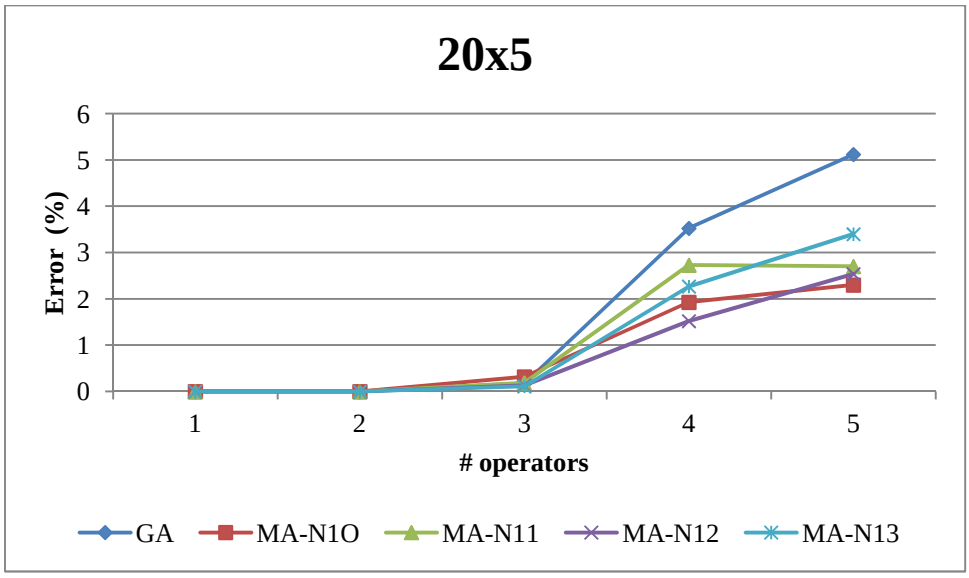


Figure 46. Comparison between the average solution of the considered algorithms (20x5).

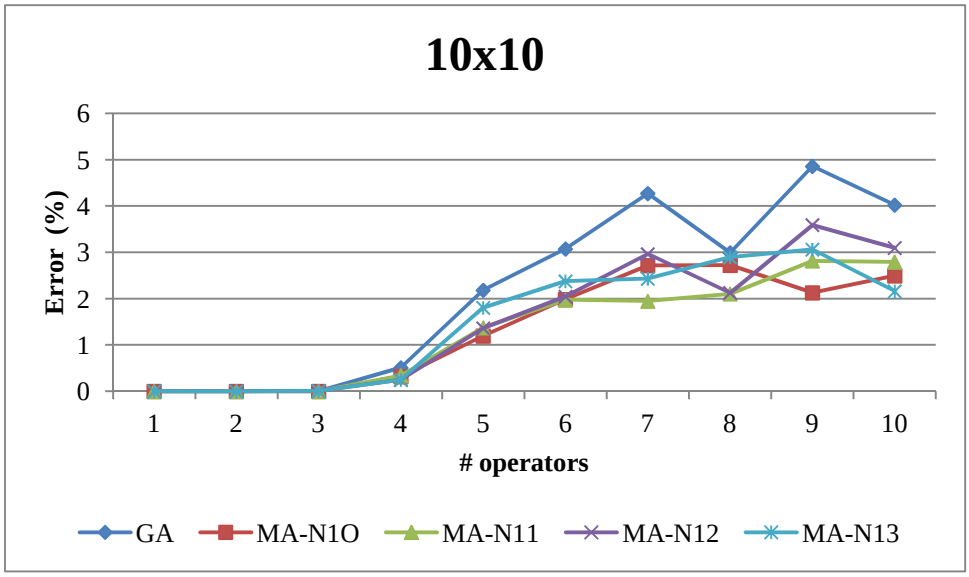


Figure 47. Comparison between the average solution of the considered algorithms (10x10).

For small instances, with up to 100 operations, we can observe that the memetic algorithms achieve best results than with applying the local search method to all the population. In this case, GA never outperforms any of the MAs.

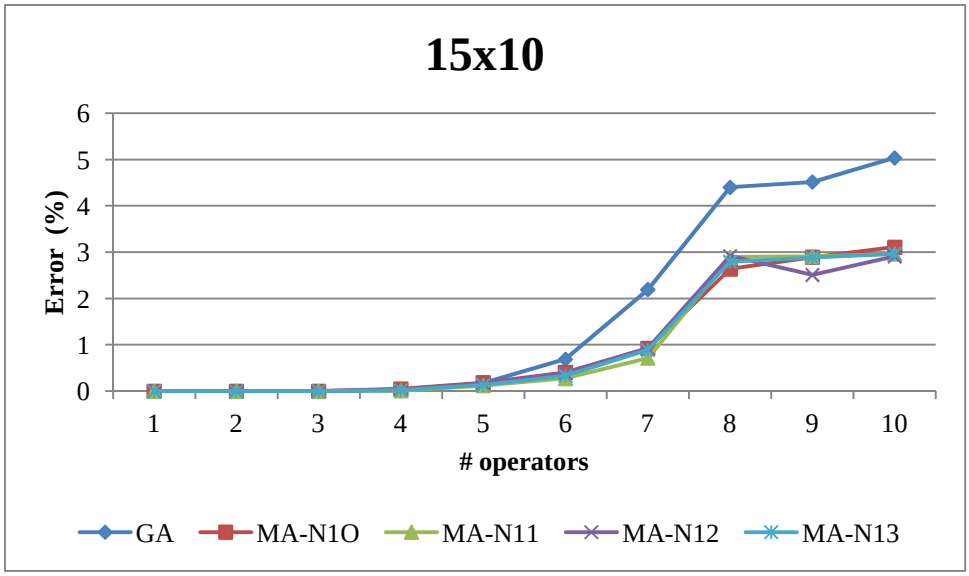


Figure 48. Comparison between the average solution of the considered algorithms (15x10).

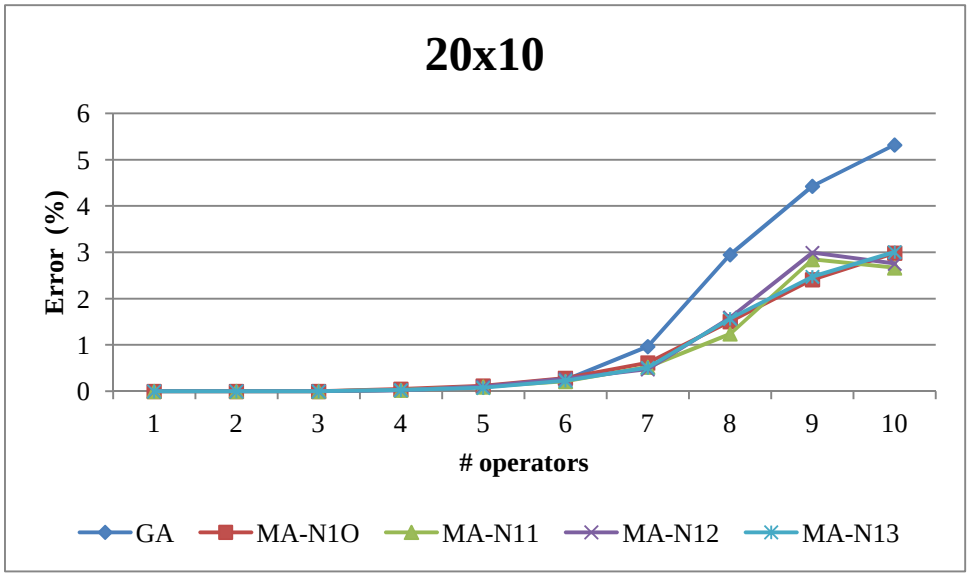


Figure 49. Comparison between the average solution of the considered algorithms (20x10).

