A Hybrid Variable Neighborhood Search Algorithm for the Flexible Open Shop Scheduling Problem with Travel Times

Vanessa Londoño -200726003

Industrial Engineering, Universidad de los Andes, Bogotá, Colombia

E-mail: v.londono966@uniandes.edu.co

A Hybrid Variable Neighborhood Search Algorithm for the Flexible Open Shop Scheduling Problem with Travel Times

Londoño, Vanessa[1]; Mejía, Gonzalo[1]; Casallas, Rubby[2]; Romero, Jaime[2]
Universidad de Los Andes
[1]Department of Industrial Engineering PYLO Research Group
[2]Department of Systems and Computer Engineering, TICSw Research Group

Abstract

This paper studies a Flexible Open Shop Scheduling Problem (FOSSP) which consists of determining the schedule of n jobs that visit m workstations which have identical parallel machines. Additionally, travel times between workstations are considered. The objective is to minimize total flow time. The FOSSP is formulated as a Mixed Integer Program (MIP) and due to its NP-Hard nature a Variable Neighborhood Search (VNS) algorithm is proposed and tested. The VNS algorithm is also hybridized with Tabu Search and with Simulated Annealing. The results are compared against the optimal solutions when available, or against proposed lower bounds and previous work. The results show very good solutions in short computational times for the studied instances improving the results of previous work.

Keywords: Flexible Open Shop, Variable Neighborhood Search, Tabu Search, Simulated Annealing, Travel Times.

1. Introduction

The classical Flexible Open Shop Scheduling Problem (FOSSP) can be defined as a problem where n jobs have to be scheduled on m workstations with identical or non identical parallel machines. In this particular case, identical parallel machines are considered. In an Open Shop problem, in addition to the usual machine capacity constraints, the routes of the jobs are not known. Therefore, both, the route of each job and the sequence at each machine in a workstation have to be determined (Pinedo 2007). Additionally, the studied problem takes into account a constraint of travel times between workstations. The objective function is to minimize the total flow time which is the total time that all jobs spend in the system including waiting, travel and

processing times. This way, both circulation time of jobs throughout the system and queue times are minimized.

This paper describes an algorithm to solve a FOSSP considering the aforementioned service constraint and travel times between workstations. In general, there are two different but closely related problems: (i) deciding the route that each job should follow and (ii) establishing the sequence in which all jobs should be processed at the workstations. For the purpose of this work, all jobs start at a common origin (depot), visit all workstations and return to the depot. The workstation processing times and the travel times between the workstations and from/to the depot are known as well as the capacity of the workstations in terms of the number of jobs that can be processed simultaneously. In Graham et al. (1979) notation this problem can be represented as a $FO_m/TT_{il}/\sum C_{j0}$where $TT_{il}$are the travel times between workstations i and l and $\sum C_{j0}$ is the total flow time.

Such a problem can be applied to a manufacturing plant where different jobs have to visit all the workstations and can be processed in different machines. However, it can also be applied to a vehicle scheduling and routing with travel times. For example, the vehicles could be treated as jobs while the workstations are the warehouses. The different loading or unloading docks could be treated as the parallel machines in each warehouse. Hence, the route that each truck should follow and the sequence in which all trucks should be served at the warehouses could be established.

An existing framework that was developed for previous work by Mejia et al. (2013) and by Cendales et al. (2013) was modified and extended to solve the problem described above. This approach is part of a larger project intended to provide different solution approaches for open-shop scheduling problems.

The following sections of the paper are organized as follows. Section 2 presents the literature review. Section 3 introduces a Mixed Integer Program formulation and the calculation of a lower bound. The proposed Variable Neighborhood Search algorithm is explained in section 4. Execution results and comparisons with lower bounds and with previous work are shown in section 5. Finally, conclusions and recommendations for future work are given in section 6.

2. Literature review

The Flexible Open Shop Scheduling Problem (FOSSP) is the general case of the Open Shop Scheduling Problem (OSSP) allowing the utilization of parallel machines in the different workstations. The OSSP is known to be NP-Hard when the number of stages is greater than 2 (Graham et al. 1979) and so is the FOSSP (Lawler et al. 1982, Chen et al. 1993, Schuurman & Woeginger 1999, Sevastianov & Woeginger 2001, Matta 2009, Naderi et al. 2011).

In literature the OSSP has received more attention from academics than the FOSSP. Different solution approaches have been explored over the years for the OSSP. Initially, several authors focused their attention in the development of polynomial time algorithms (Brucker 1997, Jurcik 2009). Later on, other researchers explored exact solution methods such as Branch and Bound algorithms (Brucker et al. 1997, Liaw 2005, Zobolas et al. 2009), Constraint Programming (Malapert et al. 2012) and Linear Programming (Hashemi 2010). Due to the complexity of such problems, the utilization of heuristic and meta-heuristic solution methods such as Genetic Algorithms (Ahmadizar & Farahani 2012, Andersen et al. 2008, Low & Yehh 2009, Liaw 2000), Simulated Annealing (Andersen et al. 2008, Roshanaei et al. 2010, Yu et al. 2010), constructive heuristic algorithms (Naderi et al. 2010), Tabu Search (Andersen et al. 2008, Alcaide et al. 1997, Liaw 2003), Ant Colony Optimization and Particle Swarm

algorithms (Blum 2005, Lin & Sha 2011,Yang et al. 2006, Sha & Hsu 2008, Panahi & Tavakkoli-Moghaddam 2011), Variable Neighborhood Search (Zobolas et al. 2009), among others, has been widely explored by researchers in recent years.

The research on the specific problem of this paper, the FOSSP, has been limited due to the fact that the study of scheduling problems involving parallel machines is more common in Flow and Job Shop frameworks than in Open Shops (Lawler et al. 1982, Matta 2009, Naderi et al. 2011). Regarding the reviewed literature related to flexible Flow and Job shops, some approaches have been Gravitational Search algorithm and Colored Petri Net (Barzegar et al. 2012), Large Neighborhood Search and Adaptive Randomize Decomposition (Pacino & Hentenryck 2011), Particle Swarm Optimization, Simulated Annealing and Tabu Search (Li et al. 2010), among others. However, a very commonly used approach when considering parallel machines in the Flow or Job shops has been the Variable Neighborhood Search (VNS) meta-heuristic. In some papers VNS is solely used (Cheng et al. 2012, Davidovic et al. 2001) while in others it is hybridized with other heuristics such as Genetic Algorithm (Li et al. 2010) and Particle Swarm Optimization (Chen et al. 2013, Liu et al. 2007). In all cases, VNS by itself or hybridized appeared to perform better than other approaches.

Among the few studies of the FOSSP, some researches aimed at the utilization of polynomial time algorithms such as Lawler et al. (1982) who proposed a polynomial time algorithm to minimize makespan with preemptions for three specific cases with identical and non identical parallel machines. For the non-preemptive FOSSP, some researchers have focused their attention in the development of polynomial time approximation algorithms (Lawler et al. 1982, Matta 2009, Naderi et al. 2010). Particularly, Chen et al. (1993) and Schuurman & Woeginger (1999) studied different scenarios for minimizing the makespan in a 2-stage non-preemptive FOSSP considering

different configurations related to the number of machines in each stage. Jansen et al. (2000) proposed a polynomial time approximation scheme using dynamic programming. Also, an approximation scheme based on a Greedy Algorithm to minimize the makespan was proposed by Sevastianov & Woeginger (2001).

Given the complexity of the FOSSP, several heuristic and meta-heuristic approaches have also been explored. Naderi et al. (2010) propose four constructive heuristics for such a problem, as well as Jurcik (2009) who proposes different algorithms to minimize the makespan. For the Multiprocessor Open Shop, Matta (2009) proposes a Genetic Algorithm to minimize the makespan. Finally, Naderi et al. (2011) propose a mathematical formulation and a meta-heuristic based on the Memetic Algorithm and Simulated Annealing for the FOSSP to minimize the total flow time.

This paper intends to study the FOSSP taking into account the travel times between workstations. However, none of the previous revised work considers both the FOSSP and travel times. Regarding the travel times between workstations, Low et al. (2009) hybridize the Genetic Algorithm with Simulated Annealing and with Tabu Search to minimize the total tardiness in an Open Shop problem considering travel times, removal times and setup times. Also, Yu et al. (2010) present a simulated annealing algorithm to minimize the makespan in an Open Shop with travel times. Within the literature review, the only papers that consider travel times between stations for the FOSSP available for reference are the ones developed by Cendales et al. (2013), which implements Simulated Annealing to solve the FOSSP considering travel times, and by Mejia et al. (2013), where Tabu Search and Simulated Annealing are implemented to solve the same problem.

3. Problem formulation and lower bound

This section introduces a MIP formulation to solve the FOSSP with travel times with the objective of minimizing the total flow time ($FO_m/TT_{il}/\sum C_{j0}$). A lower bound (LB) proposed by Mejía et al. (2013) is presented as well with the purpose of evaluating the performance of the later on proposed algorithm.

3.1. MIP formulation

The formulation of the MIP was adapted from the MIP models developed by Yu et al. (2010) and Naderi et al. (2010) and it is the same formulation used in previous work (Mejia et al. 2013, Cendales et al. 2013). The formulation is as follows.

