A Matheuristic Based on Lagrangian Relaxation for the Multi-Activity Shift Scheduling Problem

Noberto A. Hernández-Leandro, Vincent Boyer,
M. Angélica Salazar-Aguilar, Louis-Martin Rousseau

Abstract

The multi-activity shift scheduling problem involves assigning a sequence of activities to a set of employees. In this paper, we consider the variant where the employees have different qualifications and each activity must be performed in a specified time window; i.e., we specify the earliest start period and the latest finish period. We propose a matheuristic in which Lagrangian relaxation is used to identify a subset of promising shifts, and a restricted set covering problem is solved to find a feasible solution. Each shift is represented by a context-free grammar. Computational tests are carried out on two sets of instances from the literature. For the first set, the matheuristic finds a solution with an optimality gap less than 0.01% for 70% of the instances and improves the best-known solution for 16% of them; for the second set, the matheuristic reaches the best-known solutions for 55% of the instances and finds better solutions for 37.5% of them.

Keywords: Scheduling, Shift Scheduling Problem, Context-Free Grammar, Lagrangian Relaxation, Matheuristic

1 Introduction

For many companies, designing a schedule for their employees is complex. An efficient schedule impacts directly on the quality of customer service, the human resource management, and the employee satisfaction. The constraints involved in the assignment of shifts to employees include employee availability, qualifications, and preferences; company policies; the number of activities; and collective agreements. Additionally, the assignment of the activities depends on the forecast demand over the planning horizon. A shortage of employees for a particular activity may result in customer losses, and an excess number of employees performing the same activity may lead to inactivity.

In this context, we design a matheuristic to efficiently solve the multi-activity shift scheduling problem (MASSP), also known in the literature as the personalized multi-activity shift scheduling problem, which involves assigning a sequence of activities to each employee. The goal is to minimize the total cost of undercovering or overcovering activities in each period of the planning horizon. Each activity has a different demand in each period; the same activity can be assigned to multiple employees and it must be performed in

Noberto A. Hernández-Leandro (Corresponding author), Graduate Program in Systems Engineering. Universidad Autónoma de Nuevo León, San Nicolás de los Garza, Nuevo León, Mexico
E-mail: norberto.hernandezlnd@uanl.edu.mx

Vincent Boyer , Graduate Program in Systems Engineering. Universidad Autónoma de Nuevo León, San Nicolás de los Garza, Nuevo León, Mexico
E-mail: vincent.boyer@uanl.edu.mx

M. Angélica Salazar-Aguilar, Graduate Program in Systems Engineering. Universidad Autónoma de Nuevo León, San Nicolás de los Garza, Nuevo León, Mexico
E-mail: maria.salazaragl@uanl.edu.mx
Louis-Martin Rousseau, Centre interuniversitaire de recherche sur les réseaux d'entreprise, la logistique et le transport. Université de Montréal, C.P. 6128, succursale Centre-ville. Montréal (Québec), Canada H3C 3J7
E-mail: Louis-Martin.Rousseau@cirrelt.ca

a specified time window. Other side constraints such as employee availability, preferences, and qualifications are taken into account.

To the best of our knowledge, few researchers have addressed this problem. Demassey et al. [12] solve a set covering problem (SCP) and use a column generation approach, where the subproblems are modeled using automatas and constraint programming. Lequy et al. [15] consider the case where the shifts are built a priori. They propose three integer programming models, a branch and bound method for small instances, and a rolling-horizon heuristic for large instances. Quimper & Rousseau [17] use formal languages and context-free grammars to model the constraints of each shift; they design two operators within a large-neighborhood search. Dahmen & Rekik [11] propose a hybrid method based on tabu search: a branch and bound algorithm is used in the improvement, intensification, and diversification phases of the procedure. Côté et al. [8, 9, 10] present two grammar-based models and a column generation embedded into a branch-and-price approach. Computational tests of [10] with the instances of Demassey et al. [12] and Lequy et al. [15] show that their approach improves the solutions reported in the literature. The branch-and-price algorithm is able to solve instances with up to 100 employees and 10 activities over a planning horizon of 7 days, however in the worst case it takes more than two hours to obtain a solution with a relative gap of 1%. Finally, Restrepo et al. [18] present a column generation algorithm coupled with an auxiliary shortest path problem with resource constraints for an application in Bogota, Colombia, where employees must be assigned to car parks.

Elahipanah et al. [13] present a two-phase heuristic for a variant of the MASSP in which multiple tasks are considered. The first phase solves a mixed integer linear model to generate partial shifts by assigning the tasks, and the second uses a rolling-horizon procedure to assign the activities. Lequy et al. [16] propose a two-stage heuristic for the multi-task variant. The first phase assigns the tasks using a mixed integer linear model, and the second assigns activities and reassigns the tasks using a column generation heuristic. Boyer et al. [4] present an extension of the branch-and-price algorithm of Côté et al. [10] for the MASSP with multiple tasks. The authors compare two formulations for the precedence constraints on the tasks and three branching strategies.

Lagrangian relaxation has been extensively used to solve the SCP; the derived reduced costs are often used to discard variables. Balas and Ho [2] design a branch and bound for the SCP based on the Lagrangian relaxation of the problem. Balas and Carrera [1] use the same approach to derive a heuristic: at each iteration of the branch and bound algorithm, variables are heuristically fixed to reduce the problem size. Ceria et al. [7] and Caprara et al. [5] also investigate different diving strategies built on the solution of the Lagrangian relaxation.

Umetani and Yagiura [20] solve the SCP using a variant of column generation, called the shifting method (Bixby et al. [3]). This approach uses a greedy algorithm to produce feasible solutions and the Lagrangian multipliers to fix variables. Caserta [6] proposes a tabu search algorithm that uses Lagrangian relaxation in the intensification phase to fix variables.

We propose a matheuristic based on context-free grammars and Lagrangian relaxation, where the set of shifts explored by the subgradient method is integrated into the SCP to find a high-quality solution. We perform computational experiments on a large set of instances from Demassey et al. [12] and Lequy et al. [15]. The results are compared to the best-known solutions in the literature which, to the best of our knowledge, are due to Côté et al. [10]. Our approach also provides dual bounds that can be used to measure the quality of the solutions. Furthermore, the proposed procedure is able to find good quality solutions in a shorter processing time than the one reported by Côté et al. [10].

The paper is organized as follows. We describe the MASSP in Section 2 and discuss the context-free grammar model used to represent the feasible shifts. Section 3 presents the solution method, and Section 4 shows the experimental results. Section 5 provides concluding remarks.