For the large instances with sizes 15x10, 20x10, 15x15 and 20x15, the GA is always worse than the MAs, with the exception of the instances with few available operators, for which it obtains the same results. As we can see, the memetic algorithms limiting the number of local searches by introducing a probability of 0.2 of being issued do not show the trend shown in the previous section. Also, there seem to be less differences among the MAs.

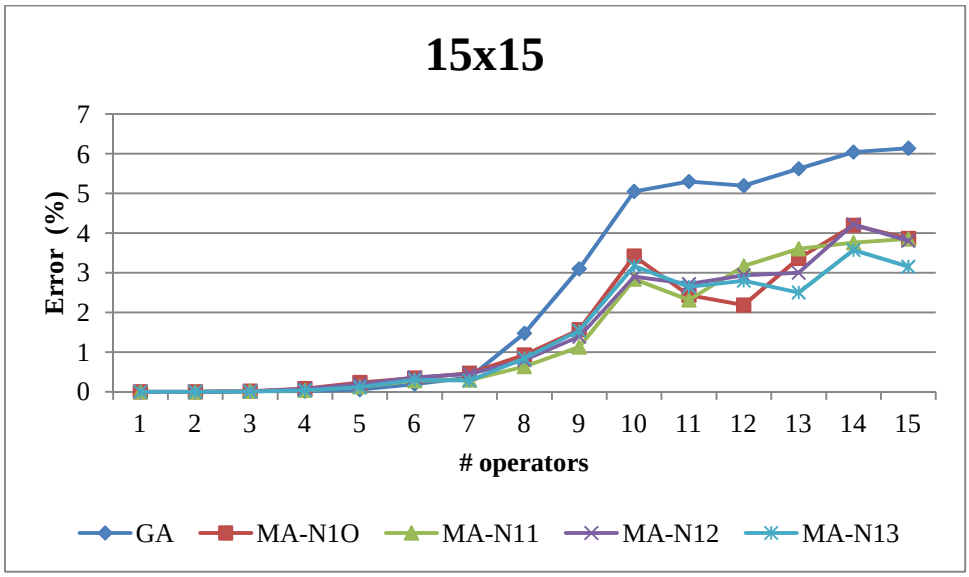


Figure 50. Comparison between the average solution of the considered algorithms (15x15).

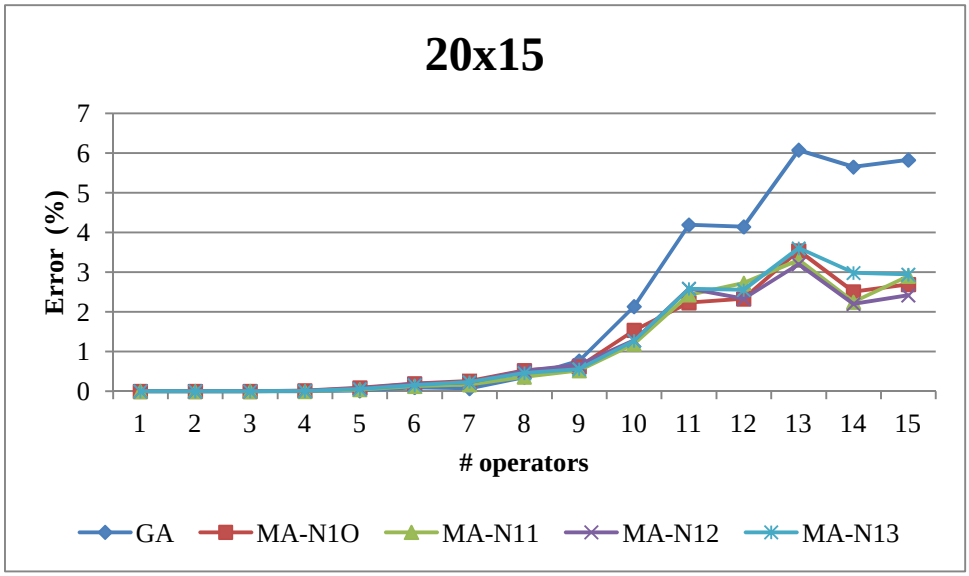


Figure 51. Comparison between the average solution of the considered algorithms (20x15).

Finally, we made a Wilcoxon paired test for comparing MA-N11 issuing a local search from all the chromosomes in the population (MA-N11-ALL) and MA-N11 issuing a local search from each chromosome with a probability of 0.2 (MA-N11-0.2). The alternative hypothesis is that the average error reached by MA-N11-0.2 is less than that reached by MA-N11-ALL. This hypothesis is supported with a p-value of 1.94E-08 by the Wilcoxon test.

So, one important conclusion from these experiments is that it is worthwhile limiting the number of local searches performed along the search.

9. Conclusions and future work

9.1. Conclusions

In this master project we have followed two separate lines of work.

On the one hand,

- We introduced a strategy to improve the OG&T schedule generation scheme used as decoding algorithm in the genetic algorithm for the JSO(n, p) problem with total flow time minimization. This strategy is intended to reduce the search space.
- The new algorithm, termed Hybrid OG&T, was tested across a large number of instances with different sizes, improving the quality of the solutions reached by the simple OG&T algorithm.
- The results supported our expectations: the larger the instance is, the more it is worthwhile to reduce the search space.

On the other hand,

- We defined 4 neighborhood structures, termed N_1^1, N_1^2, N_1^3 and N_1^O for the JSO(n, p) problem with makespan minimization.
- We used these neighborhood structures in a local search method that was combined with the genetic algorithm, defining 4 memetic algorithms MA-N11, MA-N12, MA-N13 and MA-N1O.
- We tested the memetic algorithms issuing a local search from each individual and compared them with the genetic algorithm.
- Although the results from the memetic algorithms were better in average than those from the genetic algorithm, we found out that they reached worse results than the GA for an intermediate number of operators, because of a premature convergence due to the lack of a proper intensification/diversification tradeoff.
- In order to gain diversification, we tested the algorithms again but this time using local search only with a probability of 0.2 for each individual.
- In this case, we have observed that the memetic algorithms clearly outperform the genetic algorithm for any number of available operators, being MA-N11 the best one.

9.2. Future work

As future work we propose to make a thorough experimental study aimed to analyze some issues as the reasons for which premature convergence is produced in some kind of instances. Also, we propose to do some refinements in the neighborhood structures and study the possibility of extending other neighborhoods from the classic job shop problem. Finally, we will try to define new structures based on the properties of the problem derived from the operators.

References

[Agnetis et al. 2011] Agnetis A., Flamini M., Nicosia G. and Pacifici A. A job-shop problem with one additional resource type. *Journal of Scheduling*, 14(3):225-237, 2011.

[Applegate and Cook 1991] Applegate D., Cook W. A computational study of the job-shop scheduling problem. *ORSA Journal of Computing*, 3:149-156, 1991.

[Beasley 1990] Beasley J.E. OR-Library: Distributing test problems by electronic mail. *J. Oper Res Soc*, 41(11):1069-1072, 1990.

[Bierwirth 1992] Bierwirth, C. A generalized permutation approach to job shop scheduling with genetic algorithms. *OR Spectrum* 17:17-92, 1995.

[Brucker and Knust 2006] Complex Scheduling. Srpinger, 2006.

[García et al. 2010] García S., Fernández A., Luengo J. and Herrera F. Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power. *Information Sciences*. 180:2044-2064, 2010.

[Giffler and Thompson 1960] Giffler B. and Thompson G.L. Algorithms for solving production scheduling problems. *Operations Research*, 8:487-503, 1960.

[González et al. 2010] González M.A., R. Vela C., Sierra M.R. and Varela R. Tabu search and genetic algorithm for scheduling with total flow time minimization. In *COPLAS'2010*, pp. 33-41, 2010.