Parameters:

- $n: number\ of\ jobs$
- $m: number\ of\ workstations$
- $m_i: number\ of\ identical\ machines\ in\ workstation\ i\ (indexed\ in\ r)$
- $p_{ji}: processing\ time\ of\ job\ j\ in\ workstation\ i$
- $TT_{il}: travel\ time\ from\ workstation\ i\ to\ workstation\ l$
- $M: a\ large\ positive\ number$

Sets:

- $J = \{1, \dots n\}\ set\ of\ jobs\ (indexed\ in\ j\ and\ k)$
- $I = \{1, \dots, m\}\ set\ of\ workstations\ (indexed\ in\ i\ and\ l)$

Variables:

- $C_{ij}: completion\ time\ of\ job\ j\ in\ workstation\ i$
- $x_{jil} = \begin{cases} 1\ if\ operation\ O_{ji}\ is\ processed\ after\ O_{jl} \\ 0\ otherwise \end{cases}$
- $y_{jik} = \begin{cases} 1\ if\ operation\ O_{ji}\ is\ processed\ after\ O_{ki} \\ 0\ otherwise \end{cases}$

- $z_{jir} = \begin{cases} 1 \; if \; operation \; O_{ji} \; is \; processed \; on \; machine \; r \; of \; workstation \; i \\ 0 \; otherwise \end{cases}$

The MIP formulation is as follows:

Minimize:

$$\sum_{j \, \epsilon \, J} C_{j0} \tag{1}$$

Subject to:

$$\sum_{r=1}^{m_i} z_{jir} = 1 \; \forall j \, \epsilon J, i \, \epsilon I \tag{2}$$

$$\mathrm{C_{jl}} \geq \mathrm{C_{ji}} + \mathrm{p_{jl}} + \mathrm{TT_{il}} - \mathrm{Mx_{jil}} \quad \forall j \, \epsilon J, i \, \epsilon I, l \, \epsilon I | l \neq i \tag{3}$$

$$\mathrm{C_{ji}} \geq \mathrm{C_{jl}} + \mathrm{p_{ji}} + \mathrm{TT_{li}} - \mathrm{M}(1 - \mathrm{x_{jil}}) \quad \forall j \, \epsilon J, i \, \epsilon I, l \, \epsilon I | l \neq i \tag{4}$$

$$\mathrm{C_{ji}} \geq \mathrm{C_{ki}} + \mathrm{p_{ji}} - \mathrm{M}\big(1 - \mathrm{y_{jik}}\big) - \mathrm{M}\big(2 - \mathrm{z_{jir}} - \mathrm{z_{kir}}\big) \quad \forall i \, \epsilon I, j \, \epsilon J, \mathrm{k} \, \epsilon J, r \, \epsilon \{1..m_i\} | k \neq j \tag{5}$$

$$\mathrm{C_{ki}} \geq \mathrm{C_{ji}} + \mathrm{p_{ki}} - \mathrm{M}\big(\mathrm{y_{jik}}\big) - \mathrm{M}\big(2 - \mathrm{z_{jir}} - \mathrm{z_{kir}}\big) \quad \forall i \, \epsilon I, j \, \epsilon J, \mathrm{k} \, \epsilon J, r \, \epsilon \{1..m_i\} | k \neq j \tag{6}$$

$$\mathrm{C_{ji}} \geq \mathrm{TT_{oi}} + \mathrm{p_{ji}} \quad \forall i \, \epsilon I | i > 0, j \, \epsilon J \tag{7}$$

$$\mathrm{C_{j0}} \geq \mathrm{TT_{i0}} + \mathrm{C_{ji}} \quad \forall i \, \epsilon I | i > 0, j \, \epsilon J \tag{8}$$

$$\mathrm{C_{ji}} \geq 0 \quad \forall i \, \epsilon I, j \, \epsilon J \tag{9}$$

$$\mathrm{x_{jil}} \epsilon \{0,1\} \; \forall i \, \epsilon I, l \, \epsilon I, j \, \epsilon J | l \neq i \tag{10}$$

$$\mathrm{y_{jik}} \, \epsilon \{0,1\} \; \forall i \, \epsilon I, k \, \epsilon J, j \, \epsilon J | k \neq j \tag{11}$$

$$\mathrm{z_{jir}} \epsilon \{0,1\} \; \forall i \, \epsilon I, j \, \epsilon J, r \, \epsilon \{1..m_i\} \tag{12}$$

The objective is to minimize the total flow time (1). The constraint set (2) ensures that each work is processed by only one machine in each workstation. Constraint sets (3) and (4) guarantee that each job is processed in one workstation at a time, taking into account the travel times between one workstation and another. Constraint sets (5) and (6) make sure that a machine in any workstation can process only one job at a time. Constraint set (7) indicates that all jobs start from a common depot and constraint set (8) indicates that all jobs must return to the depot. Finally, constraint sets (9)-(12) define the decision variables.

In this work, all jobs visit all workstations exactly one time and any job can be processed in any machine in a specific workstation given that all machines are identical. Additionally, all machines are available at the beginning and all machines complete the processing of each job without interruptions.

3.2. Lower Bound

As mentioned before, given the complexity of this problem, a lower bound (LB) proposed in previous work by Mejía et al. (2013) is presented in order to test the performance of the algorithm solution. The LB relaxes the machine capacity constraints. Therefore, the minimum time spent in the system by any job would correspond to the minimum tour time (*min TT*) plus all its service times. This value of the minimal tour time can be stated as the value of the objective function of the optimal solution of the Travelling Salesman Problem (TSP) considering only travel times between workstations. Given that such travel times are not job dependent, a lower bound for the completion time of each job j (C''_j) can be obtained by adding its total processing time $\left(\sum_{i \epsilon I} p_{ji}\right)$ to the value of *minTT*. Hence, the LB can be computed as next:

$$C_j'' = \left(\sum_{i \epsilon I} p_{ji} + \min TT\right) \forall j \epsilon J \qquad (13)$$

$$LB = \sum_{j \in J} C_j'' \quad (14)$$

The optimal solution of the associate TSP is obtained by solving the MIP proposed by Miller et al. (1960). Previously, Mejia et al. (2013) used a standard MIP solver and the same results obtained by them will be used for this work.

4. Solution method: Variable Neighborhood Search

A meta-heuristic approach is proposed to solve the FOSSP considering travel times given the NP-Hard nature of the problem. A Variable Neighborhood Search (VNS) algorithm is suggested because it has show to be very effective when considering parallel machines in Job and Flow (Cheng et al. 2012, Davidovic et al. 2001, Li et al. 2010, Chen et al. 2013, Liu et al. 2007) and performs well in large instances (Hansen & Mladenovic 2003) obtaining good quality results within reasonable time. This section gives a brief background on the meta-heuristic used and explains some aspects regarding its implementation.

4.1. Variable Neighborhood Search (VNS)

Variable Neighborhood Search, a recent meta-heuristic proposed in 1997, is based upon a simple principle: systematic change of neighborhood within the search (Hansen & Mladenovic 2003). It performs a local search but changes systematically of neighborhood to avoid getting trapped in local optima (Sevkli & Aydin 2006). The implemented VNS scheme follows the Basic VNS method proposed by Hansen & Mladenovic (2003) which consists of performing small modifications to generate neighboring solutions and moving to the best one; it combines deterministic and stochastic changes of neighborhood. **Figure 1** shows the steps of the algorithm.

Initialization: find an initial solution x, select the set of neighborhood structures N_k, for $k = 1, .., k_{max}$, that will be used in the search, and choose a stopping condition.

Repeat the following sequence until the stopping condition is met:

(1) Set $k \leftarrow 1$

(2) Repeat the following until $k = k_{max}$:

a) Shaking: generate a point x' at random from the k^{th} neighborhood of x $\left(x \in N_k(x)\right)$.

b) Local search: apply some local search method with x' as initial solution; denote x'' the so obtained local optimum.

c) Move or not: if this local optimum is better than the incumbent, move there ($x \leftarrow x''$) and continue the search with N_1 ($k \leftarrow 1$); otherwise, set $k \leftarrow k + 1$.

Figure 1. Steps of the basic VNS

The meta-heuristic is composed of two phases. The first one is an initialization phase where an initial solution, neighborhood structures, and stopping conditions are defined. The second phase, iteratively explores the search space guided by a neighborhood search meta-heuristic. When the local search method applied in step 2b of the second phase is some other heuristic, the algorithm becomes a hybrid VNS (Hansen & Mladenovic 2003). In this paper, besides using VNS alone, the algorithm is hybridized with two different heuristics: Tabu Search and Simulated Annealing. In other words, the local search in the algorithm is performed by VNS, Tabu Search and Simulated Annealing, using one at a time.

The rest of this section explains the solution encoding and decoding, how the initial solution is generated, the neighborhood structures, the stopping conditions, and the different ways in which the local search is performed.

4.2. Solution encoding

The solution encoding or representation is based on a permutation list which indicates the relative order in which the nxm operations are performed. In the permutation list, an operation is defined by O_{ji} , where j refers to the job and i refers to the workstation. The

operations are scheduled in the first machine available in workstation i. The order in which the operations appear in the permutation list defines the final schedule. However, such order is interpreted differently depending on the selected decoding strategies explained in the next section.

4.3. Solution decoding

The usual decoding scheme that places the current operation of the permutation list in the best possible position in the schedule does not guarantee an active schedule in this case of route/dependent travel times. For this reason, three different decoding schemes are proposed to overcome this potential problem.

- **Sequential decoding:** this is a two-pass decoding (Mejia et al. 2013). In the first pass, the current operation is temporarily assigned at the end of the schedule. In the second pass, the first operation in the permutation list which can be scheduled without changing the start time of the current operation is searched. If such operation exists, then it is scheduled. This results in an active schedule.
- **Non-Delay decoding:** in a non delay schedule no machine is left idle if there is any job that can be processed at the time (Brasel et al. 2008). This results in an active schedule.
- **Active decoding:** in an active schedule no operation can be completely processed earlier without delaying some other operation (Brasel et al. 2008). This is an active schedule.

Here is an example of how the different decoding schemes work. Consider three jobs (1,2,3) and three workstations (A,B,C) each with a single machine. The permutation list would be: {O1a, O1b, O1c, O2a, O2b, O2c, O3a, O3b, O3c}. This list works as a priority list; if there are several operations that can be scheduled, the one that

appears first in the list is the one to be scheduled first. Suppose that the processing times, in minutes, are the ones shown in **Figure 2**, that the travel time between workstations is 30 minutes and that the travel time between the depot and every workstation is zero.