2 Multi-Activity Shift Scheduling Problem

The MASSP assigns shifts $s \in \Omega^e$ to employees $e \in E$, where Ω^e represents the set of feasible shifts for employee e. The objective is to cover at minimum cost the demand b_{ia} for each activity $a \in A$ in period $i \in I$. We assume that the employees have differing availabilities and qualifications (these determine the set of activities that the employee is able to perform). The set of feasible shifts is determined by the characteristics of each employee and the company policies. Shift s for employee e is feasible if e performs only activities

for which he/she is qualified and s satisfies the duration constraints and includes the required rest periods. Moreover, each activity in s must be performed in a given time window.

The planning horizon I is discretized into periods of equal length. Shift s is represented by the sequence of activities that the employee performs; this sequence can include lunch periods or breaks. We associate a cost $c_s^e \geq 0$ with each $s \in \Omega^e$. This includes the cost of transitions between activities during a shift. Furthermore, we allow undercovering and overcovering of the activities, with associated penalties denoted by c_{ia}^u and c_{ia}^o, respectively. Parameter $\delta_{ias}^e \in \{0,1\}$ indicates whether or not activity a is in shift s in period i for employee e.

Côté et al. [10] propose the following SCP:

$$\textbf{(SCM)}\ \text{Min} \quad z = \sum_{e\in E}\sum_{s\in\Omega^e} c_s^e x_s^e + \sum_{i\in I}\sum_{a\in A}(c_{ia}^u u_{ia} + c_{ia}^o o_{ia}) \tag{1}$$

subject to

$$\sum_{e\in E}\sum_{s\in\Omega^e} \delta_{ias}^e x_s^e + u_{ia} - o_{ia} = b_{ia} \qquad \forall i \in I, a \in A \tag{2}$$

$$\sum_{s\in\Omega^e} x_s^e = 1 \qquad \forall e \in E \tag{3}$$

$$x_s^e \in \{0,1\} \qquad \forall e \in E, s \in \Omega^e \tag{4}$$

$$u_{ia} \geq 0 \qquad \forall i \in I, a \in A \tag{5}$$

$$o_{ia} \geq 0 \qquad \forall i \in I, a \in A \tag{6}$$

Where:

- $x_s^e = 1$ if shift s is assigned to employee e, and 0 otherwise.
- u_{ia} represents the undercovering of activity a in period i.
- o_{ia} represents the overcovering of activity a in period i.

The objective (1) minimizes the cost of assigning a shift to an employee and the costs of overcovering and undercovering. Constraints (2) ensure that the demand for each activity a in each period i is satisfied. Constraints (3) ensure that a shift is assigned to each employee.

To model the set of feasible shifts Ω^e, we use a context-free grammar, as suggested by Côté et al. [10]. We summarize the model below; for further details, see [14]. Note that the context-free grammar is used to represent the set Ω^e, and this set is not explicitly generated.

A context-free grammar G is defined by the tuple (Σ, N, P, S) where

- Σ is an alphabet of symbols, called terminals;
- N is a set of nonterminal symbols;
- P is a set of productions of the form $X \to \alpha$, where $X \in N$ and α is a sequence of terminal and nonterminal symbols;
- S is the starting nonterminal symbol.

The context-free grammar generates a set of sequences called language, where these sequences are named words and are conformed only by terminal symbols. A production rule is a relationship between the nonterminals symbols and a sequence of terminal/nonterminal symbols. In this context, the set P defines whether a word belongs to the language or not, by checking if it can be derived from S using these production rules. The nonterminal symbols can be seen as transition symbols given that, depending on the productions, these symbols produce a subsequence of terminal and/or nonterminal symbols.

A shift can be represented as a sequence of terminal symbols giving the activities to be performed by the employee. The position of the symbol in the sequence corresponds to the period in which the corresponding activity is performed. Consequently, starting from the initial symbol S, the set of productions allows us to

derive a set of sequences of symbols corresponding to the feasible shifts of an employee. Côté et al. [10] use a directed acyclic graph (DAG) to represent a context-free grammar.

For example, consider an employee who can perform two activities (a_1 and a_2) and a shift with 4 periods of 1 hour. The first activity is a_1, and no activity can be assigned for less than 2 hours. The feasible shifts for this employee can be modeled by the following context-free grammar G:

- $\Sigma = \{a_1, a_2\}$;
- $N = \{J_1, J_2, A_1, A_2\}$;
- P is defined by:

$$S \rightarrow_{[4,4]} J_1J_1|J_1J_2$$
$$J_1 \rightarrow A_1A_1 \qquad J_2 \rightarrow A_2A_2$$
$$A_1 \rightarrow A_1a_1|a_1 \qquad A_2 \rightarrow A_2a_2|a_2.$$

Where the symbol "|" is the logical OR operator and "$\rightarrow_{[l,r]}$" indicates that the subsequences of terminals derived from this production must have a length between l and r. For instance, $S \rightarrow_{[4,4]} J_1J_1|J_1J_2$ means that from S we can produce the sequence J_1J_1 or the sequence J_1J_2, and all sequences that can be derived from S with these production rules should yield to a sequence of exactly four terminals.

To generate sequences of length four, we must apply the production rules starting from S until the sequence contains only terminal symbols. Table 1 shows the derivations of the shifts of length four; the columns P and *Result* give the production rule and the result of the application of the rule, respectively. For example, as shown in Table 1a, from the symbol S, the sequence J_1J_1 is produced from the production rule $S \rightarrow_{[4,4]} J_1J_1$; then the production $J_1 \rightarrow A_1A_1$ is applied twice in order to generate the sequence $A_1A_1A_1A_1$; and finally each non-terminal A_1 gives a terminal a_1 by executing successively the rule $A_1 \rightarrow a_1$, which results in the sequence $a_1a_1a_1a_1$.

Furthermore, the production rules used to generate a sequence can be represented by a parse tree. The parse trees of the shifts $a_1a_1a_1a_1$ and $a_1a_1a_2a_2$ from Table 1 are given in Figure 1.

P	**Result**
-	S
$S \rightarrow J_1J_1$	J_1J_1
$J_1 \rightarrow A_1A_1$	$A_1A_1J_1$
$J_1 \rightarrow A_1A_1$	$A_1A_1A_1A_1$
$A_1 \rightarrow a_1$	$a_1A_1A_1A_1$
$A_1 \rightarrow a_1$	$a_1a_1A_1A_1$
$A_1 \rightarrow a_1$	$a_1a_1a_1A_1$
$A_1 \rightarrow a_1$	$a_1a_1a_1a_1$

(a)

P	**Result**
-	S
$S \rightarrow J_1J_2$	J_1J_2
$J_1 \rightarrow A_1A_1$	$A_1A_1J_2$
$J_2 \rightarrow A_2A_2$	$A_1A_1A_2A_2$
$A_1 \rightarrow a_1$	$a_1A_1A_2A_2$
$A_1 \rightarrow a_1$	$a_1a_1A_2A_2$
$A_2 \rightarrow a_2$	$a_1a_1a_2A_2$
$A_2 \rightarrow a_2$	$a_1a_1a_2a_2$

(b)

Table 1 – Derivations of the shifts $a_1a_1a_1a_1$ (a) and $a_1a_1a_2a_2$ (b).