[Mattfeld 1995] Mattfeld D. Evolutionary search and the job shop investigations on genetic algorithms for production scheduling. Springer-Verlag, 1995.

[Mencía 2010] Mencía R. Un algoritmo genético para problemas de scheduling con operarios. Proyecto Fin de Carrera de Ingeniería Técnica en Informática de Sistemas. EPIG. Universidad de Oviedo. (2010)

[Mencía 2011] Mencía R. Un algoritmo memético para un problema de scheduling con operarios. Trabajo fin de Grado de Ingeniería Informática en sistemas de la Información. EPIG. Universidad de Oviedo, 2010.

[Mencía et al. 2011] Mencía R., Sierra M., Mencía C., Varela R. Genetic algorithm for job-shop scheduling with operators. *IWINAC*, (2):305-314, 2011.

[Mencía 2012] Mencía R. Course Project. Evolutionary Computation and Metaheuristics. Master in Soft Computing Intelligent Data Analysis. European Centre for Soft Computing, 2012.

[Roy and Sussman 1964] Roy B., Sussman B. Les problemes d'ordonnancements avec contraintes disjonctives. Notes DS no. 9 bis, SEMA, Paris, 1964.

[Sierra et al. 2011] Sierra M.R., Mencía C. and Varela R. Searching for optimal schedules to the job-shop problem with operators. Technical report. Computing Technologies Group. University of Oviedo, 2011.

[Talbi 2009] Talbi E. Metaheuristics: From Design to Implementation. Wiley Series on Parallel and Distributed Computing. John Wiley & Sons, 2009.

[Vela et al. 2010] Vela, C.R., Varela, R. and González M.A. Local search and genetic algorithm for the job shop scheduling problem with sequence dependent setup times. *Journal of Heuristics*, 16:139-165, 2010.

Appendix: Detailed results

In this appendix we first report, for each instance, the cost of the best solution found along the whole experimental study, both for makespan than for total flow time. Then, we give the results presented in this project in a more detailed way.

Instance	Makespan	TFT	Instance	Makespan	TFT	Instance	Makespan	TFT
ABZ7_10op	756	13489	FT10_2op	2555	15992	LA27_4op	2708	40681
ABZ7_11op	709	12767	FT10_3op	1703	11768	LA27_5op	2168	33178
ABZ7_12op	699	12365	FT10_4op	1296	9646	LA27_6op	1809	28472
ABZ7_13op	695	12224	FT10_5op	1063	8663	LA27_7op	1554	25624
ABZ7_14op	693	12198	FT10_6op	972	7993	LA27_8op	1378	23292
ABZ7_15op	693	12142	FT10_7op	945	7693	LA27_9op	1289	22206
ABZ7_1op	7366	102947	FT10_8op	936	7638	LA29_10op	1209	20513
ABZ7_2op	3683	58069	FT10_9op	930	7602	LA29_1op	9929	130308
ABZ7_3op	2456	39228	FT20_1op	5109	52044	LA29_2op	4965	68862
ABZ7_4op	1842	29832	FT20_2op	2555	30268	LA29_3op	3310	46958
ABZ7_5op	1474	24175	FT20_3op	1703	20568	LA29_4op	2483	36320
ABZ7_6op	1229	20207	FT20_4op	1322	16947	LA29_5op	1987	29348
ABZ7_7op	1056	17596	FT20_5op	1174	15621	LA29_6op	1659	24869
ABZ7_8op	925	15722	LA21_10op	1060	13344	LA29_7op	1431	22974
ABZ7_9op	828	14525	LA21_1op	7994	75875	LA29_8op	1279	21573
ABZ8_10op	777	13855	LA21_2op	3997	40044	LA29_9op	1225	20591
ABZ8_11op	729	13110	LA21_3op	2665	27286	LA38_10op	1257	16675
ABZ8_12op	720	12753	LA21_4op	1999	20918	LA38_11op	1256	16286
ABZ8_13op	703	12663	LA21_5op	1600	17871	LA38_12op	1248	16317
ABZ8_14op	708	12635	LA21_6op	1339	15828	LA38_13op	1236	16210
ABZ8_15op	706	12558	LA21_7op	1168	14535	LA38_14op	1233	16280
ABZ8_1op	7586	112151	LA21_8op	1084	13666	LA38_15op	1237	16288
ABZ8_2op	3793	59214	LA21_9op	1068	13335	LA38_1op	11217	113432
ABZ8_3op	2529	39864	LA24_10op	956	12574	LA38_2op	5609	61936
ABZ8_4op	1897	29847	LA24_1op	7727	76318	LA38_3op	3739	42050
ABZ8_5op	1518	24128	LA24_2op	3864	37752	LA38_4op	2805	32716
ABZ8_6op	1265	20673	LA24_3op	2576	26141	LA38_5op	2245	26490
ABZ8_7op	1087	18012	LA24_4op	1932	20422	LA38_6op	1874	22967
ABZ8_8op	953	16198	LA24_5op	1546	17341	LA38_7op	1618	20346
ABZ8_9op	852	14877	LA24_6op	1292	15495	LA38_8op	1441	18715
ABZ9_10op	773	13576	LA24_7op	1113	14294	LA38_9op	1325	17468
ABZ9_11op	745	12818	LA24_8op	999	13330	LA40_10op	1263	17021
ABZ9_12op	739	12348	LA24_9op	969	12859	LA40_11op	1252	16676
ABZ9_13op	730	12415	LA25_10op	989	12498	LA40_12op	1251	16544
ABZ9_14op	731	12429	LA25_1op	7509	69872	LA40_13op	1252	16528
ABZ9_15op	726	12374	LA25_2op	3755	37166	LA40_14op	1245	16385
ABZ9_1op	7442	109159	LA25_3op	2503	25202	LA40_15op	1237	16513

ABZ9_2op	3721	58654	LA25_4op	1878	20003	LA40_1op	11472	124588
ABZ9_3op	2481	39352	LA25_5op	1503	16559	LA40_2op	5736	64636
ABZ9_4op	1861	29449	LA25_6op	1255	14555	LA40_3op	3824	43660
ABZ9_5op	1489	24254	LA25_7op	1085	13611	LA40_4op	2868	33156
ABZ9_6op	1242	20426	LA25_8op	1006	12976	LA40_5op	2295	27166
ABZ9_7op	1066	17848	LA25_9op	989	12570	LA40_6op	1915	23673
ABZ9_8op	937	15856	LA27_10op	1272	21479	LA40_7op	1651	20986
ABZ9_9op	843	14583	LA27_1op	10832	149848	LA40_8op	1456	19352
FT10_10op	930	7638	LA27_2op	5416	79093	LA40_9op	1330	17886
FT10_1op	5109	28366	LA27_3op	3611	53000			

Table 6. Cost of the best known solution (makespan and total flow time).

	Delta = 0				Delta = 0,25				Delta = 0,5				Delta = 0,75				Delta = 1			
Instances size	**E. Best (%)**	**E. Avg (%)**	**VC (%)**	**t (s)**	**E. Best (%)**	**E. Avg (%)**	**VC (%)**	**t (s)**	**E. Best (%)**	**E. Avg (%)**	**VC (%)**	**t (s)**	**E. Best (%)**	**E. Avg (%)**	**VC (%)**	**t (s)**	**E. Best (%)**	**E. Avg (%)**	**VC (%)**	**t (s)**
10x10	16,73	18,58	0,77	1,41	10,49	12,36	1,02	1,34	4,24	6,89	1,26	1,5	0,7	3,7	1,57	1,33	0,53	4,6	2,13	1,33
15x10	7,53	9,81	1,01	3,2	4,84	7,71	1,17	3,3	2,84	5,86	1,31	3,2	1,1	4,4	1,54	3,01	0,8	4,9	1,74	2,97
15x15	5,29	6,58	0,59	5,19	1,95	3,97	0,88	5,61	1,08	3,29	1,04	5,2	0,5	3,2	1,14	5,52	1,47	4,4	1,28	5,1
20x5	19,18	24,4	2,1	1,33	8,93	14,14	2,38	1,36	4,54	9,72	2,45	1,4	1,6	7,1	2,67	1,31	0,53	6,7	2,89	1,33
20x10	6,89	9	0,85	4,76	4,61	6,97	1,05	4,85	2,38	5,24	1,18	4,6	0,8	4	1,39	4,71	0,82	4,7	1,73	4,53
20x15	2,55	4,15	0,73	9,93	1,5	3,43	0,84	10,01	0,88	3,01	0,96	10	0,5	3,1	1,17	9,74	1,8	4,6	1,26	9,94

Table 7. Comparison between values for the delta parameter.