		Jobs		
Workstations		1	2	3
	a	60	60	30
	b	90	30	60
	c	30	30	90

Figure 2. Processing times (in minutes)

Under a sequential decoding scheme, the scheduling would work as follows. The first operation in the permutation list (O1a) is assigned in the empty schedule at time zero. Then, the first operation in the permutation list which can be scheduled without changing the start time of the current operation is searched. Operation O1b is the first one that meets this condition and it is then scheduled in workstation b, 30 minutes after O1a finishes. The same search is performed once again looking for the first operation in the list that can be scheduled without changing the start time of O1b. The resulting operation is O1c so it is scheduled in workstation c, 30 minutes after O1b finishes. When searching again, the first operation that can be scheduled without modifying the start time of O1c is O2a, which is scheduled to start in workstation a when O1a ends. Next, the first operation that can be scheduled is O2b without changing the start time of O2a. This operation is scheduled at time zero in workstation b given that it does not affect the start time of O2a. The scheduling continues with this logic and the resulting schedule is shown in **Figure 3** where each box represents 30 minutes**.**

A	O1a	O1a	O2a	O2a	O3a						
B	O2b	O3b	O3b	O1b	O1b	O1b					
C						O2c		O1c	O3c	O3c	O3c

Figure 3. Final schedule under a sequential decoding scheme

Under a non-delay decoding scheme, the scheduling would work as follows. The first operation within the permutation list that can start the earliest is scheduled first. In this case, all operations could start at time zero so the first one that appears in the permutation list, O1a, is scheduled. Afterwards, the earliest starting times of all operations remaining in the permutation list are updated and the one that can start earlier is scheduled. In this case, O2b, O2c, O3b and O3 could start at time zero. Therefore, O2b is scheduled first because it appears first in the permutation list. The start times are updated once again and O3c is the one that can start earlier so it is scheduled. The next operation to be scheduled is O2a because it is the one that can begin earlier. Afterwards, O1b or O1c could be scheduled because they have the same start time and it is the earliest start time within the start times of the remaining operations. However, O1b is scheduled first because it appears first in the permutation list. The scheduling continues whit this logic and the resulting schedule is shown in **Figure 4** where each box represents 30 minutes**.**

A	O1a	O1a	O2a	O2a	O3a			
B	O2b			O1b	O1b	O1b	O3b	O3b
C	O3c	O3c	O3c			O2c		O1c

Figure 4. Final schedule under a non-delay decoding scheme

Under an active decoding scheme, the scheduling would work as follows. The ending times of all operations in the permutation list that can be scheduled at time zero are compared. The operation that could finish earlier is selected. Afterwards, the operations that correspond to the same machine as the selected operation and that could start before that selected operation's end time are selected. The one that appears first in the permutation list is the one scheduled. In this case, all operations could start at time zero and the one that would finish earlier would be O1c, O2b, O2c and O3a. All operations could be scheduled before the end time of those operations. Therefore, the

first one that appears in the permutation list is the one scheduled. In this case, O1a is the first one to be scheduled. Afterwards, the starting and ending times are updated for all the remaining operations in the permutation list. The operations that could finish earlier are now O2b and O2c. The operations that can start in the respective machines before such ending time are O2b, O2c, O3b and O3c. Operation O2b is the one scheduled because it appears first in the permutation list. Once again the starting and ending times for the remaining operations are updated. The operations that could finish earlier than the rest are O3a, O3b and O3c. The operations that could start before such ending times, in the respective machines, are O2a, O2c, O3a, O3b and O3c. The scheduled operation is O2a because it appears first in the permutation list. The process continues with this logic and the resulting schedule is shown in **Figure 5** where each box represents 30 minutes**.**

A	O1a	O1a	O2a	O2a	O3a				
B	O2b	O3b	O3b			O1b	O1b	O1b	
C				O1c		O2c	O3c	O3c	O3c

Figure 5. Final schedule under an active decoding scheme

It can be seen that the three schedules shown above result in active schedules because no operation can be scheduled earlier without delaying another operation. However, the schedule resulting from the active decoding scheme does not result in a non-delay schedule given that the machine in workstation c is available at time zero and it could be processing O3c, but instead it is left idle.

4.4. Initial Solution

Four initial solutions are generated using a Non-Delay scheme such as in (Cendales et al. 2013) applied to different dispatching rules: Shortest and Longest Processing Time (SPT and LPT respectively) and Shortest and Longest Remaining Processing Time

(SRPT and LRPT respectively). Under a Non-Delay decoding scheme, the SPT (LPT) selects the operation in the set of candidates to be scheduled with the smallest (longest) processing time and schedules it as soon as there is an available machine, provided that no machine should be left idle if there is any job to be processed. The SRPT (LRPT), selects the operation in the set of candidates to be scheduled with the smallest (longest) value of the sum of the processing times of all unscheduled operations of the same job, and schedules it as soon as there is an available machine. The chosen initial solution to start the algorithm is the best one of the four generated solutions.

4.5. Neighborhood structures

Hansen & Mladenovic (2003) state in their Variable Neighborhood Search manual that the best value for the parameter k_{max} is often 2. Therefore, two neighborhood structures were implemented: swap at random and insertion at random.

- **Swap at random:** this movement selects randomly two positions on the permutation list and the operations located in such positions are then exchanged. This movement can only be performed on operations belonging to the same job or between jobs that are processed in a same workstation.
- **Insertion at random:** this movement selects randomly two positions on the permutation list and moves the operation located in the first position to the second selected position of the list.

4.6. Stopping conditions

The algorithm stops when one of these four conditions is first met:

- The optimal solution (if know) is found.
- The maximum number of iterations is reached.

- The maximum number of iterations without improvement is reached.
- The running time is exhausted.

The value that each criterion takes depends on the size of the instance to be solved. This will be discussed later on in the execution result section.

4.7. Local search

This paper considers three different ways in which the local search is performed within the VNS algorithm: basic VNS, VNS hybridized with Tabu Search (VNS+TS) and VNS hybridized with Simulated Annealing (VNS+SA). The Tabu Search (TS) and Simulated Annealing (SA) heuristics were adapted from previous work (Mejia et al. 2013) and their parameters were modified to use the heuristics as the local search methods in the VNS algorithm. A detailed explanation of TS and SA is out of the scope of this paper, but the general concepts are introduced below.

- Basic VNS: uses a simple local search which consists of exploring a certain number of neighbors of a solution and selecting the one that reaches the best objective function.
- VNS+TS: Tabu Search is used in the local search. Given a current solution, Tabu Search generates a set of candidate solutions from the current neighborhood, and chooses one as its new current solution while keeping this movement as the newest into a limited-size list of chronological tabooed movements. If one of the candidates has an objective function better than the “current best so far" solution the candidate replaces both the current and the “current best so far" solutions. Otherwise, the current solution is replaced by the best candidate from the neighborhood which is not tabooed.

- VNS+SA: Simulated Annealing is used in the local search. Given a current solution and a certain temperature, a candidate solution is generated from the current neighborhood. If the candidate's objective function is better than the current solution, then the candidate is adopted as the current solution; otherwise, the candidate replaces the current solution with probability computed by using the Boltzmann constant and the current temperature. This process is done a fixed number of trials for each temperature. After that, the temperature is decreased by using the cooling factor and the process continues until the current temperature reaches the minimum allowed. At any time, if the objective function of a candidate is better than the "current best so far" solution then the candidate is adopted as the "current best so far" solution.

5. Execution results

The algorithm was tested on two different types of problems. It was fist tested on the same problem studied by Yu et al. (2010), which uses Simulated Annealing to minimize the makespan of an Open Shop Scheduling Problem that considers travel times ($O_m/TT_{il}/CMax$), to validate the performance of the implemented algorithms by comparing it to the results published by Yu et al. (2010) and the results obtained by Mejia et al. (2013) when considering the same problem. Once the algorithm worked for the Open Shop problem, it was then tested on the Flexible Open Shop problem with travel times ($FO_m/TT_{il}/\sum C_{j0}$), with a set of randomly generated instances, the same ones used in previous work by Mejia et al. (2013). The results were validated with the optimal solutions (when available), against the obtained results by Mejia et al. (2013) and against the lower bound presented in section 3.

This section explains how the instances on which the algorithms were tested were generated, the parameter calibration for the proposed meta-heuristics and presents the obtained results followed by an analysis for each of the problems tested.

5.1 Computational tests on the $O_m/TT_{il}/CMax$

5.1.1. Instances

The validation of the proposed algorithms was realized by testing a set of instances proposed by Yu et al. (2010). There are six different sets of instances where in each set the number of jobs equals the number of workstations (4x4, 5x5, 7x7, 10x10, 15x15 and 20x20). Every set contains ten different instance configurations.

5.1.2. Parameter calibration

The VNS algorithm has various parameters that had to be established for the initialization phase. The first one was the initial solution, followed by the order in which the neighborhood structures are visited, and finally the stopping criteria. Additionally, some parameters had to be decided for the local search performed within the VNS. All these parameters are described in the rest of this subsection.

As mentioned in section 4.4, the initial solution in the algorithm was established with the best dispatching rule for each problem. However, tests showed that the initial solution did not have significant impact on the final result.

With respect to the order in which the neighborhoods are visited, the reason of which neighborhood is explored first follows the logic of Prandtstetter & Raidl (2005) where swap at random neighborhood goes first followed by the insertion at random neighborhood. This is because the number of neighbors arising from the swapping operations applied to the current best solution is less than the number of neighbors resulting from all the possible inserting operations, and therefore its evaluation is faster.

However, tests showed that the order in which the neighborhoods were explored did not affect the final result.

As mentioned in section 4.6, the algorithm stops when one of four conditions is met. The first one is if the optimal solution is found. This optimal solutions were found by Mejia et al. (2013) for instances 4x4 only. The second condition is if the maximum number of total iterations allowed is reached. This maximum number of iterations was set to 10,000 after experimenting with several trials, for all instances. The third condition is if the algorithm iterates a certain number of times without reaching an improvement in the current solution. This maximum number of iterations without improvement allowed was set to 1,000 (10% of the value of maximum total iterations allowed). The fourth and last stopping condition refers to the maximum execution time allowed. This maximum execution time (in seconds) differs for every set of instances i. It depends on the number of jobs of the current instance set i and the number of jobs in the previous instance i-1. The maximum execution time is determined with the following formula: $time_i = 10n_i + 10(n_i - n_{i-1})$ where the time for the first set of instances was arbitrarily set to 30 seconds[1]. **Table 1** shows the maximum execution time allowed for each set of instances.

Table 1. Maximum execution times.

Instance set i	Number of jobs n_i	$max.execution\ time_i$ (sec.)
1 (4x4)	4	30
2 (5x5)	5	60
3 (7x7)	7	90
4 (10x10)	10	130
5 (15x15)	15	200
6 (20x20)	20	250

[1] One of the authors of the paper written by Mejía et al (2013) suggested execution times for each instance set. The presented formula relates the suggested times with the number of jobs in the instance sets.

Additional parameters had to be decided for the local search performed within the VNS. On one side, the local search requires a neighborhood size which indicates the amount of neighbors to be explored in the local search. This neighborhood size is different for each set of instances and is set as $\alpha * time_i$ where α is an arbitrary constant set to0.25.

On the other hand, when the local search is executed with TS or with SA, additional parameters are needed. These parameters for the TS and for the SA were adapted from Mejia et al. (2013) experimenting with different trials. These parameters and their values are explained below.

When TS was used as the local search method, the number of iterations was set to 50 (10% of the maximum number of iterations fixed by Mejia et al. (2013)). The reason for using 10% is that TS is used for a small local search and not to solve the entire problem. For all instance sizes, the size of the Tabu list was set to 7, which is the amount of jobs in a medium sized instance; the number of neighbors explored every iteration was set to 10.