Finally, all the parse trees of a grammar can be embedded in a DAG, denoted Γ. Figure 2 gives the DAG Γ for G; it is an AND/OR graph where the A nodes are AND-nodes and the O nodes are OR-nodes. The node $A_{ij}^{p,n}$ is an AND-node that uses the production rule p to generate the $n-th$ sequence starting from the position i with length j, note that any node $A_{ij}^{p,n}$ has as many children as the number of symbols of the subsequence generated by the production rule p. Besides, the node O_{ij}^{X} is an OR-node corresponding to the terminal/nonterminal symbol X, which produces a sequence starting at position i with length j. The inner nodes represent production rules and nonterminal symbols, whilst the leaves of the tree correspond to the terminal symbols at each position of the produced word. In an AND/OR graph, one should start from the root and select only one outgoing arc of an OR-node, and all the outgoing arcs of an AND-node to find a path from the root node to the leaves. In the example from Figure 2, the sequences $a_1a_1a_1a_1$ and $a_1a_1a_2a_2$ can be derived by choosing one of the outcoming arcs of the root O_{14}^{S}.

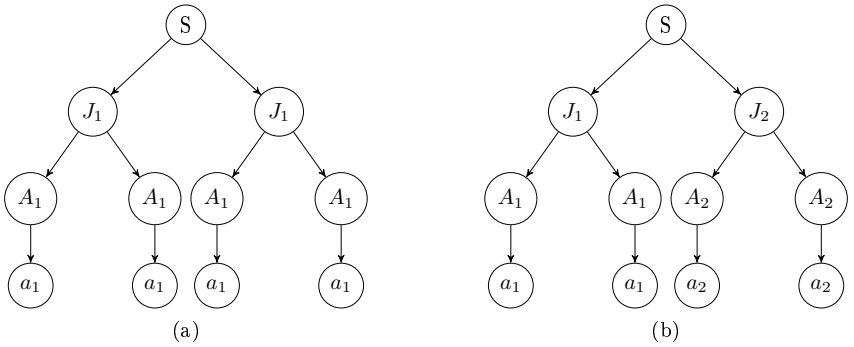

Figure 1 – Parse trees of the shifts $a_1a_1a_1a_1$ (a) and $a_1a_1a_2a_2$ (b).

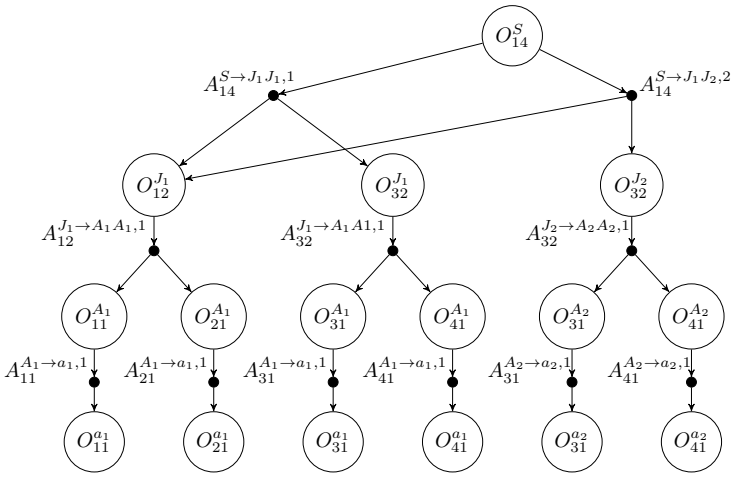

Figure 2 – DAG Γ.

3 Solution Method

In this section, we present our matheuristic (MH) based on Lagrangian relaxation and the SCP. We use the Lagrangian relaxation of SCM to identify promising shifts for the employees, and then we obtain the best combination of these shifts by solving a restricted SCP. This restricted SCP contains only a subset of the feasible shifts for an employee and hence has fewer variables than SCM.

3.1 Lagrangian Relaxation

The Lagrangian relaxation of the SCM (called $RSCM$) is as follows:

$$\textbf{(RSCM)} \quad \text{Min } z = \sum_{e\in E}\sum_{s\in\Omega^e} c_s^e x_s^e + \sum_{i\in I}\sum_{a\in A}(c_{ia}^u u_{ia} + c_{ia}^o o_{ia}) + \sum_{i\in I}\sum_{a\in A}\lambda_{ia}\left(\sum_{e\in E}\sum_{s\in\Omega^e}\delta_{ias}^e x_s^e + u_{ia} - o_{ia} - b_{ia}\right)$$

subject to

$$\begin{aligned} \sum_{s\in\Omega^e} x_s^e &= 1 && \forall e\in E \\ x_s^e &\in \{0,1\} && \forall e\in E, s\in\Omega^e \\ u_{ia} &\geq 0 && \forall i\in I, a\in A \end{aligned}$$

$$\begin{array}{rcll} o_{ia} & \geq & 0 & \forall i \in I, a \in A \\ \lambda_{ia} & \in & \mathbb{R} & \end{array}$$

where λ_{ia} are the Lagrangian multipliers that can take any real value because of the nature of the relaxed constraints.

We can simplify $RSCM$ to the following model:

$$\begin{aligned} \text{Min } z \quad = \quad & \sum_{i \in I} \sum_{a \in A} \left((c^u_{ia} + \lambda ia)\, u_{ia} + (c^o_{ia} - \lambda_{ia})\, o_{ia} \right) \\ & + \sum_{e \in E} \sum_{s \in \Omega^e} \left(c^e_s + \sum_{i \in I} \sum_{a \in A} \lambda_{ia} \delta^e_{ias} \right) x^e_s \\ & - \sum_{i \in I} \sum_{a \in A} \lambda_{ia} b_{ia} \end{aligned}$$

subject to

$$\begin{array}{rcll} \displaystyle\sum_{s \in \Omega^e} x^e_s & = & 1 & \forall e \in E \\ x^e_s & \in & \{0,1\} & \forall e \in E, s \in \Omega^e \\ u_{ia} & \geq & 0 & \forall i \in I, a \in A \\ o_{ia} & \geq & 0 & \forall i \in I, a \in A \\ \lambda_{ia} & \in & \mathbb{R} & \end{array}$$