	GA				$MA\text{-}N_1^O$				$MA\text{-}N_1^1$				$MA\text{-}N_1^2$				$MA\text{-}N_1^3$			
Instances size	**E. Best (%)**	**E. Avg (%)**	**VC (%)**	**t (s)**	**E. Best (%)**	**E. Avg (%)**	**VC (%)**	**t (s)**	**E. Best (%)**	**E. Avg (%)**	**VC (%)**	**t (s)**	**E. Best (%)**	**E. Avg (%)**	**VC (%)**	**t (s)**	**E. Best (%)**	**E. Avg (%)**	**VC (%)**	**t (s)**
20x15	1,22	1,96	0,41	60,1	0,78	1,51	0,45	60	0,56	1,34	0,43	60,02	0,62	1,43	0,47	60	0,79	1,49	0,43	60
10x10	0,80	2,19	0,73	60,11	0,64	1,73	0,69	60	0,29	1,53	0,64	60,01	0,59	1,74	0,67	60	0,45	1,74	0,74	60
20x5	0,99	1,76	0,53	60,1	0,23	0,75	0,43	60	0,20	1,02	0,52	60,01	0,52	1,23	0,48	60	0,38	1,26	0,52	60
15x10	1,06	1,70	0,4	60,1	0,57	1,21	0,42	60	0,43	1,17	0,49	60,01	0,47	1,25	0,45	60	0,57	1,27	0,42	60
20x10	0,87	1,40	0,3	60,1	0,41	0,94	0,36	60	0,39	0,90	0,31	60,01	0,45	1,08	0,4	60	0,58	1,07	0,36	60
15x15	1,55	2,57	0,58	60,1	0,74	1,79	0,6	60	0,71	1,73	0,65	60,01	0,77	1,79	0,62	60	0,68	1,70	0,61	60

Table 8. Comparison between the MAs and the GA with the same computation time.

20x15	GA				$MA\text{-}N_1^0$				$MA\text{-}N_1^1$				$MA\text{-}N_1^2$				$MA\text{-}N_1^3$			
# op	**E. Best (%)**	**E. Avg (%)**	**VC (%)**	**t (s)**	**E. Best (%)**	**E. Avg (%)**	**VC (%)**	**t (s)**	**E. Best (%)**	**E. Avg (%)**	**VC (%)**	**t (s)**	**E. Best (%)**	**E. Avg (%)**	**VC (%)**	**t (s)**	**E. Best (%)**	**E. Avg (%)**	**VC (%)**	**t (s)**
1	0,00	0,00	0,00	60,11	0,00	0,00	0,00	60,01	0,00	0,00	0,00	60,01	0,00	0,00	0,00	60,01	0,00	0,00	0,00	60,01
2	0,00	0,00	0,00	60,10	0,00	0,00	0,00	60,03	0,00	0,00	0,00	60,02	0,00	0,00	0,00	60,03	0,00	0,00	0,00	60,02
3	0,00	0,00	0,00	60,10	0,00	0,00	0,01	60,03	0,00	0,01	0,01	60,02	0,00	0,00	0,01	60,02	0,00	0,00	0,01	60,01
4	0,00	0,00	0,01	60,10	0,02	0,10	0,05	60,02	0,07	0,11	0,03	60,01	0,07	0,10	0,03	60,03	0,05	0,11	0,03	60,01
5	0,00	0,02	0,03	60,10	0,20	0,30	0,05	60,04	0,18	0,27	0,05	60,01	0,13	0,27	0,07	60,03	0,18	0,28	0,05	60,02
6	0,00	0,10	0,06	60,10	0,45	0,56	0,08	60,03	0,35	0,45	0,08	60,02	0,43	0,56	0,08	60,03	0,40	0,51	0,06	60,02
7	0,00	0,07	0,06	60,10	0,56	0,76	0,12	60,02	0,47	0,61	0,10	60,02	0,56	0,74	0,11	60,03	0,56	0,73	0,09	60,02
8	0,07	0,28	0,15	60,10	0,82	1,07	0,17	60,04	0,64	0,84	0,16	60,02	0,82	1,02	0,21	60,02	0,75	0,95	0,18	60,02
9	0,28	0,49	0,28	60,10	0,67	1,00	0,38	60,02	0,79	0,92	0,30	60,02	0,83	1,09	0,31	60,02	0,91	1,08	0,24	60,02
10	0,99	1,43	0,63	60,10	1,47	1,69	0,67	60,03	1,00	1,37	0,63	60,02	1,25	1,45	0,53	60,02	1,34	1,35	0,50	60,02
11	2,61	3,48	0,98	60,10	1,71	2,36	0,80	60,03	0,74	1,82	0,84	60,02	1,48	2,31	1,03	60,02	1,14	2,23	0,91	60,02
12	1,99	4,00	1,08	60,10	0,97	2,63	1,23	60,02	0,69	2,42	1,03	60,02	0,62	2,59	1,25	60,01	1,35	2,93	1,15	60,02
13	4,14	6,08	0,87	60,10	1,50	3,30	1,14	60,02	0,61	3,18	1,13	60,02	1,29	3,21	1,04	60,02	1,50	3,69	1,36	60,02
14	3,95	4,95	0,99	60,10	0,90	2,45	1,07	60,03	1,03	2,38	1,08	60,01	0,85	2,48	1,15	60,02	2,06	2,63	0,87	60,02
15	4,20	4,99	1,02	60,12	2,44	3,07	0,93	60,02	1,76	2,40	0,95	60,02	0,94	2,29	1,28	60,01	1,65	2,44	0,95	60,02

Table 9. Comparison between the MAs and the GA with the same computation time (20x15).

10x10	GA				MA-N_1^O				MA-N_1^1				MA-N_1^2				MA-N_1^3			
# op	E. Best (%)	E. Avg (%)	VC (%)	t (s)	E. Best (%)	E. Avg (%)	VC (%)	t (s)	E. Best (%)	E. Avg (%)	VC (%)	t (s)	E. Best (%)	E. Avg (%)	VC (%)	t (s)	E. Best (%)	E. Avg (%)	VC (%)	t (s)
1	0,00	0,00	0,00	60,13	0,00	0,00	0,00	60,01	0,00	0,00	0,00	60,01	0,00	0,00	0,00	60,01	0,00	0,00	0,00	60,01
2	0,00	0,00	0,00	60,13	0,00	0,00	0,00	60,02	0,00	0,00	0,00	60,01	0,00	0,00	0,00	60,01	0,00	0,00	0,00	60,01
3	0,00	0,00	0,00	60,13	0,00	0,05	0,02	60,01	0,00	0,02	0,03	60,01	0,00	0,04	0,03	60,01	0,00	0,04	0,03	60,01
4	0,08	0,43	0,27	60,10	0,39	0,47	0,13	60,01	0,23	0,39	0,11	60,01	0,15	0,43	0,16	60,01	0,31	0,34	0,09	60,01
5	0,94	1,99	0,98	60,10	0,19	1,95	0,87	60,01	1,13	2,21	0,62	60,01	0,75	2,24	0,95	60,01	1,03	2,11	0,92	60,01
6	1,34	2,23	0,95	60,10	1,75	2,29	1,09	60,01	0,82	1,72	1,31	60,01	1,65	2,35	1,06	60,01	1,03	2,34	1,29	60,01
7	1,80	4,05	1,25	60,10	0,32	1,48	0,98	60,02	0,21	1,99	0,81	60,01	0,74	2,12	1,11	60,02	0,53	2,64	1,30	60,01
8	0,21	2,99	1,42	60,09	1,07	2,90	1,24	60,01	0,00	2,68	1,24	60,01	0,96	3,38	1,15	60,01	0,11	2,93	1,32	60,01
9	2,47	4,86	0,96	60,10	1,51	3,63	1,24	60,01	0,00	2,82	1,05	60,01	0,86	2,84	1,10	60,01	0,86	2,96	1,09	60,01
10	1,18	3,47	1,44	60,12	1,18	2,67	1,30	60,01	0,54	1,59	1,27	60,01	0,75	2,17	1,20	60,01	0,65	2,14	1,40	60,01