When SA is used as the local search, after experimenting with several trials, the initial temperature was set to 5 (10% of the initial temperature considered by Mejia et al. (2013)), the final temperature was left the same 0.1, the cooling factor and the Boltzmann constant were left the same (0.995 and 1 respectively) and the number of neighbors explored at each temperature was set to 10 (1% of the established by Mejia et al. (2013)).

5.1.3. Result analysis

For evaluating the performance of the proposed VNS, VNS+TS, VNS+SA algorithms, the results are compared against the optimal solution when available, against the results

published by Yu et al. (2010) and against the best solution obtained by Mejia et al. (2013). Optimal values are known for the 4x4 and 5x5 instances.

Every instance was executed 5 times for every variation of the VNS (VNS, VNS+TS, VNS+SA) with three different decoding schemes (active, non-delay and sequential). Only the best of the five results found for every decoding scheme is reported for every instance in each algorithm. **Table 2** and **Table 3** show for each of the 60 tested instances the best result obtained by Yu et al. (2010), the best result obtained by Mejia et al. (2013), referred to as MEJIA, the best result found with each decoding for each of the three variations of the VNS algorithm and the gap between the best solution (BS) and the results found by Yu et al. (2010), referred to as YU. The gap was calculated using the following formula:

$$GAP = \frac{\mathrm{BS-YU}}{\mathrm{YU}} \times 100 \qquad (15)$$

Table 2. Experimental results $O_m/TT_{il}/CMax$ (small instances)

PROBLEM	VNS			VNS+SA			VNS+TS			Overall[1]		Mejia		YU
	Act	ND	Seq	Act	ND	Seq	Act	ND	Seq	Best	Gap	Best	Gap	
04x04_01	72	72	73	72	72	72	72	72	72	72*	0.0%	72	0.0%	72
04x04_02	74	74	74	74	74	74	74	74	74	74*	0.0%	74	0.0%	74
04x04_03	76	80	76	76	80	76	76	80	76	76*	0.0%	76	0.0%	76
04x04_04	79	79	79	79	79	79	79	79	79	79*	0.0%	79	0.0%	79
04x04_05	79	79	80	79	79	79	79	79	79	79*	0.0%	79	0.0%	79
04x04_06	78	79	78	78	79	78	78	79	78	78*	0.0%	78	0.0%	78
04x04_07	71	74	71	71	74	71	71	74	71	71*	0.0%	71	0.0%	71
04x04_08	68	68	69	68	68	68	68	68	68	68*	0.0%	68	0.0%	68
04x04_09	83	86	83	83	86	83	83	86	83	83*	0.0%	83	0.0%	83
04x04_10	76	76	78	76	76	76	76	76	76	76*	0.0%	76	0.0%	76
05x05_01	90	92	91	90	92	89	90	92	91	89*	0.0%	89	0.0%	89
05x05_02	84	84	86	84	84	84	84	84	85	84*	0.0%	84	0.0%	84
05x05_03	97	100	98	98	100	99	97	100	98	97*	0.0%	97	0.0%	97
05x05_04	94	96	95	94	96	94	94	96	94	94*	0.0%	94	0.0%	94
05x05_05	92	94	95	92	94	93	94	94	93	92*	0.0%	92	0.0%	92
05x05_06	92	95	94	93	95	93	93	95	93	92*	0.0%	92	0.0%	92
05x05_07	96	98	97	96	98	96	96	98	97	96*	0.0%	96	0.0%	96
05x05_08	90	90	94	91	90	92	91	90	92	90*	0.0%	90	0.0%	90
05x05_09	92	93	94	94	93	94	93	93	93	92*	0.0%	92	0.0%	92
05x05_10	94	94	94	95	94	94	94	94	94	94	1.1%	93	0.0%	93
07x07_01	125	119	123	127	121	126	125	119	123	119***	0.0%	119	0.0%	119
07x07_02	122	117	123	126	119	124	122	118	123	117***	0.9%	117	0.9%	116
07x07_03	133	131	134	138	130	134	136	130	134	130	0.8%	129	0.0%	129
07x07_04	130	124	125	133	125	131	127	124	127	124***	-1.6%	124	-1.6%	126
07x07_05	126	126	127	131	125	130	130	125	129	125**	-0.8%	124	-1.6%	126
07x07_06	134	128	132	135	127	135	132	127	133	127***	-0.8%	127	-0.8%	128
07x07_07	126	119	126	130	119	126	127	119	126	119***	-1.7%	119	-1.7%	121
07x07_08	124	124	127	129	123	128	128	124	127	123	0.8%	122	0.0%	122
07x07_09	122	122	125	128	122	129	122	121	125	121***	-0.8%	121	-0.8%	122
07x07_10	116	117	121	125	116	123	116	116	122	116**	-1.7%	115	-2.5%	118

[1.] Refers to the best solution found within the three algorithms.

*Optimal solution

** The found solution is better than that of YU but not better than that of MEJIA.

*** The solution found is better than, or equal to, the solutions found by both YU and MEJIA.

Table 3. Experimental results $O_m/TT_{il}/CMax$ (large instances)

PROBLEM	VNS			VNS+SA			VNS+TS			Overall[1]		MEJIA		YU
	Act	ND	Seq	Act	ND	Seq	Act	ND	Seq	Best	Gap	Best	Gap	
10x10_01	183	174	187	193	181	193	187	178	189	174***	-1.1%	176	0.0%	176
10x10_02	175	163	171	183	164	172	173	164	170	163**	-1.2%	163	-1.2%	165
10x10_03	168	164	165	172	165	169	172	165	169	164***	0.0%	164	0.0%	164
10x10_04	170	161	166	179	163	169	174	162	167	161**	-0.6%	160	-1.2%	162
10x10_05	182	174	175	188	171	182	185	173	178	171	0.6%	172	1.2%	170
10x10_06	171	160	163	176	160	173	169	159	166	159**	0.0%	157	-1.3%	159
10x10_07	176	163	168	182	162	171	177	163	169	162**	-1.2%	161	-1.8%	164
10x10_08	174	167	172	174	166	173	174	166	171	166	0.6%	163	-1.2%	165
10x10_09	178	168	169	183	167	172	178	167	171	167	0.6%	165	-0.6%	166
10x10_10	168	158	165	178	161	172	171	160	167	158***	-1.9%	160	-0.6%	161
15x15_01	268	251	258	276	249	268	274	252	265	249***	-2.4%	249	-2.4%	255
15x15_02	264	245	248	264	248	253	262	247	251	245**	-3.5%	244	-3.9%	254
15x15_03	265	240	248	268	238	249	268	241	248	238**	-3.3%	235	-4.5%	246
15x15_04	281	252	253	283	252	257	283	251	257	251**	-0.8%	248	-2.0%	253
15x15_05	268	242	248	275	244	249	273	241	246	241***	-4.4%	241	-4.4%	252
15x15_06	266	242	245	269	243	248	273	240	246	240**	-2.8%	238	-3.6%	247
15x15_07	266	246	258	269	249	265	270	246	271	246**	-2.4%	245	-2.8%	252
15x15_08	264	241	246	265	243	251	265	242	247	241***	-2.0%	241	-2.0%	246
15x15_09	267	238	246	271	241	247	271	239	247	238**	-3.6%	236	-4.5%	247
15x15_10	276	246	258	274	250	268	280	247	261	246**	-2.8%	245	-3.2%	253
20x20_01	350	312	316	351	314	320	351	313	319	312**	-6.9%	308	-8.1%	335
20x20_02	359	328	327	369	326	332	369	328	331	326**	-6.1%	321	-7.5%	347
20x20_03	364	319	324	369	324	332	364	320	330	319**	-6.7%	317	-7.3%	342
20x20_04	365	327	332	373	326	336	370	327	335	326**	-5.8%	323	-6.6%	346
20x20_05	361	320	325	361	322	329	370	320	326	320**	-6.4%	316	-7.6%	342
20x20_06	357	318	334	354	324	338	358	319	338	318**	-6.2%	315	-7.1%	339
20x20_07	372	332	344	366	331	357	373	334	350	331**	-4.9%	325	-6.6%	348
20x20_08	354	312	322	357	316	324	354	316	324	312***	-8.2%	312	-8.2%	340
20x20_09	346	314	325	345	315	326	349	314	325	314**	-6.5%	310	-7.7%	336
20x20_10	361	324	331	361	329	339	368	325	338	324**	-5.3%	320	-6.4%	342

[1.] Refers to the best solution found within the three algorithms.

*Optimal solution

** The found solution is better than that of YU but not better than that of MEJIA.

*** The solution found is better than, or equal to, the solutions found by both YU and MEJIA.

Note: the average results obtained in each of the three algorithms with every decoding scheme are shown for every instance in **Annex 1.**

All optima were found for the 4x4 instances, as well as in Mejia et al. (2013), with the three algorithms. In VNS+TS and VNS+SA, the active and the sequential decoding schemes found all the optima while the non-delay found 60% of the optimal solutions. With VNS, all optimal values were found with the active decoding scheme, while the non-delay and sequential decoding schemes found the 60% of the optimal solutions, with solutions averaging around 1.4% and 0.7%, respectively, from the optimal values.

With respect to the 5x5 instances, 9 of the 10 optima were found, while in MEJIA the 10 optimal values were obtained. However, the optimum that was not obtained resulted 1.1% away from the optimal value. The active decoding scheme performed better in the three algorithms obtaining 80% of the optima in VNS, 40% in VNS+SA and VNS+TS. With non-delay only 20% of the optima were achieved, in every algorithm. The sequential decoding scheme found no optimal solutions in the VNS, 10% of the optima in the VNS+TS, and 40% of the optima in the VNS+SA.

For the remaining instances, the optimal values are not known. Therefore, the obtained solutions are compared to those found by Yu et al. (2010) and by Mejia et al. (2013). For the 7x7 instances, when comparing the three algorithms, VNS+TS found the best solution in 8 of the 10 instances. VNS+SA found 6 times the best solution while VNS alone found 5 of the times the best solutions. The best solutions were always found with the non-delay decoding scheme. No best solutions were found with the active or with the sequential decoding schemes. Of the ten instances, six of the best solutions found were better than the results published by Yu et al (2010) (4 of them were the same solutions found in MEJIA and two of them were worse). The resulting average gap for the 7x7 instance set when compared to the solutions found by Yu et al (2010) is -0.5%, while in Mejia et al. (2013) it was -0.8%.