We solve this model using the classical subgradient method. When the Lagrangian multipliers λ_{ia} are fixed, the relaxed model can be decomposed into two subproblems. The first ($SubP1$) determines the overcovering and undercovering of each activity in each period; the second ($SubP2$) assigns a shift to each employee.

$$\begin{aligned} \textbf{(SubP1}(\Lambda)\textbf{)} \ \text{Min } z_1 \quad = \quad & \sum_{i \in I} \sum_{a \in A} (c^u_{ia} + \lambda_{ia})\, u_{ia} + \sum_{i \in I} \sum_{a \in A} (c^o_{ia} - \lambda_{ia})\, o_{ia} \\ & - \sum_{i \in I} \sum_{a \in A} \lambda_{ia} b_{ia} \end{aligned}$$

subject to

$$\begin{array}{rcll} u_{ia} & \geq & 0 & \forall i \in I, a \in A \\ o_{ia} & \geq & 0 & \forall i \in I, a \in A \end{array}$$

$$\textbf{(SubP2}(\Lambda)\textbf{)} \ \text{Min } z_2 \quad = \quad \sum_{e \in E} \sum_{s \in \Omega^e} \left(c^e_s + \sum_{i \in I} \sum_{a \in A} \lambda_{ia} \delta^e_{ias} \right) x^e_s$$

subject to

$$\begin{array}{rcll} \displaystyle\sum_{s \in \Omega^e} x^e_s & = & 1 & \forall e \in E \\ x^e_s & \in & \{0,1\} & \forall e \in E, s \in \Omega^e \end{array}$$

At each step of the subgradient method, we solve $SubP1$ and $SubP2$ and compute the new Lagrangian multipliers λ_{ia}. The subgradient method iterates until we reach a maximum number of iterations or the gap is below a given threshold. Note that we can easily derive a feasible solution for the original problem SCM from the solution of RSCM, obtained at each step of the algorithm, by using the shift assignment returned by $SubP2$ and recalculating the corresponding under and over covering of each activity as stated in constraints 2.

3.1.1 Solving the Subproblems

$SubP1$ can easily be solved because the undercovering and overcovering of each activity in each period are bounded by the problem structure. Maximum undercovering of activity a in period i occurs when no employees perform activity a in period i. Maximum overcovering occurs when all the available employees perform activity a in period i. Algorithm 1 gives the solution procedure for $SubP1$.

Algorithm 1 Solution method for SubP1.

Input: Matrix Λ.
Output: Solution $[U, O]$.

```
for all i ∈ I do
    for all a ∈ A do
        if (c^u_ia + λ_ia) < 0 then
            u_ia ← b_ia
        else
            u_ia ← 0
        end if
        if (c^o_ia − λ_ia) < 0 then
            o_ia ← |E| − b_ia
        else
            o_ia ← 0
        end if
    end for
end for
return [U, O]
```

To solve $SubP2$, we use the DAG Γ model for the shifts and apply the dynamic programming algorithm used by Côté et al. [10] for their pricing subproblems. The algorithm starts by labeling each leaf O^a_{i1}, representing the activity a at period i, to its corresponding Lagrangian multiplier λ_{ia}. Indeed, the Lagrangian multiplier λ_{ia} represents the cost of performing activity a in period i. The costs are then backtracked in the tree: an OR-node is labeled with the minimum cost of its children, and an AND-node is labeled with the sum of the costs of its children. When the root is labeled, we obtain the path that generates the sequence of activities with minimum cost. The whole procedure is presented in Algorithm 2.

For instance, consider the DAG Γ from Figure 2 with the following values of the Lagrangian multipliers for two activities and four positions:

$$\Lambda = \begin{pmatrix} 1 & 0 \\ -2 & 0 \\ 0 & 1 \\ 1 & -1 \end{pmatrix}$$

Figure 3 shows the cost propagation in the DAG Γ; notice that the best path is highlighted in bold. In this case, the leaf $O^{a_1}_{11}$ is labeled with $\lambda_{1a_1} = 1$, $O^{a_1}_{21}$ with $\lambda_{2a_1} = -2$, and so on. In particular, during the backtracking, the AND-node $A^{J_1 \to A_1 A_1, 1}_{12}$ takes the label -1 since it is the sum of the costs of its two incident OR-nodes $O^{A_1}_{11}$ and $O^{A_1}_{21}$; and the OR-node O^S_{14} (i.e. the root node) takes the label -1 since it is the minimum value of the costs of its incident AND-nodes $A^{S \to J_1 J_1, 1}_{14}$ and $A^{S \to J_1 J_1, 2}_{14}$.

Once the root O^S_{14} is reached, we can then identify the propagation that leads to the solution and then the sequence with minimum cost, that is to say $a_1 a_1 a_2 a_2$. Note that, when transition costs are considered, i.e. the cost for an employee to shift from one activity to another, they are associated with any AND-node leading to a transition (e.g., node $A^{S \to J_1 J_2, 2}_{14}$ in Figure 3) and added to the associated label during the propagation procedure.

This procedure gives the shift with the lowest cost for each employee according to the Lagrangian multipliers. Constraints (3) are satisfied trivially since we choose one shift for each employee. This approach solves the $SubP2$ exactly.

Algorithm 2 Dynamic Programming Algorithm

Input: DAG Γ and Matrix Λ.
Output: A shift s with the smallest cost.

$N \leftarrow \emptyset$
for all node k in Γ **do**
  if $k = O^a_{i1},\ i \in I,\ a \in A$ **then**
    $c_k \leftarrow \lambda_{ia}$
    $N \leftarrow N \cup \{k\}$
  else
    $c_k \leftarrow +\infty$
  end if
end for
while N $\neq \emptyset$ **do**
  Get a node $k \in N$
  $N \leftarrow N \setminus \{k\}$
  Let C be the set of children of node k in Γ
  if k is a OR-Node **then**
    $c_k \leftarrow \min\{c_j \mid j \in C\}$
  else
    $c_k \leftarrow \sum_{j \in C} c_j$
  end if
  Add to N all the parents of node k in Γ
end while
Let r be the root node of Γ
Get the path $\mathcal{P}$ in Γ that yields to the cost c_r
Build the shift s from the leaves in $\mathcal{P}$
return s

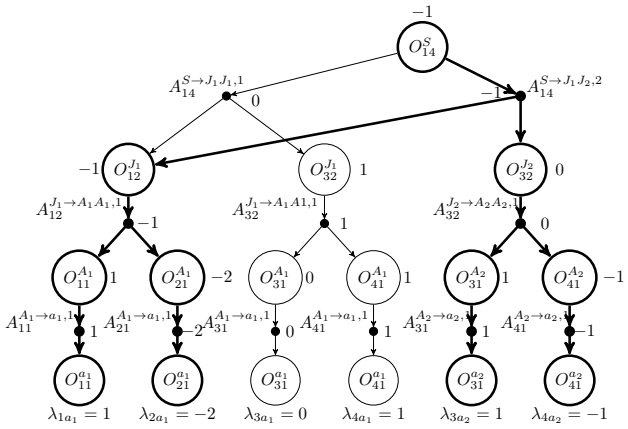

Figure 3 – Cost propagation for DAG Γ.