Table 10. Comparison between the MAs and the GA with the same computation time (10x10).

20x5	GA				MA-N_1^O				MA-N_1^1				MA-N_1^2				MA-N_1^3			
# op	E. Best (%)	E. Avg (%)	VC (%)	t (s)	E. Best (%)	E. Avg (%)	VC (%)	t (s)	E. Best (%)	E. Avg (%)	VC (%)	t (s)	E. Best (%)	E. Avg (%)	VC (%)	t (s)	E. Best (%)	E. Avg (%)	VC (%)	t (s)
1	0,00	0,00	0,00	60,10	0,00	0,00	0,00	60,01	0,00	0,00	0,00	60,01	0,00	0,00	0,00	60,01	0,00	0,00	0,00	60,01
2	0,00	0,00	0,00	60,10	0,00	0,00	0,00	60,01	0,00	0,00	0,00	60,01	0,00	0,00	0,00	60,01	0,00	0,00	0,00	60,01
3	0,00	0,18	0,17	60,10	0,00	0,07	0,14	60,01	0,00	0,29	0,21	60,01	0,06	0,35	0,18	60,01	0,00	0,28	0,27	60,01
4	1,13	2,67	1,51	60,10	0,83	0,77	0,56	60,01	0,98	1,03	0,52	60,01	1,59	2,46	1,14	60,02	1,21	1,74	0,83	60,01
5	3,83	5,12	0,95	60,10	0,34	2,09	1,45	60,01	0,00	2,92	1,88	60,01	0,94	2,51	1,10	60,01	0,68	3,44	1,52	60,01

Table 11. Comparison between the MAs and the GA with the same computation time (20x5).

15x10	GA				MA-N_1^0				MA-N_1^1				MA-N_1^2				MA-N_1^3			
# op	E. Best (%)	E. Avg (%)	VC (%)	t (s)	E. Best (%)	E. Avg (%)	VC (%)	t (s)	E. Best (%)	E. Avg (%)	VC (%)	t (s)	E. Best (%)	E. Avg (%)	VC (%)	t (s)	E. Best (%)	E. Avg (%)	VC (%)	t (s)
1	0,00	0,00	0,00	60,10	0,00	0,00	0,00	60,01	0,00	0,00	0,00	60,01	0,00	0,00	0,00	60,01	0,00	0,00	0,00	60,01
2	0,00	0,00	0,00	60,10	0,00	0,00	0,00	60,01	0,00	0,00	0,00	60,01	0,00	0,00	0,00	60,02	0,00	0,00	0,00	60,01
3	0,00	0,00	0,00	60,10	0,00	0,02	0,02	60,02	0,00	0,02	0,02	60,01	0,00	0,02	0,02	60,02	0,00	0,03	0,02	60,01
4	0,00	0,01	0,02	60,11	0,07	0,18	0,06	60,02	0,07	0,16	0,05	60,01	0,09	0,18	0,07	60,01	0,07	0,19	0,08	60,01
5	0,04	0,13	0,10	60,11	0,30	0,50	0,12	60,02	0,30	0,43	0,11	60,02	0,34	0,47	0,10	60,01	0,28	0,50	0,14	60,01
6	0,26	0,43	0,27	60,11	0,80	0,83	0,18	60,01	0,56	0,63	0,21	60,02	0,59	0,79	0,28	60,01	0,52	0,67	0,20	60,01
7	1,36	1,72	0,49	60,10	0,89	1,22	0,45	60,02	0,46	1,05	0,62	60,01	1,12	1,43	0,45	60,02	1,18	1,45	0,52	60,02
8	2,79	3,71	1,00	60,11	1,46	2,09	0,79	60,01	1,04	2,32	1,16	60,02	1,38	2,40	1,10	60,02	1,49	2,38	0,93	60,01
9	2,95	4,18	0,92	60,10	1,25	2,49	1,16	60,01	0,86	2,41	1,41	60,01	1,12	2,63	0,94	60,02	1,71	3,12	1,02	60,02
10	3,23	5,04	1,23	60,10	0,88	3,01	1,42	60,01	1,04	2,91	1,36	60,01	0,06	2,76	1,53	60,01	0,42	2,63	1,25	60,01

Table 12. Comparison between the MAs and the GA with the same computation time (15x10).

20x10	GA				MA-N_1^0				MA-N_1^1				MA-N_1^2				MA-N_1^3			
# op	E. Best (%)	E. Avg (%)	VC (%)	t (s)	E. Best (%)	E. Avg (%)	VC (%)	t (s)	E. Best (%)	E. Avg (%)	VC (%)	t (s)	E. Best (%)	E. Avg (%)	VC (%)	t (s)	E. Best (%)	E. Avg (%)	VC (%)	t (s)
1	0,00	0,00	0,00	60,10	0,00	0,00	0,00	60,01	0,00	0,00	0,00	60,02	0,00	0,00	0,00	60,01	0,00	0,00	0,00	60,01
2	0,00	0,00	0,00	60,10	0,00	0,00	0,00	60,02	0,00	0,00	0,00	60,01	0,00	0,00	0,00	60,01	0,00	0,00	0,00	60,01
3	0,00	0,00	0,00	60,09	0,00	0,02	0,02	60,02	0,00	0,01	0,02	60,02	0,00	0,02	0,02	60,01	0,00	0,03	0,01	60,01
4	0,00	0,02	0,02	60,10	0,11	0,17	0,03	60,02	0,06	0,15	0,06	60,01	0,10	0,18	0,05	60,02	0,10	0,18	0,05	60,01
5	0,03	0,06	0,05	60,10	0,24	0,35	0,10	60,02	0,19	0,35	0,10	60,01	0,24	0,36	0,10	60,02	0,17	0,36	0,11	60,01
6	0,00	0,24	0,11	60,10	0,52	0,79	0,14	60,02	0,25	0,57	0,18	60,02	0,46	0,72	0,11	60,02	0,43	0,61	0,10	60,01
7	0,40	0,64	0,29	60,10	0,73	0,90	0,27	60,02	0,49	0,59	0,22	60,01	0,70	0,70	0,19	60,01	0,66	0,74	0,29	60,01
8	1,95	2,36	0,62	60,10	0,72	1,29	0,75	60,02	0,85	1,25	0,76	60,01	0,64	1,80	1,02	60,01	1,20	1,82	0,83	60,01
9	3,00	3,24	0,97	60,10	1,23	1,52	1,34	60,02	1,31	1,87	0,97	60,01	1,19	2,11	1,16	60,02	1,60	2,08	1,23	60,01
10	3,38	5,16	0,96	60,10	0,55	2,07	0,97	60,01	0,74	1,90	0,81	60,01	1,18	2,67	1,31	60,01	1,64	2,60	0,94	60,01

Table 13. Comparison between the MAs and the GA with the same computation time (20x10).