For the 10x10 instances, VNS found 5 times the best solutions while VNS+SA found 4 of the best results and VNS+TS found 3 of the best solutions. The Non-Delay decoding scheme worked better than the other two; no best solutions were found with active or sequential decoding schemes. Of the ten instances, five of the best solutions found were better than the results published by Yu et al (2010) (1 of them was the same solution found in MEJIA, two of them were worse and two of them were better). The resulting average gap for the 10x10 instance set when compared to the solutions found by Yu et al (2010) is -0.4%, while in Mejia et al. (2013) it was -0.7%.

With respect to the larger instances (15x15 and 20x20), all the solutions found outperformed those of Yu et al. (2010). However, the solutions found in this case are not better than the results obtained by Mejia et al. (2013). Four results were equal to the ones found in MEJIA but the other 16 were worse. Additionally, the best solutions were always found with the non-delay decoding scheme. Active and sequential decoding schemes did not work well for large instances. The VNS algorithm averaged -2.5% of improvement over the YU results in the 15x15 instances and -6.2% in the 20x20 instances. VNS+TS averaged -2.4% and -5.9%, respectively, and VNS+SA averaged -1.9% and -5.6%, respectively. Therefore, it can be seen that the VNS algorithm performed better than the other two. When considering only the best results, the average gap in the 15x15 instances was -2.8%, while in MEJIA it was -3.3%; the average gap in the 20x20 instances was -6.3%, while in MEJIA it was -7.3%.

Given the preceding analysis, it can be concluded that the VNS algorithm and its hybrids generally found better solutions than those found by Yu et al (2010) and found similar solutions to those of Mejia et al. (2013). The average gaps with the VNS algorithms were close to the ones found by Mejia et al. (2013). Therefore, the algorithms can be validated and are proved to work and find good solutions.

5.2 Computational tests on the $FO_m/TT_{il}/\sum C_{j0}$

5.2.1. Instances

The instances on which the algorithm was tested were generated for previous work (Cendales et al. 2013) and used again by Mejia et al. (2013). Six different sets of instances, where the number of jobs equals the number of workstations, were generated (4x4, 5x5, 7x7, 10x10, 15x15 and 20x20). Each set is composed of ten different instances.

The instances were generated using the methodology proposed by Taillard (1993) for the Open Shop. The travel times, which are not considered by Taillard (1993), were generated with the timed-seed values provided by Yu et al. (2010) over a uniform distribution U[2,10]. Similarly, the processing times were generated using a uniform distribution U[8,14] as in (Yu et al. 2010) where they used the time and machine seeds suggested by Taillard (1993). The number of machines in each workstation was created with the same machine seed. For instances 4x4 to 10x10 the number of parallel machines is uniformly distributed U[1,2] whereas for instances 15x15 and 20x20 the number of parallel machines was uniformly distributed U[1,3].

5.2.2. Parameter calibration

The parameters used in the VNS algorithms are the same ones that were established and validated for the previous problem. The initial solution in the algorithm was established with the best dispatching rule for each problem. And once again, tests showed that the initial solution did not have significant impact on the final result. Additionally, the order of the visited neighborhoods had no impact in the final solution. As established before, the swap at random neighborhood was explored before the insertion at random neighborhood. The stopping conditions were left the same as calibrated for the previous

problem (the OSSP). The algorithm stops if the optimal solution is found; or if the maximum number of iterations, set to 10,000, is reached; or if the maximum number of iterations without improvement, set to 1000, is reached; or if the previously established execution time is exhausted. Finally, the neighborhood size to be explored and the rest of the parameters for the local search using Tabu Search or Simulated Annealing were left the same.

5.2.3. Result analysis

For evaluating the performance of the proposed VNS, VNS+TS, VNS+SA algorithms, the obtained results are compared against the optimal solution when available, against the lower bound (LB) of each instance and against the best solution obtained by Mejia et al (2013). The values of the optimal solutions for the 4x4 instances and the lower bounds for the rest of the instances were computed by Mejia et al. (2013) using the Xpress MP®Solver on the model introduced in section 3. The execution time of the MIP model developed to obtain an optimal solution was limited to 3600 seconds.

Every instance was executed 5 times for every algorithm with each decoding scheme (active, non-delay and sequential). The best result found within the 5 iterations is reported for every instance in every VNS variation. **Table 4** and **Table 5** show for each instance the LB (or optimal solution), the best result from Mejia et al. (2013), referred to as MEJIA, the best result found with each decoding scheme in each VNS algorithm and the gap between the best solution (BS) and the respective lower bound (LB) or optimal solution (OPT) when available.

$$GAP = \frac{\text{BS}-\text{LB(OPT)}}{\text{LB(OPT)}} \times 100 \qquad (16)$$

Table 4. Experimental results $FO_m/TT_{il}/\sum C_{j0}$ (small instances)

PROBLEM	VNS			VNS + SA			VNS + TS			Overall[1]		MEJIA		LB
	Act	ND	Seq	Act	ND	Seq	Act	ND	Seq	Best	Gap	Best	Gap	
04x04_01	228	228	228	228	228	228	228	228	228	228*	0.0%	228	0.0%	228
04x04_02	291	291	287	291	291	286	291	291	286	286*	0.0%	286	0.0%	286
04x04_03	295	297	294	295	297	294	295	297	294	294*	0.0%	294	0.0%	294
04x04_04	290	294	290	290	294	290	290	294	90	290*	0.0%	290	0.0%	290
04x04_05	345	357	328	345	357	328	345	357	328	328*	0.0%	328	0.0%	328
04x04_06	258	260	256	258	260	253	258	260	256	253*	0.0%	253	0.0%	253
04x04_07	264	279	263	264	279	263	264	279	263	263*	0.0%	263	0.0%	263
04x04_08	272	272	272	272	272	272	272	272	272	272*	0.0%	272	0.0%	272
04x04_09	324	334	315	324	334	315	324	334	315	315*	0.0%	315	0.0%	315
04x04_10	273	287	273	273	287	273	273	287	273	273*	0.0%	273	0.0%	273
05x05_01	421	425	417	420	418	417	420	418	417	417*	3.5%	417	3.5%	403
05x05_02	370	380	368	370	380	368	370	380	368	368*	3.1%	368	3.1%	357
05x05_03	453	455	455	453	455	451	453	455	451	451*	6.4%	451	6.4%	424
05x05_04	458	465	454	459	465	449	452	465	448	448	3.7%	447	3.5%	432
05x05_05	477	480	478	477	480	478	477	480	477	477	3.0%	473	2.2%	463
05x05_06	431	453	435	431	453	431	431	453	430	430**	2.4%	431	2.6%	420
05x05_07	449	457	443	446	457	443	446	457	443	443*	1.1%	443	1.1%	438
05x05_08	457	477	455	455	477	455	457	477	455	455*	3.6%	455	3.6%	439
05x05_09	487	493	482	486	486	480	487	486	481	480**	3.7%	482	4.1%	463
05x05_10	464	459	458	458	459	461	460	459	460	458	7.5%	454	6.6%	426
07x07_01	863	849	863	867	842	855	864	842	857	842**	7.4%	846	7.9%	784
07x07_02	807	776	778	792	782	776	803	769	778	769*	8.5%	769	8.5%	709
07x07_03	914	902	903	909	903	897	908	896	898	896**	7.4%	897	7.6%	834
07x07_04	858	853	852	864	851	861	851	851	854	851*	4.4%	851	4.4%	815
07x07_05	800	781	799	791	779	784	801	781	790	779**	7.2%	787	8.3%	727
07x07_06	875	874	883	880	882	878	893	874	874	874**	6.5%	875	6.6%	821
07x07_07	839	824	827	834	832	827	852	822	834	822**	7.5%	827	8.1%	765
07x07_08	821	791	805	817	791	793	822	789	802	789**	8.2%	792	8.6%	729
07x07_09	826	806	822	817	813	815	837	808	822	806*	5.2%	806	5.2%	766
07x07_10	745	726	741	736	727	740	744	725	740	725**	4.3%	726	4.5%	695

[1.] Refers to the best solution found within the three algorithms.

*The solution is the same one found by MEJIA.

** The solution is better that that of MEJIA.

Table 5. Experimental results $FO_m/TT_{il}/\sum C_{j0}$ (large instances)

PROBLEM	VNS			VNS + SA			VNS + TS			Overall[1]		MEJIA		LB
	Act	ND	Seq	Act	ND	Seq	Act	ND	Seq	Best	Gap	Best	Gap	
10x10_01	1721	1596	1,647	1699	1602	1,640	1716	1603	1,647	1596**	8.4%	1599	8.6%	1472
10x10_02	1672	1564	1,585	1660	1572	1,569	1664	1567	1,594	1564**	8.8%	1567	9.0%	1438
10x10_03	1731	1626	1,643	1721	1619	1,648	1732	1607	1,649	1607**	7.8%	1628	9.2%	1491
10x10_04	1643	1552	1,588	1669	1564	1,596	1665	1556	1,593	1552**	8.2%	1564	9.0%	1435
10x10_05	1737	1653	1,690	1714	1665	1,671	1735	1663	1,681	1653**	8.6%	1659	9.0%	1522
10x10_06	1590	1499	1,505	1602	1502	1,517	1586	1499	1,525	1499*	7.6%	1499	7.6%	1393
10x10_07	1661	1553	1,594	1660	1562	1,596	1659	1553	1,591	1553**	6.4%	1557	6.6%	1460
10x10_08	1645	1524	1,533	1617	1521	1,537	1629	1515	1,535	1515**	8.6%	1516	8.7%	1395
10x10_09	1671	1587	1,600	1670	1588	1,601	1667	1580	1,609	1580**	8.1%	1588	8.6%	1462
10x10_10	1624	1508	1,548	1619	1514	1,545	1597	1501	1,545	1501**	7.4%	1511	8.1%	1398
15x15_01	3821	3442	3,440	3840	3459	3,439	3827	3461	3,465	3439	10.2%	3433	10.0%	3120
15x15_02	3836	3443	3,532	3772	3468	3,531	3779	3440	3,535	3440**	7.9%	3471	8.9%	3188
15x15_03	3856	3484	3,559	3871	3454	3,569	3851	3488	3,579	3454**	11.3%	3501	12.8%	3103
15x15_04	3766	3384	3,438	3729	3404	3,429	3735	3393	3,443	3384**	9.4%	3416	10.4%	3094
15x15_05	3736	3370	3,418	3715	3391	3,420	3718	3377	3,429	3370**	4.3%	3381	4.6%	3231
15x15_06	3705	3351	3,366	3709	3325	3,371	3663	3330	3,398	3325	6.3%	3324	6.3%	3127
15x15_07	3626	3400	3,451	3656	3417	3,410	3636	3405	3,462	3400**	11.5%	3403	11.6%	3050
15x15_08	3719	3351	3,381	3705	3369	3,379	3690	3326	3,383	3326**	9.1%	3368	10.5%	3048
15x15_09	3719	3408	3,459	3687	3452	3,456	3744	3387	3,486	3387**	12.0%	3403	12.5%	3024
15x15_10	3834	3480	3,550	3858	3489	3,554	3859	3454	3,554	3454**	9.6%	3484	10.5%	3152
20x20_01	6705	5899	5,947	6702	5970	5,992	6689	5887	5,992	5887**	12.4%	5975	14.0%	5239
20x20_02	6743	6011	5,904	6691	5939	5,947	6757	5959	5,944	5904**	9.4%	5920	9.6%	5399
20x20_03	6638	6252	6,230	6550	6213	6,201	6625	6190	6,302	6190**	11.0%	6238	11.9%	5575
20x20_04	6699	5975	6,073	6747	6035	6,065	6705	6015	6,097	5975**	11.1%	6047	12.5%	5376
20x20_05	6961	6331	6,387	7028	6341	6,382	7104	6282	6,413	6282**	14.1%	6330	15.0%	5504
20x20_06	6735	5856	5,887	6799	5906	5,906	6762	5878	5,906	5856**	9.7%	5881	10.2%	5337
20x20_07	6556	5917	5,967	6572	5919	5,988	6517	5910	5,988	5910**	6.4%	5917	6.5%	5554
20x20_08	7061	6340	6,274	7015	6352	6,284	6998	6315	6,284	6274**	12.6%	6278	12.7%	5573
20x20_09	6813	6125	6,086	6695	6131	6,029	6717	6067	6,109	6029**	12.4%	6082	13.4%	5364
20x20_10	6805	6112	6,076	6827	6025	6,083	6795	6122	6,081	6025**	11.6%	6078	12.5%	5401