3.1.2 Updating the Lagrangian Multipliers

After both subproblems have been solved, we recompute the Lagrangian multipliers using the subgradient method introduced by Shor [19]. We use the following equation to update the Lagrangian multipliers at the kth iteration (λ^k_{ia}):

$$\lambda^k_{ia} = \lambda ia^{k-1} + \left[\frac{s^{k-1}_{ia} \cdot \epsilon_{k-1} \left(f_{k-1} - f'_{k-1} \right)}{\| s^{k-1}_{ia} \|_2} \right] \tag{7}$$

Here $\epsilon_{k-1} \in (0, 2]$, and f_{k-1} and f'_{k-1} are the best bounds for SCM and $RSCM$, respectively, found by the $(k-1)$th iteration. If $x_s^{e(k-1)}$, $e \in E$, $s \in \Omega_e$ denotes the $RSCM$ solution obtained at the $(k-1)$th iteration, then

$$s_{ia}^{k-1} = \sum_{e \in E} \sum_{s \in \Omega^e} \delta_{ias}^{e(k-1)} x_s^{e(k-1)} + u_{ia}^{k-1} - o_{ia}^{k-1} - b_{ia}.$$

The parameter ϵ_1 is fixed to two, and we divide it by two after a given number K of iterations without improvement of the relaxed bound value. The procedure stops at iteration k when $\epsilon_k < \varepsilon$, where ε is small enough that updating the Lagrangian multipliers does not lead to significant changes in the solution.

The shifts generated at each step of the Lagrangian relaxation are also feasible for SCM. In our experiments, the quality of this solution is poor, but it can be used to update the Lagrangian multipliers. Furthermore, since the Lagrangian relaxation provides a lower bound for SCM, it can be used to compute the optimality gap and provides information about the quality of the solution found by MH. The complete procedure used to solve RSCM is summarized in Algorithm 3.

Algorithm 3 Subgradient Method.

Input: RSCM
Output: Set Ω

$\Omega \leftarrow \emptyset$
$\epsilon_R \leftarrow 2$
$LB \leftarrow -\infty$
$\Lambda \leftarrow 0_{|I|,|A|}$ {All the dual variables are set to 0}
while $\epsilon_R > \varepsilon$ **do**
    $LB1 \leftarrow SubP1(\Lambda)$
    $LB2 \leftarrow SubP2(\Lambda)$
    Let S be the set of shifts assigned to the employees in the solution of $SubP2(\Lambda)$
    $\Omega \leftarrow \Omega \cup S$
    $LB \leftarrow \max(LB, LB1 + LB2)$
    if LB has not changed in K iterations **then**
        $\epsilon_R \leftarrow \epsilon_R/2$
    end if
    Update Λ
end while
return Ω

3.2 Matheuristic

We now present a general description of MH. We assume that the Lagrangian relaxation produces a set of good-quality shifts. This allows us to consider only a restricted subset of the feasible shifts and to reduce the complexity of finding a good solution.

The model (SSCM(Ω)), which allows only a subset Ω of the feasible shifts, is as follows:

$$\begin{aligned} \textbf{(SSCM}(\Omega)\textbf{)} \quad \text{Min} \quad z \;=\; & \sum_{e \in E} \sum_{s \in \Omega'_e} c_s^e x_s^e \\ & + \sum_{i \in I} \sum_{a \in A} (c_{ia}^u u_{ia} + c_{ia}^o o_{ia}) \end{aligned}$$

subject to

$$\begin{aligned} \sum_{e \in E} \sum_{s \in \Omega'_e} \delta_{ias}^e x_s^e + u_{ia} - o_{ia} &= b_{ia} && \forall i \in I, a \in A \\ \sum_{s \in \Omega'_e} x_s^e &= 1 && \forall e \in E \end{aligned}$$

$$
\begin{array}{rcll}
x_s^e & \in & \{0,1\} & \forall e \in E, s \in \Omega'_e \\
u_{ia} & \geq & 0 & \forall i \in I, a \in A \\
o_{ia} & \geq & 0 & \forall i \in I, a \in A
\end{array}
$$

where $\Omega = \bigcup_{e \in E} \Omega'_e$ and $\Omega'_e \subset \Omega^e$ is the set of feasible shifts for employee e, generated at each iteration of the subgradient method.

It is easier to solve SSCM(Ω) than SCM. This is because the sets Ω'_e are considerably smaller than the sets Ω^e. Hence, we propose to solve SSCM(Ω) exactly using a commercial solver such as CPLEX. A flowchart of the complete procedure is presented in Figure 4.