15x15	GA				MA-N_1^0				MA-N_1^1				MA-N_1^2				MA-N_1^3			
# op	**E. Best (%)**	**E. Avg (%)**	**VC (%)**	**t (s)**	**E. Best (%)**	**E. Avg (%)**	**VC (%)**	**t (s)**	**E. Best (%)**	**E. Avg (%)**	**VC (%)**	**t (s)**	**E. Best (%)**	**E. Avg (%)**	**VC (%)**	**t (s)**	**E. Best (%)**	**E. Avg (%)**	**VC (%)**	**t (s)**
1	0,00	0,00	0,00	60,12	0,00	0,00	0,00	60,01	0,00	0,00	0,00	60,01	0,00	0,00	0,00	60,01	0,00	0,00	0,00	60,01
2	0,00	0,00	0,00	60,10	0,00	0,00	0,00	60,02	0,00	0,00	0,00	60,01	0,00	0,00	0,00	60,02	0,00	0,00	0,00	60,01
3	0,00	0,02	0,01	60,10	0,01	0,05	0,02	60,01	0,01	0,05	0,02	60,02	0,00	0,04	0,02	60,02	0,01	0,06	0,03	60,01
4	0,00	0,02	0,02	60,10	0,14	0,22	0,06	60,02	0,19	0,29	0,06	60,01	0,12	0,23	0,07	60,03	0,16	0,24	0,05	60,01
5	0,00	0,06	0,03	60,10	0,53	0,63	0,07	60,03	0,40	0,55	0,11	60,01	0,40	0,60	0,11	60,02	0,44	0,60	0,08	60,01
6	0,03	0,17	0,12	60,10	0,77	1,01	0,17	60,02	0,48	0,87	0,19	60,01	0,79	0,93	0,12	60,01	0,69	0,97	0,17	60,02
7	0,00	0,36	0,30	60,10	1,01	1,43	0,25	60,03	0,80	1,14	0,24	60,02	0,98	1,28	0,20	60,02	0,98	1,21	0,19	60,02
8	0,90	0,64	0,36	60,11	1,07	1,07	0,51	60,02	0,97	0,69	0,35	60,02	1,62	1,18	0,31	60,02	1,52	1,05	0,28	60,02
9	1,32	2,52	0,92	60,09	0,90	1,85	0,76	60,01	1,43	1,48	0,78	60,01	1,20	1,63	0,55	60,02	0,79	1,32	0,64	60,02
10	2,98	4,84	1,19	60,11	0,87	3,07	1,27	60,01	0,80	2,65	1,27	60,02	2,26	3,81	1,21	60,02	1,55	3,41	1,55	60,01
11	3,67	5,18	1,05	60,10	1,16	2,96	1,02	60,03	0,12	2,64	1,45	60,01	0,84	2,98	1,16	60,02	0,68	2,98	1,18	60,01
12	3,12	5,19	1,28	60,10	0,00	2,65	1,32	60,02	1,04	3,26	1,62	60,02	1,16	2,85	1,11	60,01	0,28	2,37	1,62	60,01
13	2,94	5,62	1,37	60,10	1,77	3,27	1,08	60,01	1,01	3,26	1,26	60,02	0,28	2,35	1,36	60,01	0,08	2,71	1,10	60,02
14	3,92	5,53	1,05	60,11	1,38	2,79	1,17	60,02	1,21	2,76	1,30	60,01	0,53	2,97	1,63	60,01	1,62	2,87	0,99	60,02
15	4,37	6,14	0,93	60,11	1,46	3,58	1,32	60,01	2,14	4,11	1,03	60,01	1,33	3,72	1,48	60,01	1,46	3,48	1,25	60,01

Table 14. Comparison between the MAs and the GA with the same computation time (15x15).

	GA				$MA\text{-}N_1^O$				$MA\text{-}N_1^1$				$MA\text{-}N_1^2$				$MA\text{-}N_1^3$			
Instances size	**E. Best (%)**	**E. Avg (%)**	**VC (%)**	**t (s)**	**E. Best (%)**	**E. Avg (%)**	**VC (%)**	**t (s)**	**E. Best (%)**	**E. Avg (%)**	**VC (%)**	**t (s)**	**E. Best (%)**	**E. Avg (%)**	**VC (%)**	**t (s)**	**E. Best (%)**	**E. Avg (%)**	**VC (%)**	**t (s)**
20x15	1,22	1,96	0,41	60,1	0,35	1,10	3,6	60	0,38	1,07	3,33	60	0,27	1,05	3,53	60,01	0,37	1,16	3,34	60
10x10	0,80	2,19	0,73	60,11	0,22	1,36	6,79	60	0,29	1,34	6,41	60	0,30	1,54	7,72	60,01	0,24	1,50	7,43	60
20x5	0,99	1,76	0,53	60,1	0,28	0,91	6,35	60	0,48	1,12	5,03	60	0,10	0,84	6,5	60,01	0,27	1,16	7,49	60
15x10	1,06	1,70	0,4	60,1	0,22	1,02	5,23	60	0,27	0,99	4,7	60	0,27	0,99	4,29	60,01	0,34	1,00	4,44	60
20x10	0,87	1,40	0,3	60,1	0,37	0,80	3,94	60	0,13	0,76	4,69	60	0,30	0,82	4,78	60,01	0,28	0,79	4,79	60
15x15	1,55	2,57	0,58	60,1	0,56	1,54	7,4	60	0,58	1,47	7,55	60	0,46	1,52	8,85	60,01	0,35	1,40	8,5	60

Table 15. Comparison between the MAs and the GA with the same computation time and 20% Local Search.

20x15	GA				$MA\text{-}N_1^O$				$MA\text{-}N_1^1$				$MA\text{-}N_1^2$				$MA\text{-}N_1^3$			
# op	**E. Best (%)**	**E. Avg (%)**	**VC (%)**	**t (s)**	**E. Best (%)**	**E. Avg (%)**	**VC (%)**	**t (s)**	**E. Best (%)**	**E. Avg (%)**	**VC (%)**	**t (s)**	**E. Best (%)**	**E. Avg (%)**	**VC (%)**	**t (s)**	**E. Best (%)**	**E. Avg (%)**	**VC (%)**	**t (s)**
1	0,00	0,00	0,00	60,11	0,00	0,00	0,00	60,01	0,00	0,00	0,00	60,01	0,00	0,00	0,00	60,01	0,00	0,00	0,00	60,01
2	0,00	0,00	0,00	60,10	0,00	0,00	0,00	60,01	0,00	0,00	0,00	60,01	0,00	0,00	0,00	60,01	0,00	0,00	0,00	60,01
3	0,00	0,00	0,00	60,10	0,00	0,00	0,00	60,01	0,00	0,00	0,00	60,01	0,00	0,00	0,00	60,01	0,00	0,00	0,00	60,01
4	0,00	0,00	0,01	60,10	0,00	0,01	0,40	60,02	0,00	0,00	0,00	60,01	0,00	0,01	0,39	60,01	0,00	0,01	0,23	60,01
5	0,00	0,02	0,03	60,10	0,02	0,09	0,66	60,01	0,00	0,04	0,48	60,01	0,00	0,07	0,68	60,01	0,00	0,05	0,45	60,01
6	0,00	0,10	0,06	60,10	0,11	0,20	0,83	60,02	0,05	0,13	0,73	60,01	0,11	0,18	0,76	60,02	0,05	0,15	0,77	60,01
7	0,00	0,07	0,06	60,10	0,12	0,26	0,88	60,01	0,03	0,17	1,11	60,01	0,03	0,22	1,17	60,02	0,12	0,24	0,78	60,02
8	0,07	0,35	0,15	60,10	0,28	0,52	1,62	60,02	0,18	0,37	1,17	60,02	0,28	0,51	1,28	60,01	0,14	0,46	1,47	60,02
9	0,28	0,77	0,28	60,10	0,28	0,62	2,22	60,02	0,00	0,52	3,22	60,01	0,20	0,66	2,59	60,02	0,28	0,56	1,49	60,02
10	0,99	2,14	0,63	60,10	0,56	1,54	5,24	60,02	0,44	1,19	4,11	60,01	0,47	1,28	3,97	60,01	0,22	1,28	4,24	60,02
11	2,61	4,20	0,98	60,10	0,37	2,24	8,52	60,01	0,68	2,43	9,01	60,02	0,90	2,59	7,04	60,01	0,96	2,58	7,71	60,02
12	1,99	4,15	1,08	60,10	0,32	2,33	8,86	60,02	0,83	2,73	8,47	60,02	0,73	2,33	8,02	60,01	0,75	2,55	8,18	60,01
13	4,14	6,08	0,87	60,10	1,38	3,54	9,23	60,01	1,47	3,30	8,08	60,02	1,04	3,20	9,00	60,01	1,22	3,60	8,42	60,01