[1.] Refers to the best solution found within the three algorithms.

*The solution is the same one found by MEJIA.

** The solution is better that that of MEJIA.

Note: the average results obtained in each of the three algorithms with every decoding scheme are shown for every instance in **Annex 2.**

It is possible to see that in the 4x4 instances all optima were found as well as in Mejia et al. (2013). In the remaining 50 instances, 36 of the solutions were improved and 9 resulted in the same solution found in MEJIA. Only in 5 of the instances the solutions were not equaled nor improved (the results in 3 of the 5x5 instances and 2 of the 15x15 instances were not improved). Therefore, it is possible to state that the VNS algorithm and its hybrids substantially improved the results found by Mejia et al. (2013) which implemented Tabu Search and Simulated Annealing.

When comparing the results between the three algorithms, it is possible to see that VNS+TS performed better than the rest given that it found in 39 of the 60 instances the best result, followed by VNS alone where 29 of the best solutions were found. However, VNS+SA found the best solution in 26 of the instances indicating that it is also a good algorithm for solving the FOSSP. This comparison can also be seen in **Table 6** below where the average gaps for the different instance sets and algorithms are presented.

Table 6. Summary of gaps for the experimental results

Instance	VNS	VNS+SA	VNS+TS	With best[1]
4x4	0.15%	0.00%	0.12%	0.00%
5x5	4.05%	3.85%	3.85%	3.80%
7x7	7.02%	7.03%	6.71%	6.65%
10x10	8.26%	8.57%	8.14%	7.97%
15x15	9.59%	9.88%	9.43%	9.16%
20x20	11.47%	11.66%	11.45%	11.07%
Average	6.76%	6.83%	6.61%	6.44%

[1.] The gap of the best solution found for an instance, within the three algorithms, with respect to the lower bound.

It is possible to state that good results were achieved in terms of the quality of the GAP, 6.44% in average, considering the best solutions of all the tested instances. This is a better average gap than the one found by Mejia et al. (2013), 8%. It can also be

seen that the VNS+TS outperforms the other two algorithms obtaining an average gap of 6.61%. Regarding the 4x4 instances, the VNS+SA algorithm was the only one that reached optimality in every instance. For the rest of the instance sets, the smallest average gap was obtained with the VNS+TS. In the 5x5 instances, the gap ranged between 3.85% and 4.05% while in the 7x7 and the 10x10 instances the gap ranged between 6.71% and 8.57%. In the 15x15 and 20x20 instances, the gap ranged between 9.45% and 11.66%. Anyhow, the three algorithms achieve similar average gaps around 6% and all of them proved to be good algorithms to solve this problem.

With respect to the decoding schemes, **Table 7** shows the average gap for every decoding scheme in every set of instances and every algorithm. It is worth noticing that the active and the sequential decoding schemes found the best results for the 4x4 and 5x5 instances with the three algorithms. The non-delay decoding worked better for the 7x7, 10x10, 15x15 and 20x020 instances. Therefore, it is possible to say that the quality of the solutions depends heavily on the decoding scheme used.

Table 7. Summary of gaps by decoding scheme

Instance	VNS			VNS+SA			VNS+TS			With best[1]
	Act	ND	Seq	Act	ND	Seq	Act	ND	Seq	
4x4	1.25%	3.30%	**0.15%**	1.25%	3.30%	**0.00%**	1.25%	3.30%	**0.12%**	0.00%
5x5	4.73%	6.56%	**4.22%**	4.45%	6.24%	**3.94%**	4.41%	6.24%	**3.87%**	3.80%
7x7	9.27%	**7.03%**	8.25%	8.69%	**7.30%**	7.62%	9.61%	**6.71%**	7.94%	6.65%
10x10	15.42%	**8.26%**	10.13%	14.99%	**8.59%**	10.05%	15.10%	**8.14%**	10.38%	7.97%
15x15	20.84%	**9.60%**	11.14%	20.60%	**9.97%**	11.02%	20.48%	**9.43%**	11.59%	9.16%
20x20	24.70%	**11.96%**	11.99%	24.53%	11.99%	12.08%	24.61%	**11.61%**	12.51%	11.07%
Average	12.70%	7.79%	7.65%	12.42%	7.90%	7.45%	12.58%	7.57%	7.74%	6.44%

[1.] The gap of the best solution found for an instance within the three algorithms with respect to the lower bound.

Finally, it is important to clarify that in the smaller instance, 4x4, the optimal solutions were found before the maximum running time was exhausted while in the other instances, the running time determined the termination of the runs. The cases where the condition of the maximum number of iterations without improvement ended the algorithm were minimal.

6. Conclusions and future work

This paper has presented a Variable Neighborhood Search approach to solve a Flexible Open Shop Problem considering travel times. The performance of the proposed algorithm for the studied problem is remarkable in terms of solution quality. In the proposed tests, the VNS and the two hybrids performed consistently well. On one hand, when testing the algorithms on the Open Shop Scheduling Problem (no parallel machines) the algorithms found much better solutions as compared with those of Yu et al. (2010) which have been the benchmark for such problems. Additionally, when comparing the solutions to the results obtained by Mejia et al. (2013), the solution for two of the instances was improved while the other 58 solutions were equal or very close to the ones obtained previously.

On the other hand, the algorithm found near-optimal solutions for the Flexible Open Shop Scheduling Problem with travel times between stations and found much better results than those of Mejia et al. (2013) in which SA and TS were used. Therefore, given the parameters used, it is possible to say that the VNS algorithm and its hybrids worked better for the problem that considered parallel machines and whose objective was to minimize the total flow time, than on the problem considering only one machine and with the objective to minimize the makespan.

When testing both problems, it is clear that the quality of the solutions depends a great deal on the decoding scheme. Specifically, the active and the sequential decoding

schemes performed better for small instances whereas the non-delay decoding scheme worked better for the large instances. For that reason, future work could consider variable decoding schemes. As well, future work will be focused on incorporating additional constraints to make the models more realistic, such as time-of-the-day dependent travel times. In addition population-based algorithms such as Genetic Algorithms and Ant Colony Optimization could be tested.

As mentioned before, this paper is the result of a larger project which has, among other things, the objective of implementing different algorithms to solve Open Shop Scheduling Problems.

7. Acknowledgements

This Project has been funded by the Research Center of the School of Engineering (CIFI), project P12.244822.003/01. This is a joint effort by the PyLO (Production and Logistics) and TICSw (Information Technologies and Software Construction) research groups of the School of Engineering of the Universidad de los Andes.

8. References

Ahmadizar, F. & Farahani, M. (2012). A novel hybrid genetic algorithm for the open shop scheduling problem. Int J Adv Manuf Technol 62(5-8), 775–787.

Alcaide, D., Sicilia, J. & Vigo, D. (1997). A Tabu Search Algorithm for the Open Shop Problem. TOP, (Vol. 5), 283 - 286.

Andersen, M., Brasel, H., Morig, M., Tusch J., Werner, F. & Willenius, P. (2008). Simulated annealing and genetic algorithms for minimizing mean flow time in an open shop. Mathematical and Computer Modeling, (Vol. 48), 1279-1293.

Barzegar, B.; Motameni, H. & Bozorgi, H. (2012). Solving flexible job-shop scheduling problem using gravitational search algorithm and colored Petri net. Journal of Applied Mathematics. Vol. 2012.

Blum, C. (2005). Beam-Aco – Hybridizing Ant Colony Optimization with Beam Search: An Application to Open Shop Scheduling, Computers & Operations Research, (Vol. 32), 1565 - 1591.

Brasel, H., Herms, A., Moring, M., Tautenhahn, T., Tush, J. & Werner, F. (2008). Heuristic constructive algorithms for open shop scheduling to minimize mean flow time. European Journal of Operational Research,189(3), 856-870.

Brucker, P. (2001). Scheduling Algorithms. New York, USA: Spring-Verlag.

Brucker, P., Hurnik, J., Jurisch, B. & Wstmann, B. (197). A branch and bound algorithm for the open-shop problem. Discrete Applied Mathematics, (Vol. 76), 43-59.

Cendales, O., Mejía, G., Méndez, D. & Casallas, R. (2013). A simulated annealing algorithm for the vehicle routing and scheduling problem. 22nd International Conference on Production Research. Iguazu, Brazil.

Chen, B. & Strusevich, V. A. (1993) Worst-case analysis of heuristics for open shops with parallel machines. European Journal of Operational Research, 70(3), 379-390.

Chen, Y., Cheng, C., Wang, L. & Chen, T. (2013). A hybrid approach based on the variable neighborhood search and particle swarm optimization for parallel machine scheduling problems. Int. J. Production of Economics, 141 (1), 66-78.