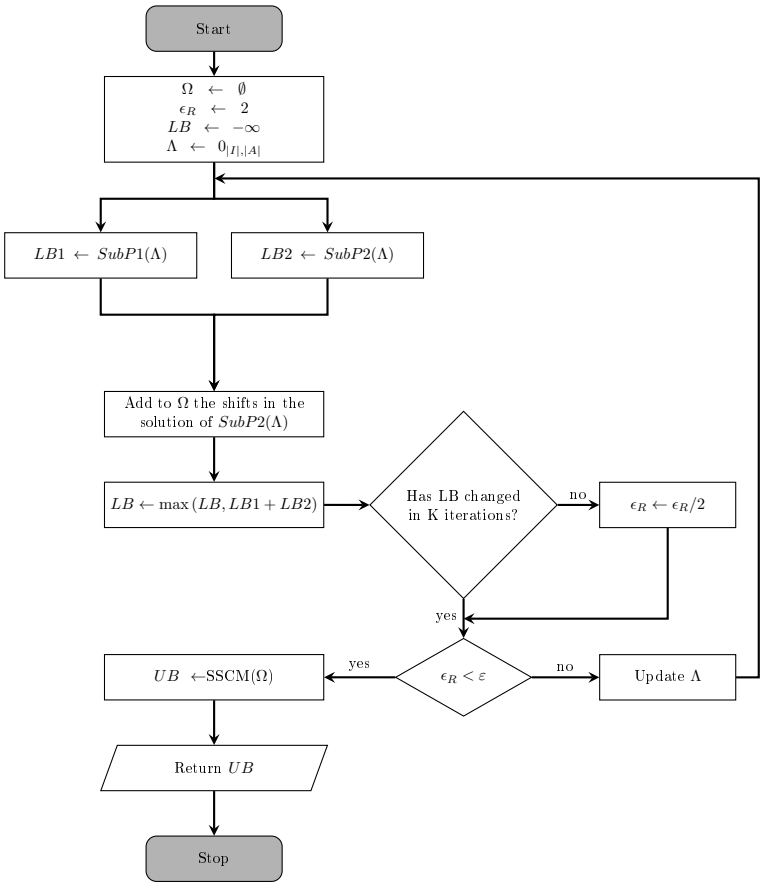

Figure 4 – Flowchart of the Matheuristic

4 Experimental results

We now present our results. We consider two instance sets from the literature: the Demassey instances [12] and the Lequy instances [15]. In the former, the working periods of the employees are not fixed, and in the latter, they are fixed *a priori*. In both set of instances, the planning horizon is divided into periods of 15 minutes. We compare our results with the ones reported by Côté et al. [10], which are the best solutions of the literature for both benchmarks, and have been obtained from a Branch and Price algorithm (BP) executed on an Intel Xeon 2.4 Ghz with 48 GB RAM and using CPLEX 11.2. When our approach reports an equivalent or a better solution than the best-known in the literature, it is highlighted in bold font in the tables.

In the Branch and Price of Côté et al. [10], the linear relaxation of the SCM is used as the restricted master problem (RMP), where only a subset $\tilde{\Omega}_e \subset \Omega_e$ of the feasible shifts for the employee $e \in E$ is considered. The

pricing subproblems are modeled with a DAG Γ for each employee, as detailed in Section 2. As described in Section 3.1.1, the subproblems are solved through a dynamic programing algorithm where the terminals are initialized with their corresponding reduced cost obtained from the dual variables of the RMP. The solution of each subproblem corresponds to the shift with the smallest reduced cost for an employee e and it is added to $\tilde{\Omega}_e$ through a column generation approach. Their branching strategy consists in forbidding some activities for an employee at a specific period based on the fractional values of the variables in the RMP solution. The Branch and Price stops when the optimality gap is below 1% or when a time limit of two-hours is reached.

Additionally, we also compare our heuristic with a Column Generation Heuristic (CGH) where the Lagrangian relaxation in the MH is replaced by a column generation as proposed by Côté et al. [10]. The CGH uses the last RMP returned by the column generation approach to build the SSCM(Ω). Then, SSCM(Ω) is solved as in the MH.

The experiments were performed on an Intel Xeon E5-2687W 3.1 GHz with 64 GB RAM. MH was coded in C++, and SSCM(Ω) was solved using CPLEX 12.6. The Lagrangian relaxation starts with $\epsilon_1 = 2$, and we set $\epsilon = 0.0001$ for the stopping criterion. The Lagrangian multipliers are updated every 375 iterations on the Demassey instances and every 75 iterations on the Lequy instances. We tuned these parameters to obtain the best compromise between computational time and solution quality for each set of instances. CPLEX stops when the optimality gap between the feasible solution and the best Lagrangian bound of SSCM(Ω) is less than 1%, and we terminate the algorithm if 300 s passed with no improvement in the solution.

4.1 Problem Instances

4.1.1 Demassey Instances

We consider 100 instances introduced by Demassey et al. [12]. This set is divided into 10 subsets, each with 10 instances, based on the total number of activities (from 1 to 10). For each instance, the employees are able to perform all the activities and their working periods are not fixed; i.e., they can start working in any period of the planning horizon (one day) and they can perform a part-time or full-time shift. A part-time shift has between 3 and 6 hours of work and includes a break of 30 minutes. A full-time shift has between 6 and 8 hours of work with 2 breaks of 30 minutes and a lunch break of 1 hour. A break (or lunch) is necessary between two different activities.

4.1.2 Lequy Instances

The 40 Lequy instances were introduced by Lequy et al. [15]. In these instances, the employees have qualifications, the working periods are fixed *a priori*, the activities have time windows, and transitions between activities during a shift have a fixed cost.

These instances are divided into 8 classes of 5 instances each. Table 2 provides the details of each class: the first column gives the name of the class, the second gives the number of working days D, the third gives the number of available employees E, and the fourth gives the number of activities A. Class 1 is the same size as class 2, and class 4 is the same size as class 5. However, classes 1 and 4 have larger shifts and employees with more qualifications.

Set of Instances	Days	Employees	Activities
Class 1	1	50	10
Class 2	1	50	10
Class 3	2	75	12
Class 4	7	20	5
Class 5	7	20	5
Class 6	7	50	7
Class 7	7	50	10
Class 8	7	100	15

Table 2 – Classification of the Lequy instances.

4.2 Matheuristic

4.2.1 Demassey Instances

Table 3 summarizes our solutions for the Demassey instances. The columns labeled BP give the results obtained by the branch and price as reported by Côté et al [10]. The columns NbS(0.01%) and NbS(1%) give the number of instances for which the optimality gap was less than or equal to 0.01% and 1%, respectively. For MH, the optimality gap is computed via $100 * (z - z_l)/z_l$ where z is the feasible bound and z_l the Lagrangian bound. The best average results for each class are in bold.

In 70 of the instances MH achieves a gap less than or equal to 0.01%, whereas the BP achieves this in only 54 instances. However, the MH gap is computed with the lower bound obtained by the Lagrangian relaxation, which can obtain better bounds than the linear relaxation. Furthermore, although the MH and BP tests were performed on different machines, the results show that MH has a substantially lower processing time on average.

4.2.2 Lequy Instances

Table 4 shows the MH solutions for each set of instances, where the optimality gap is computed based on the MH solution and the best bound found by the Lagrangian relaxation, and the BP solution is from Côté et al. [10]. The instances where our algorithm reports better or equivalent solutions are in bold.

MH finds better solutions for 15 of the 40 instances and obtains a worse solution for only 3 instances. However, the difference between the MH solution and the best-known solution is the cost of one transition (15). For the remaining instances, our method finds the BP solutions.

Table 5 shows the average gap for each instance set: Gap (MH/LR) is the average gap between the MH bound and the lower bound of the Lagrangian relaxation; Gap (BP/LR) is the average gap between the best bound reported by Côté et al. [10] and the lower bound of the Lagrangian relaxation; Gap (BP/MH) is the average gap between the best bound reported by Côté et al. [10] and the MH bound.

The overall Gap (MH/LR) is 1.56%, and the overall Gap (BP/LR) is 2.00%. This shows the quality of the lower bounds provided by the Lagrangian relaxation. Also, on average, the MH solution is better than the best known solution, with an optimality gap below 2%. This result is confirmed by the value of the overall Gap (BP/MH): -0.41%, which means that the MH is able to find better solutions than the BP.