14	3,95	5,65	0,99	60,10	0,95	2,51	8,02	60,01	0,71	2,25	6,90	60,01	0,19	2,20	9,64	60,02	0,51	2,98	8,69	60,02
15	4,20	5,83	1,02	60,12	0,84	2,70	7,54	60,02	1,36	2,90	6,70	60,02	0,10	2,42	8,36	60,01	1,27	2,94	7,64	60,01

Table 16. Comparison between the MAs and the GA with the same computation time and 20% Local Search (20x15).

10x10	**GA**				**MA-N_1^0**				**MA-N_1^1**				**MA-N_1^2**				**MA-N_1^3**			
# op	**E. Best (%)**	**E. Avg (%)**	**VC (%)**	**t (s)**	**E. Best (%)**	**E. Avg (%)**	**VC (%)**	**t (s)**	**E. Best (%)**	**E. Avg (%)**	**VC (%)**	**t (s)**	**E. Best (%)**	**E. Avg (%)**	**VC (%)**	**t (s)**	**E. Best (%)**	**E. Avg (%)**	**VC (%)**	**t (s)**
1	0,00	0,00	0,00	60,13	0,00	0,00	0,00	60,01	0,00	0,00	0,00	60,01	0,00	0,00	0,00	60,01	0,00	0,00	0,00	60,01
2	0,00	0,00	0,00	60,13	0,00	0,00	0,00	60,01	0,00	0,00	0,00	60,01	0,00	0,00	0,00	60,01	0,00	0,00	0,00	60,01
3	0,00	0,00	0,00	60,13	0,00	0,00	0,00	60,01	0,00	0,00	0,00	60,02	0,00	0,01	0,30	60,01	0,00	0,01	0,30	60,01
4	0,08	0,51	0,27	60,10	0,15	0,32	0,83	60,01	0,23	0,34	0,92	60,01	0,00	0,25	1,54	60,01	0,00	0,25	1,72	60,01
5	0,94	2,18	0,98	60,10	0,19	1,19	7,99	60,01	0,38	1,37	7,90	60,01	0,00	1,36	8,50	60,01	0,47	1,81	10,06	60,01
6	1,34	3,08	0,95	60,10	0,31	1,99	10,51	60,02	0,62	1,98	9,35	60,01	0,93	2,05	8,55	60,01	0,00	2,38	11,81	60,01
7	1,80	4,28	1,25	60,10	0,00	2,72	14,44	60,02	0,00	1,95	11,54	60,01	1,06	2,96	15,48	60,01	0,85	2,43	9,56	60,01
8	0,21	2,99	1,42	60,09	0,11	2,72	11,53	60,01	0,11	2,10	10,55	60,01	0,21	2,13	11,76	60,01	0,32	2,90	12,57	60,01
9	2,47	4,86	0,96	60,10	0,65	2,13	11,17	60,01	0,86	2,82	11,77	60,01	0,00	3,59	16,92	60,01	0,75	3,06	13,12	60,01
10	1,18	4,02	1,44	60,12	0,75	2,49	11,45	60,01	0,75	2,80	12,12	60,01	0,75	3,10	14,14	60,01	0,00	2,16	15,18	60,01

Table 17. Comparison between the MAs and the GA with the same computation time and 20% Local Search (10x10).

20x5	**GA**				**MA-N_1^0**				**MA-N_1^1**				**MA-N_1^2**				**MA-N_1^3**			
# op	**E. Best (%)**	**E. Avg (%)**	**VC (%)**	**t (s)**	**E. Best (%)**	**E. Avg (%)**	**VC (%)**	**t (s)**	**E. Best (%)**	**E. Avg (%)**	**VC (%)**	**t (s)**	**E. Best (%)**	**E. Avg (%)**	**VC (%)**	**t (s)**	**E. Best (%)**	**E. Avg (%)**	**VC (%)**	**t (s)**
1	0,00	0,00	0,00	60,10	0,00	0,00	0,00	60,01	0,00	0,00	0,00	60,01	0,00	0,00	0,00	60,01	0,00	0,00	0,00	60,01
2	0,00	0,00	0,00	60,10	0,00	0,00	0,00	60,01	0,00	0,00	0,00	60,01	0,00	0,00	0,00	60,01	0,00	0,00	0,00	60,01
3	0,00	0,18	0,17	60,10	0,00	0,32	3,44	60,01	0,00	0,19	3,43	60,01	0,00	0,13	3,25	60,01	0,00	0,11	3,11	60,01
4	1,13	3,52	1,51	60,10	1,06	1,93	11,95	60,02	1,06	2,73	12,34	60,01	0,00	1,52	16,16	60,01	0,83	2,27	11,30	60,01
5	3,83	5,12	0,95	60,10	0,34	2,30	16,38	60,01	1,36	2,70	9,40	60,01	0,51	2,54	13,10	60,01	0,51	3,40	23,03	60,01

Table 18. Comparison between the MAs and the GA with the same computation time and 20% Local Search (20x5).

15x10	GA				$MA\text{-}N_1^O$				$MA\text{-}N_1^1$				$MA\text{-}N_1^2$				$MA\text{-}N_1^3$			
# op	**E. Best (%)**	**E. Avg (%)**	**VC (%)**	**t (s)**	**E. Best (%)**	**E. Avg (%)**	**VC (%)**	**t (s)**	**E. Best (%)**	**E. Avg (%)**	**VC (%)**	**t (s)**	**E. Best (%)**	**E. Avg (%)**	**VC (%)**	**t (s)**	**E. Best (%)**	**E. Avg (%)**	**VC (%)**	**t (s)**
1	0,00	0,00	0,00	60,10	0,00	0,00	0,00	60,01	0,00	0,00	0,00	60,01	0,00	0,00	0,00	60,01	0,00	0,00	0,00	60,01
2	0,00	0,00	0,00	60,10	0,00	0,00	0,00	60,01	0,00	0,00	0,00	60,01	0,00	0,00	0,00	60,01	0,00	0,00	0,00	60,01
3	0,00	0,00	0,00	60,10	0,00	0,00	0,00	60,01	0,00	0,00	0,00	60,01	0,00	0,00	0,00	60,01	0,00	0,00	0,20	60,01
4	0,00	0,01	0,02	60,11	0,00	0,05	0,67	60,01	0,00	0,01	0,29	60,01	0,00	0,03	0,56	60,01	0,00	0,02	0,41	60,01
5	0,04	0,17	0,10	60,11	0,04	0,18	1,31	60,01	0,00	0,12	1,42	60,01	0,04	0,16	1,35	60,01	0,04	0,13	0,99	60,01
6	0,26	0,69	0,27	60,11	0,08	0,40	2,49	60,01	0,03	0,27	2,02	60,01	0,05	0,39	2,57	60,01	0,13	0,33	1,76	60,01
7	1,36	2,19	0,49	60,10	0,18	0,92	5,43	60,01	0,18	0,71	4,24	60,01	0,15	0,93	6,09	60,01	0,33	0,89	4,18	60,01
8	2,79	4,40	1,00	60,11	0,31	2,64	13,80	60,01	1,25	2,90	11,59	60,01	0,94	2,92	9,80	60,01	1,38	2,80	11,61	60,01
9	2,95	4,52	0,92	60,10	0,82	2,89	13,10	60,01	0,34	2,91	14,92	60,01	0,45	2,51	11,54	60,01	0,69	2,88	12,90	60,01
10	3,23	5,04	1,23	60,10	0,77	3,11	15,49	60,01	0,93	2,96	12,53	60,01	1,12	2,91	10,97	60,01	0,84	2,97	12,40	60,02