Cheng, W., Guo, Peng., Shang, Z., Zeng, M. & Liang, J. (2012). Variable neighborhood search for parallel machines scheduling problem with step deteriorating jobs. Mathematical problems in engineering, (Vol. 2012), 1-20.

Davidovic, T., Hansen, P. & Mladenovic, N. (2001). Variable neighborhood search for multiprocessor scheduling problem with communication delays. MIC'2001 – 4th Metaheuristics International Conference, 737-741.

Graham, R., Lawler, E., Lenstra, J. & Kan, A. (1979) Optimization and approximation in deterministic sequencing and scheduling: a survey. Annals of Discrete Mathematics, Elsevier (Vol.5), 287-326.

Hansen, P. & Mladenovic, N. (2003). Variable Neighborhood search. Handbook of metaheuristics, chapter 6. 145-184.

Hashemi, S. (2010). A mixed integer linear formulation for the open shop earliness-tardiness scheduling problem. Applied Mathematical Sciences, 4(35), 1703 – 1710.

Jansen, K.; Lugano, I. & Sviridenko, M.I. (2010). Polynomial time approximation schemes for the multiprocessor open and flow shop scheduling problem. STACS 200, LNCS 1770, 455-565.

Jurcik, K. (2009). Open shop scheduling to minimize makespan. I-21. Thunder Bay, Ontario: Lakehead University Department of Mathematical Sciences.

Lawler, E. L., Luby, M. G. & Vazirani, V. (1982). Scheduling open shops with parallel machines. Operations Research Letters I(4), 161-164.

Li, J., Pan, Q. & Xie, S. (2010). A hybrid variable neighborhood search algorithm for solving multi-objective flexible job shop problems. Computer Science and Information Systems, 7(4), 907-930.

Liaw, C.F. (2000). A hybrid genetic algorithm for the open shop scheduling problem. European Journal of Operational Research, (Vol. 124), 28-42.

Liaw, C.F. (2003). An efficient tabu search approach for the two-machine preemptive open shop scheduling problem. Computers & Operations Research, (Vol. 30). 2081-2095.

Liaw, C.F. (2005). Scheduling preemptive open shops to minimize total tardiness. European Journal of Operational Research, (Vol. 162), 173-183.

Lin, H-H. & Sha, D-Y. (2011). A new particle swarm optimization for multi-objective open shop scheduling. Journal of Information and Optimization Sciences, (Vol. 32).

Liu, H., Abraham, A., Grosan, C. & Ninging Li. (2007). A novel variable neighborhood particle swarm optimization for multi-objective flexible job-shop scheduling

problems. 2nd International Conference on Digital Information Management, Lyon, France, 138-145.

Low, C. & Yehh, Y. (2009). Genetic algorithm-based heuristics for an open shop scheduling problem with setup, processing and removal times separated, Robotics and Computer-Integrated Manufacturing, 25(2), 314-322.

Malapert, A., Cambazard, H., Guéret, C., Jussien, N., Langevin, A. & Rousseau, L.-M. (2012). An Optimal Constraint Programming Approach to the Open-Shop Problem. Journal on Computing, 24(2), 228–244.

Matta, M.E. (2009). A genetic algorithm for the proportionate multiprocessor open shop. Computers & Operations Research, 36(9), 2601-2618.

Mejia, G., Mendez-Acuña, D., Montoya, C., Casallas, R., Romero, J., Caballero-Villalobos, J.P., Cendales, O. & Amortegui, J.G. (2013) A metaheuristic-based framework for vehicle routing and scheduling problems. Computers & Industrial Engineering – Under Review.

Miller, C.E., Tucker, A.W. & Zemlin, R.A. (1960). Integer programming formulation of traveling salesman problems. Journal of the ACM, (Vol. 7), 326-329.Naderi, B., Fatemi, S.M.T., Aminnayeri, M. & Zandieh, M. (2010). A contribution and new heuristics for open shop scheduling. Computers & Operations Research, (Vol. 37). 213-221.

Naderi, B., Fatemi, S.M.T., Aminnayeri, M. & Zandieh, M. (2011). Scheduling open shops with parallel machines to minimize total completion time. Journal of Computational and Applied Mathematics, 235(5), 1275-1287.

Pacino, D. & Hentenryck, P.V. (2011). Large neighborhood search and adaptative randomized decompositions for flexible job shop scheduling. Twenty-second International Joint Conference on Artificial Intelligence, (Vol. 3), Barcelona, Spain, 1997-2002.

Panahi, H. & Tavakkoli-Moghaddam, R. (2011). Solving a multi-objective shop scheduling problem by a novel hybrid ant colony optimization. Expert Systems with Applications, 38(3), 2817-2822.

Pinedo, M.L. (2007). Planning and Scheduling in Manufacturing and Services (Vol. 24). Springer Series in Operations Research and Financial Engineering.

Prandtstetter, M. & Raidl, G.R. (2005). A variable neighborhood search approach for solving the car sequencing problem. XVIII Mini EURO Conference on VNS. Tenerife, Spain.

Roshanaei, V., Esfehani, M.M.S. & Zandieh, M. (2010). Integrating non-preemptive open shop scheduling with sequence dependent setup times using advanced meta-heuristics. Expert Systems with Applications, (Vol. 37). 259-266.

Schuurman, P. & Woeginger, G. J. (1999). Approximation algorithms for the multiprocessor open shop scheduling problem. Operations Research Letters, 24(4),157-163.

Sevastianov, S.V. & Woeginger, G.J. (2001). Linear time approximation scheme for the multiprocessor open shop problem. Discrete Applied Mathematics, 114(1). 237-288.

Sevkli, M. & Aydin, M.E. (2006). Variable neighborhood search for job shop scheduling problems. Journal of Software, 1(2). 34-39.

Sha, D. & Hsu, C.Y. (2008). A new particle swarm optimization for the open shop scheduling problem. Computers & Operations Research, (Vol. 35), 3243-3261.

Taillard, E. (1993). Benchmarks for basic scheduling problems. European Journal of operational Research, 64(2), 278-285.

Yang, Q., Sun, J., Zhang, J. & Wang, C. (2006). A hybrid discrete particle swarm algorithm for open-shop problems. Simulated Evolution and Learning Conference, (No. 6), Berlin. 158-165.

Yu, V. F., Lin, S. & Chou, S. (2010). The museum visitor routing problem. Applied Mathematics and computation, 216(3), 719-729.

Zobolas, G., Tarantilis, C. & Ioannou, G. (2009). Solving the open shop scheduling problem via a hybrid genetic-variable neighborhood search algorithm. Cybernetics and Systems Journal, (Vol. 40), 259-285.

9. Annexes

9.1 Annex 1

Table 8 and **Table 9** show the average results obtained with every decoding scheme in each algorithm for every instance for the $O_m/TT_{il}/CMax$.

Table 8. Average results $O_m/TT_{il}/CMax$ (small instances)

PROBLEM	VNS			VNS + SA			VNS + TS		
	Act	ND	Seq	Act	ND	Seq	Act	ND	Seq
04x04_01	72.3	72.0	73.0	72.0	72.0	72.6	72.0	72.0	72.6
04x04_02	74.0	74.0	74.0	74.0	74.0	74.0	74.0	74.0	74.0
04x04_03	76.4	80.0	76.0	76.0	80.0	76.0	76.4	80.0	76.0
04x04_04	79.0	79.0	79.0	79.0	79.0	79.0	79.0	79.0	79.0
04x04_05	79.4	79.0	80.0	79.0	79.0	79.0	79.0	79.0	79.0
04x04_06	78.0	79.0	78.4	78.0	79.0	78.0	78.0	79.0	78.0
04x04_07	71.0	74.0	72.6	71.0	74.0	71.0	71.0	74.0	71.0
04x04_08	68.0	68.0	69.0	68.0	68.0	68.0	68.0	68.0	68.0
04x04_09	83.0	86.0	83.2	83.0	86.0	83.0	83.0	86.0	83.0
04x04_10	76.8	76.0	78.2	76.0	76.0	76.0	76.8	76.0	76.0
05x05_01	93.2	92.0	91.0	92.0	92.0	91.0	90.8	92.0	91.0
05x05_02	84.6	84.0	86.0	86.0	84.0	85.4	85.0	84.6	86.2
05x05_03	98.4	100.0	98.0	99.2	100.0	99.2	98.6	100.0	98.4
05x05_04	94.2	97.2	97.0	94.8	96.4	94.8	94.4	96.4	94.6
05x05_05	93.3	94.0	95.0	92.8	94.0	94.6	94.2	94.0	94.2
05x05_06	93.5	95.0	94.2	93.6	95.0	94.2	93.4	95.0	93.6
05x05_07	97.2	98.0	97.0	96.4	98.0	96.8	96.2	98.0	97.0
05x05_08	91.8	90.5	95.6	92.6	90.0	93.2	91.8	90.0	93.2
05x05_09	93.7	95.0	94.0	94.4	93.0	94.2	93.8	93.0	93.6
05x05_10	94.8	94.0	94.8	95.2	94.0	94.2	95.0	94.0	94.0
07x07_01	126.5	121.6	123.0	127.0	121.6	126.6	125.8	120.8	123.0
07x07_02	126.0	120.2	124.2	127.8	119.6	125.0	125.2	119.4	123.8
07x07_03	134.8	131.6	134.0	138.8	131.0	134.0	136.8	130.4	134.0
07x07_04	132.1	126.4	128.2	135.2	125.6	132.0	131.8	125.4	129.6
07x07_05	131.0	126.6	128.6	132.4	125.4	131.6	131.2	126.0	129.8
07x07_06	136.3	128.8	134.6	137.0	128.0	135.6	134.8	128.2	134.6
07x07_07	127.6	121.0	126.0	130.8	121.8	126.0	128.8	119.8	126.0
07x07_08	129.7	125.0	127.4	132.0	123.8	129.6	129.8	124.2	127.6
07x07_09	126.2	122.6	125.0	129.8	122.6	129.4	127.6	122.0	127.6
07x07_10	120.8	118.0	121.8	126.6	117.4	123.8	122.6	117.0	122.0

Table 9. Average results $O_m/TT_{il}/CMax$ (large instances)