4.3 Column Generation Heuristic

With the aim of having a thorough evaluation and assessing the performance of our proposed MH, we have implemented a column generation heuristic (CGH) and compared the obtained solutions with the ones reported by the MH. In this section, we present the analysis of this experiment.

Table 6 summarizes the comparison between the solutions reported by the MH and the CGH for the Demmasey instances. The column Gap(CGH/CG) displays the optimality gap reported by CGH, notice that this is the gap between the best found solution and the solution of the linear relaxation provided by the column generation; Gap(MH/CGH) represents the gap between the best found solutions obtained by the MH and the CGH; column NbS($MH \leq CGH$) shows the number of times the MH reported a better or an equivalent solution than the CGH; and the last two columns report the average computation time required by the MH and CGH, respectively. From this table one can notice that the MH is able to find better quality solutions in a shorter computation time than the CGH. Only in two cases, corresponding to the smallest instances, the CGH found better solutions than the MH. Besides, the MH achieves better or equal solutions than CGH in 98 out of 100 cases, with an average gap of 1.8%.

The summary of the results obtained for the Lequy Instances is presented in Table 7. The MH finds a better or an equivalent solution than the CGH in 39 out of 40 instances, with an average gap of 3.73%. In particular, CGH shows its worst performance when solving the largest class of instances (Class 8). Notice that both heuristics report similar optimality gap in all instance classes except Class 8, even though the column generation provides a better lower bound than the Lagrangian relaxation.

5 Conclusions

We have presented a Lagrangian relaxation and a matheuristic for the MASSP. The Lagrangian relaxation is based on the SCP, and we use a subgradient algorithm to solve this problem, taking advantage of the context-free grammar representation of the feasible shifts. In the matheuristic, the shifts explored during the solution of the Lagrangian relaxation are integrated into a restricted SCP to obtain a feasible solution.

Instead of solving the SCP for the MASSP directly, we identify promising shifts through a subgradient process to reduce the number of variables and the complexity of the problem. The solutions for the Demassey instances have an optimality gap of 0.01% for 70% of the instances with a relatively low processing time. On the Lequy instances, MH achieves a gap below 2% with respect to the lower bound of the Lagrangian relaxation. Furthermore, it finds a better solution for 15 of the 40 instances than the one reported in the literature. Besides, a comparison with the CGH (column generation heuristic) shows the efficiency of MH, particularly with the largest instances. For more than 97% of the instances the MH reports a better or an equivalent solution than the CGH within a competitive computation time.

Future work includes the design of an efficient heuristic for the restricted SCP. This would reduce the overall processing time but may also reduce the quality of the bound. We also plan to extend this approach to other classes of shift scheduling problems where, for instance, the shifts are not fixed a priori or a set of tasks must be assigned. These cases have been successfully modeled using context-free grammars. Finally, instances with over three days of planning horizon and flexible shifts generate DAGs containing an important number of nodes that makes them intractable in practice. In order to tackle this class of instances, we look forward to considering models based on constrained networks such as the one proposed by Restrepo et al. [18].

Acknowledgments

The authors are grateful to CONACYT and Polytechnique Montréal for their financial support.

Conflict of Interest

The authors declare that they have no conflicts of interest.

References

References

[1] E. Balas and M. Carrera. A dynamic subgradient-based branch-and-bound procedure for the set-covering problem. *Operations Research*, 44:875–890, 1996.

[2] E. Balas and A. Ho. Set covering algorithms using cutting planes, heuristics, and subgradient optimization: A computational study. *Combinatorial Optimization*, pages 37–60, 1980.

[3] R.E. Bixby, J.W. Gregory, I.J. Lustig, R.E. Marsten, and D.F. Shanno. Very large-scale linear programming: A case study in combining interior point and simplex methods. *Operations Research*, 40:885–897, 1992.

[4] V. Boyer, B. Gendrom, and L.M. Rousseau. A branch-and-price algorithm for the multi-activity multi-task shift scheduling problem. *Journal of Scheduling*, 17:185–197, 2014.

[5] A. Caprara, M. Fischetti, and P. Toth. A heuristic method for the set covering problem. *Operations Research*, 47:730–743, 1999.

[6] M. Caserta. Tabu search-based metaheuristic algorithm for large-scale set covering problems. In *Metaheuristics*, pages 43–63. Springer, 2007.

[7] S. Ceria, P. Nobili, and A. Sassano. A lagrangean-based heuristic for large-scale set covering problems. *Mathematical Programming*, 81:215–228, 1998.

[8] M.C. Côté, B. Gendron, C. Quimper, and L.M. Rousseau. Formal languages for integer programming modeling of shift scheduling problems. *Constraints*, 16:54–76, 2011.

[9] M.C. Côté, B. Gendron, and L.M. Rousseau. Grammar-based integer programming models for multiactivity shift scheduling. *Management Science*, 57:151–163, 2011.

[10] M.C. Côté, B. Gendron, and L.M. Rousseau. Grammar-based column generation for personalized multiactivity shift scheduling. *INFORMS Journal on Computing*, 25:461–474, 2013.

[11] S. Dahmen and M. Rekik. Solving multiactivity personalized shift scheduling problems with a hybrid heuristic. Technical report, CIRRELT, 2012.

[12] S. Demassey, G. Pesant, and L.M. Rousseau. A cost-regular based hybrid column generation approach. *Constraints*, 11:315–333, 2006.

[13] M. Elahipanah, G. Desaulniers, and È. Lacasse-Guay. A two-phase mathematical-programming heuristic for flexible assignment of activities and tasks to work shifts. *Journal of Scheduling*, 16(5):443–460, 2013.

[14] J.E. Hopcroft, R. Motwanu, and J.D. Ullman. Introduction to automata theory, languages and computation. chapter 5, pages 169–216. Addison-Wesley, 2001.

[15] Q. Lequy, M. Bouchard, G. Desaulniers, F. Soumis, and B. Tachefine. Assigning multiple activities to work shifts. *Journal of Scheduling*, 15(2):239–251, 2012.

[16] Q. Lequy, G. Desaulniers, and M.M. Solomon. A two-stage heuristic for multi-activity and task assignment to work shifts. *Computers & Industrial Engineering*, 63(4):831–841, 2012.

[17] C.G. Quimper and L.M. Rousseau. A large neighbourhood search approach to the multi-activity shift scheduling problem. *Journal of Heuristics*, 16:373–392, 2010.

[18] M.I. Restrepo, L. Lozano, and A.L. Medaglia. Constrained network-based column generation for the multi-activity shift scheduling problem. *International Journal of Production Economics*, 140(1):466–472, 2012.