Table 19. Comparison between the MAs and the GA with the same computation time and 20% Local Search (15x10).

20x10	GA				$MA\text{-}N_1^O$				$MA\text{-}N_1^1$				$MA\text{-}N_1^2$				$MA\text{-}N_1^3$			
# op	**E. Best (%)**	**E. Avg (%)**	**VC (%)**	**t (s)**	**E. Best (%)**	**E. Avg (%)**	**VC (%)**	**t (s)**	**E. Best (%)**	**E. Avg (%)**	**VC (%)**	**t (s)**	**E. Best (%)**	**E. Avg (%)**	**VC (%)**	**t (s)**	**E. Best (%)**	**E. Avg (%)**	**VC (%)**	**t (s)**
1	0,00	0,00	0,00	60,10	0,00	0,00	0,00	60,01	0,00	0,00	0,00	60,01	0,00	0,00	0,00	60,01	0,00	0,00	0,00	60,01
2	0,00	0,00	0,00	60,10	0,00	0,00	0,00	60,01	0,00	0,00	0,00	60,01	0,00	0,00	0,00	60,02	0,00	0,00	0,00	60,01
3	0,00	0,00	0,00	60,09	0,00	0,00	0,15	60,01	0,00	0,00	0,00	60,01	0,00	0,00	0,00	60,01	0,00	0,00	0,00	60,01
4	0,00	0,02	0,02	60,10	0,02	0,05	0,62	60,01	0,00	0,03	0,69	60,01	0,00	0,02	0,45	60,02	0,00	0,03	0,50	60,01
5	0,03	0,09	0,05	60,10	0,05	0,12	1,30	60,01	0,00	0,10	1,01	60,01	0,05	0,11	1,07	60,01	0,00	0,07	1,07	60,01
6	0,00	0,24	0,11	60,10	0,15	0,29	1,85	60,01	0,03	0,22	1,85	60,01	0,06	0,27	2,17	60,01	0,03	0,23	1,97	60,01
7	0,40	0,97	0,29	60,10	0,23	0,61	4,12	60,01	0,23	0,52	2,86	60,01	0,19	0,48	2,77	60,01	0,07	0,51	3,73	60,01
8	1,95	2,95	0,62	60,10	0,63	1,51	7,10	60,01	0,04	1,24	9,80	60,01	0,51	1,59	11,40	60,01	0,95	1,57	7,85	60,01
9	3,00	4,43	0,97	60,10	1,17	2,41	11,76	60,02	0,16	2,85	18,03	60,01	1,32	2,99	17,14	60,01	0,78	2,48	13,97	60,01
10	3,38	5,32	0,96	60,10	1,42	2,98	12,52	60,01	0,82	2,67	12,70	60,01	0,91	2,76	12,79	60,01	0,97	3,01	18,84	60,01

Table 20. Comparison between the MAs and the GA with the same computation time and 20% Local Search (20x10).

15x15	GA				$MA\text{-}N_1^0$				$MA\text{-}N_1^1$				$MA\text{-}N_1^2$				$MA\text{-}N_1^3$			
# op	**E. Best (%)**	**E. Avg (%)**	**VC (%)**	**t (s)**	**E. Best (%)**	**E. Avg (%)**	**VC (%)**	**t (s)**	**E. Best (%)**	**E. Avg (%)**	**VC (%)**	**t (s)**	**E. Best (%)**	**E. Avg (%)**	**VC (%)**	**t (s)**	**E. Best (%)**	**E. Avg (%)**	**VC (%)**	**t (s)**
1	0,00	0,00	0,00	60,12	0,00	0,00	0,00	60,01	0,00	0,00	0,00	60,01	0,00	0,00	0,00	60,01	0,00	0,00	0,00	60,01
2	0,00	0,00	0,00	60,10	0,00	0,00	0,00	60,01	0,00	0,00	0,00	60,01	0,00	0,00	0,00	60,01	0,00	0,00	0,00	60,01
3	0,00	0,02	0,01	60,10	0,00	0,02	0,43	60,01	0,00	0,02	0,47	60,01	0,00	0,01	0,43	60,01	0,00	0,01	0,43	60,01
4	0,00	0,02	0,02	60,10	0,04	0,08	0,98	60,01	0,00	0,04	0,62	60,01	0,02	0,07	1,30	60,01	0,02	0,04	0,49	60,01
5	0,00	0,06	0,03	60,10	0,11	0,23	1,75	60,01	0,07	0,13	0,85	60,02	0,07	0,20	2,01	60,01	0,02	0,12	1,45	60,01
6	0,03	0,20	0,12	60,10	0,11	0,35	2,41	60,01	0,16	0,28	1,63	60,01	0,05	0,36	2,72	60,01	0,11	0,31	2,15	60,01
7	0,00	0,36	0,30	60,10	0,12	0,47	2,79	60,01	0,03	0,30	3,32	60,01	0,09	0,45	3,19	60,01	0,03	0,28	2,69	60,01
8	0,90	1,48	0,36	60,11	0,41	0,93	5,35	60,01	0,10	0,64	5,22	60,01	0,14	0,80	6,84	60,02	0,00	0,85	6,39	60,01
9	1,32	3,10	0,92	60,09	0,38	1,56	9,15	60,01	0,00	1,13	11,72	60,01	0,45	1,39	9,51	60,01	0,11	1,54	12,43	60,01
10	2,98	5,05	1,19	60,11	1,79	3,42	12,16	60,01	1,11	2,84	14,91	60,01	1,03	2,90	14,48	60,01	0,79	3,17	21,26	60,01
11	3,67	5,30	1,05	60,10	0,48	2,43	12,95	60,01	1,16	2,31	10,06	60,01	0,80	2,72	15,95	60,01	0,68	2,64	15,26	60,01
12	3,12	5,19	1,28	60,10	0,40	2,19	15,31	60,01	1,28	3,17	11,81	60,01	1,00	2,94	17,64	60,01	0,80	2,80	17,61	60,01
13	2,94	5,62	1,37	60,10	1,53	3,37	12,52	60,01	1,73	3,60	18,77	60,01	0,12	3,00	21,30	60,01	0,36	2,50	16,53	60,01
14	3,92	6,04	1,05	60,11	1,46	4,20	16,95	60,01	1,37	3,76	17,11	60,02	1,33	4,22	16,39	60,01	1,50	3,57	15,55	60,01
15	4,37	6,14	0,93	60,11	1,54	3,86	18,19	60,01	1,66	3,85	16,74	60,01	1,74	3,80	21,01	60,01	0,77	3,15	15,30	60,01

Table 21. Comparison between the MAs and the GA with the same computation time and 20% Local Search (15x15).