PROBLEM	VNS			VNS + SA			VNS + TS		
	Act	ND	Seq	Act	ND	Seq	Act	ND	Seq
10x10_01	188.8	179.2	190.4	195.4	182.4	195.6	191.0	179.0	192.6
10x10_02	179.7	166.2	171.8	184.4	165.6	172.0	178.4	165.6	171.4
10x10_03	171.6	165.6	167.4	172.0	166.2	169.8	172.0	165.6	169.0
10x10_04	174.1	162.8	167.0	179.8	163.2	169.8	177.6	162.6	168.2
10x10_05	184.8	174.8	177.4	190.4	173.8	182.0	187.0	173.0	179.4
10x10_06	172.2	161.8	165.4	179.2	161.0	173.8	176.4	160.2	168.6
10x10_07	178.2	165.6	168.6	182.0	164.4	171.0	179.6	165.0	169.8
10x10_08	174.0	169.0	172.6	174.0	169.2	173.8	174.0	168.0	173.2
10x10_09	182.4	169.0	171.2	185.4	169.2	173.4	183.6	168.4	172.0
10x10_10	172.5	161.1	168.2	178.0	161.8	172.0	176.0	160.8	168.2
15x15_01	274.0	252.2	260.8	279.0	253.0	268.0	277.4	253.8	265.0
15x15_02	264.0	247.4	249.6	264.0	250.0	253.0	263.6	248.2	252.4
15x15_03	268.4	241.0	248.4	273.2	242.8	249.0	271.8	243.0	248.8
15x15_04	283.6	253.0	254.4	286.8	255.2	257.0	285.6	252.0	257.0
15x15_05	270.6	243.0	248.4	275.0	245.2	249.0	273.6	242.4	248.4
15x15_06	268.8	243.4	246.2	272.4	244.4	248.0	273.8	242.2	247.4
15x15_07	270.0	249.2	263.6	273.4	252.8	273.4	273.4	248.6	272.4
15x15_08	264.8	242.0	246.6	265.0	244.8	251.0	265.0	243.0	248.2
15x15_09	269.4	240.4	246.4	271.0	242.6	247.0	271.0	239.8	247.0
15x15_10	277.8	247.6	261.8	278.2	252.4	268.0	280.8	249.2	262.0
20x20_01	350.8	313.2	318.0	351.0	316.2	320.0	351.0	315.2	319.4
20x20_02	362.6	329.2	330.2	370.4	329.8	332.0	370.6	329.8	331.6
20x20_03	366.6	320.8	328.4	369.0	325.0	332.0	367.2	322.4	331.6
20x20_04	369.6	328.4	333.6	377.4	329.4	336.0	374.2	330.0	335.8
20x20_05	364.0	321.0	326.2	367.0	323.8	329.0	370.0	322.4	327.8
20x20_06	357.8	319.8	334.6	357.2	325.2	338.0	358.0	322.0	338.0
20x20_07	375.4	336.0	351.6	373.8	339.8	357.8	376.0	338.6	353.4
20x20_08	356.4	315.4	323.2	357.0	318.8	324.0	356.4	318.4	324.0
20x20_09	349.6	315.8	325.4	351.8	316.6	326.0	352.0	316.8	325.8
20x20_10	365.2	327.2	335.4	366.2	331.0	339.0	368.0	328.4	338.8

9.2 Annex 2

Table 10 and **Table 11** show the average results obtained with every decoding scheme in each algorithm for every instance for the $FO_m/TT_{il}/\sum C_{j0}$.

Table 10. Experimental results $FO_m/TT_{il}/\sum C_{j0}$ (small instances)

PROBLEM	VNS			VNS + SA			VNS + TS		
	Act	ND	Seq	Act	ND	Seq	Act	ND	Seq
04x04_01	228.0	228.0	228.0	228.0	228.0	228.0	228.0	228.0	228.0
04x04_02	291.0	291.0	289.4	291.4	291.0	287.2	291.8	291.0	287.4
04x04_03	295.6	297.0	294.0	295.0	297.0	294.0	295.2	297.0	294.0
04x04_04	290.0	294.0	290.0	290.0	294.0	290.0	290.0	294.0	290.0
04x04_05	345.8	357.0	329.2	345.4	357.0	328.4	345.4	357.0	328.4
04x04_06	258.6	260.0	256.0	258.2	260.0	254.8	258.8	260.0	256.2
04x04_07	265.0	279.0	263.8	264.0	279.0	263.0	264.8	279.0	263.2
04x04_08	272.0	272.0	272.0	272.0	272.0	272.0	272.0	272.0	272.0
04x04_09	324.8	334.0	318.2	324.0	334.0	315.8	324.8	334.0	315.4
04x04_10	273.0	287.0	273.0	273.0	287.0	273.0	273.0	287.0	273.0
05x05_01	422.8	425.0	417.4	422.0	419.4	418.0	421.2	418.0	417.8
05x05_02	374.0	380.0	370.2	370.0	380.6	369.8	376.6	380.0	368.8
05x05_03	456.6	456.0	458.6	455.2	455.0	453.0	454.4	455.0	454.2
05x05_04	459.4	465.0	457.6	461.0	465.0	449.0	457.6	465.0	451.6
05x05_05	477.4	484.8	478.0	477.6	482.8	478.8	478.6	480.0	478.4
05x05_06	432.4	453.0	438.6	432.8	453.2	431.0	434.4	453.0	432.2
05x05_07	450.8	457.6	444.4	448.4	457.0	443.0	450.2	457.0	443.4
05x05_08	459.8	481.4	455.2	457.0	477.0	455.0	459.6	477.2	455.0
05x05_09	489.0	494.0	485.2	488.8	492.0	482.0	488.2	486.0	481.2
05x05_10	466.6	459.0	462.0	465.0	459.0	463.8	465.2	459.0	463.8
07x07_01	867.4	850.0	863.0	874.2	851.8	860.2	869.4	849.0	861.0
07x07_02	818.0	784.8	789.0	807.8	786.0	782.2	814.2	775.6	787.4
07x07_03	925.2	903.0	911.0	919.2	907.0	902.6	917.8	900.4	902.2
07x07_04	869.6	854.4	857.6	872.2	853.4	862.8	867.8	852.6	857.4
07x07_05	808.4	794.2	803.2	802.8	792.0	794.0	813.8	783.8	793.6
07x07_06	896.2	881.2	883.0	899.2	884.6	882.0	904.8	880.2	882.6
07x07_07	860.8	832.2	842.6	849.0	837.6	836.2	860.6	828.8	839.0
07x07_08	832.6	798.2	814.4	825.2	797.6	803.8	830.8	792.4	808.4
07x07_09	834.0	813.4	826.6	835.4	819.2	824.0	841.8	811.0	826.6
07x07_10	754.2	730.6	741.0	750.8	729.6	740.0	751.4	726.6	740.6

Table 11. Experimental results $FO_m/TT_{il}/\sum C_{j0}$ (large instances)

PROBLEM	VNS			VNS + SA			VNS + TS		
	Act	ND	Seq	Act	ND	Seq	Act	ND	Seq
10x10_01	1735.4	1608.2	1657.0	1723.0	1611.6	1658.4	1732.0	1606.2	1658.4
10x10_02	1693.6	1574.6	1593.2	1681.6	1588.8	1593.8	1683.8	1576.8	1601.4
10x10_03	1744.4	1634.8	1650.2	1744.0	1638.2	1664.0	1753.8	1617.4	1658.2
10x10_04	1664.0	1569.4	1612.0	1676.6	1578.0	1607.0	1680.8	1570.0	1604.2
10x10_05	1753.0	1664.6	1693.8	1743.0	1679.4	1685.0	1757.6	1671.0	1692.6
10x10_06	1625.0	1507.8	1519.8	1618.8	1508.4	1527.4	1603.2	1506.2	1529.4
10x10_07	1678.4	1569.2	1600.4	1677.0	1578.0	1599.2	1686.6	1567.2	1598.2
10x10_08	1657.6	1530.0	1541.2	1656.2	1535.8	1541.2	1639.8	1520.0	1541.8
10x10_09	1678.6	1592.0	1611.6	1685.6	1597.4	1615.4	1678.2	1586.8	1624.2
10x10_10	1640.4	1517.2	1552.0	1642.6	1518.0	1550.4	1634.8	1513.0	1551.4
15x15_01	3869.6	3474.4	3455.2	3875.6	3501.6	3456.6	3861.4	3478.4	3465.0
15x15_02	3860.8	3466.4	3537.2	3827.0	3480.8	3537.8	3822.6	3464.2	3538.4
15x15_03	3921.8	3512.8	3571.8	3900.4	3542.0	3576.6	3872.2	3516.2	3579.0
15x15_04	3785.6	3418.2	3441.8	3768.8	3434.4	3438.4	3774.8	3420.2	3443.0
15x15_05	3756.6	3391.2	3422.2	3743.8	3402.2	3427.2	3745.8	3387.2	3429.0
15x15_06	3759.8	3359.8	3374.8	3758.4	3387.6	3388.4	3717.6	3352.6	3400.6
15x15_07	3675.8	3431.6	3463.4	3690.6	3467.6	3449.8	3687.2	3433.4	3477.0
15x15_08	3755.2	3375.0	3385.6	3733.0	3394.2	3385.8	3731.4	3369.6	3387.8
15x15_09	3799.8	3464.8	3475.4	3745.8	3498.2	3470.4	3766.4	3448.4	3488.4
15x15_10	3879.0	3504.0	3552.8	3893.8	3527.8	3554.0	3883.8	3491.2	3554.0
20x20_01	6756.2	6006.0	5978.0	6746.0	6070.0	5992.0	6749.2	5963.4	5992.0
20x20_02	6794.8	6077.8	5937.6	6786.2	6084.4	5947.8	6797.2	5998.4	5947.2
20x20_03	6667.8	6283.4	6269.8	6643.4	6302.0	6273.4	6667.4	6263.8	6302.0
20x20_04	6727.4	6069.0	6087.6	6833.2	6104.6	6092.0	6836.4	6113.2	6098.6
20x20_05	7127.0	6367.6	6406.0	7124.6	6391.6	6403.8	7152.4	6377.0	6413.0
20x20_06	6824.6	5888.6	5900.6	6888.4	5906.0	5906.0	6891.4	5905.0	5906.0
20x20_07	6616.8	5949.0	5977.8	6620.0	5956.8	5988.0	6609.0	5937.8	5988.0
20x20_08	7103.0	6431.0	6285.0	7076.8	6445.2	6290.4	7085.6	6388.6	6290.0
20x20_09	6855.6	6263.4	6105.4	6798.8	6204.2	6102.4	6756.2	6175.6	6133.0
20x20_10	6879.8	6158.6	6079.4	6868.0	6149.0	6084.6	6889.2	6167.4	6083.4