[19] N. Shor. Minimization methods for non-differentiable functions. volume 3 of *Springer Series in Computational Mathematics*, pages 22–47. Springer Berlin Heidelberg, 1985.

[20] S. Umetani and M. Yagiura. Relaxation heuristics for the set covering problem. *Journal of the Operations Research*, 50:350–375, 2007.

	Matheuristic			BP		
Number of Activities	**NbS(0.01%)**	**NbS(1%)**	**Average Time (s)**	**NbS(0.01%)**	**NbS(1%)**	**Average Time* (s)**
1	**7**	**10**	12.01	5	10	62.00
2	**8**	**9**	75.64	6	9	100.00
3	**9**	**10**	97.51	6	8	2074.00
4	**6**	7	180.35	5	9	2096.00
5	**9**	9	35.29	0	10	7200.00
6	7	7	95.91	9	10	915.00
7	**7**	**9**	131.64	5	9	2426.00
8	4	7	113.77	7	10	2163.00
9	**7**	**10**	54.61	5	7	1886.00
10	**6**	**8**	172.14	6	8	3754.00
Average:	7	8.60	96.89	5.40	9	2267.60

*Average processing times of the BP are those reported by Côté et al. [10]

Table 3 – Experimental results for the Demassey instances.

Instance (Id)	MH solution	Gap (%)	Time MH (s)	BP solution	Time BP (s)
Class 1: 1 day, 50 employees, and 10 activities.					
1808	**3225**	**4.90**	**71.46**	**3270**	**3600.00**
5066	**2440**	**2.43**	**36.69**	**2440**	**935.00**
5135	**2580**	**0.54**	**19.12**	**2580**	**8.00**
5226	**2725**	**0.34**	**20.72**	**2725**	**8.00**
8854	**2740**	**3.46**	**100.63**	**2800**	**3600.00**
Class 2: 1 day, 50 employees, and 10 activities.					
342	**1875**	**1.60**	**8.02**	**1875**	**26.70**
369	**2315**	**0.66**	**12.48**	**2315**	**146.08**
71	**2050**	**0.00**	**4.83**	**2050**	**1.26**
737	**2065**	**2.82**	**9.73**	**2065**	**53.23**
896	1890	3.88	10.20	1875	41.67
Class 3: 2 days, 75 employees, and 12 activities.					
1855	**5960**	**4.27**	**5314.58**	**6265**	**3600.00**
2106	6540	5.99	2632.76	6525	3600.00
2435	**5850**	**3.74**	**1740.24**	**6050**	**3600.00**
4225	**6150**	**5.19**	**1682.04**	**6255**	**3600.00**
9863	**5870**	**3.31**	**868.20**	**5870**	**3600.00**
Class 4: 7 days, 20 employees, and 5 activities.					
1024	**7220**	**0.56**	**81.49**	**7220**	**11.00**
1773	**6345**	**0.01**	**86.14**	**6360**	**9.00**
2732	**7420**	**0.29**	**121.35**	**7420**	**19.00**
4657	**6400**	**1.34**	**78.36**	**6400**	**2003.00**
5553	**7535**	**0.16**	**106.70**	**7600**	**15.00**
Class 5: 7 days, 20 employees, and 5 activities.					
1024	**2940**	**0.00**	**15.73**	**2940**	**0.80**
1773	**2770**	**0.00**	**13.24**	**2770**	**0.76**
2732	**3820**	**0.00**	**9.90**	**3820**	**0.54**
4657	**3210**	**0.00**	**14.22**	**3210**	**0.61**
5553	**3270**	**0.00**	**10.88**	**3270**	**0.59**
Class 6: 7 days, 50 employees, and 7 activities.					
5600	**8440**	**0.64**	**200.03**	**8440**	**59.49**
592	**7345**	**0.27**	**235.63**	**7345**	**33.03**
8597	**7645**	**0.12**	**275.13**	**7645**	**31.58**
9445	**7900**	**0.06**	**169.51**	**7900**	**14.67**
949	**8155**	**0.33**	**275.07**	**8155**	**44.19**
Class 7: 7 days, 50 employees, and 10 activities.					
1007	**14085**	**1.37**	**1157.05**	**14115**	**1321.00**
156	**13420**	**0.79**	**1139.32**	**13420**	**1793.00**
237	**13455**	**0.93**	**875.391**	**13610**	**3600.00**
4369	**13630**	**1.95**	**1703.35**	**13675**	**1536.00**
5216	**14770**	**1.15**	**1171.37**	**14800**	**1824.00**
Class 8: 7 days, 100 employees, and 15 activities.					
530	**15155**	**1.48**	**2430.48**	**15200**	**5818.36**
1024	15435	2.63	2790.49	15420	4602.99
2596	**15765**	**1.76**	**2180.99**	**15855**	**10806.10**
6384	**15235**	**1.47**	**2100.24**	**15250**	**2064.81**
7862	**15880**	**1.98**	**2101.96**	**15940**	**1391.95**

Table 4 – Solutions obtained by the matheuristic for the Lequy instances.

Set of Instances	**Gap**(MH/LR) (%)	**Gap**(BP/LR) (%)	**Gap**(BP/MH) (%)	**Time MH (s)**	**Time BP* (s)**
Class 1	2.34	3.08	-0.70	49.72	1630.20
Class 2	1.79	1.63	0.16	9.06	53.78
Class 3	4.50	6.59	-1.92	2447.56	3600.00
Class 4	0.47	0.69	-0.21	94.81	411.40
Class 5	0.00	0.00	0.00	12.79	0.66
Class 6	0.28	0.28	0.00	231.07	36.59
Class 7	1.24	1.62	-0.37	1209.29	2014.80
Class 8	1.87	2.12	-0.24	2320.83	4936.84
Average:	1.56	2.00	-0.41	796.89	1585.53

*Average processing times of the BP are those reported by Côté et al. [10]

Table 5 – Average solution obtained with the matheuristic for the Lequy instances.

Number of Activities	**Gap**(CGH/CG) (%)	**Gap**(MH/CGH) (%)	**NbS** ($MH \leq CGH$)	**Time MH(s)**	**Time CGH(s)**
1	2.06	-1.84	9	12.01	1.06
2	6.19	-4.32	9	75.64	5.83
3	2.24	-2.11	10	97.51	16.09
4	1.37	-1.15	10	180.35	131.55
5	0.76	-0.74	10	35.29	158.62
6	1.31	-1.17	10	95.91	617.7
7	1.94	-1.72	10	131.64	919.97
8	3.08	-2.32	10	113.77	1163.15
9	0.74	-0.73	10	54.61	827.03
10	2.25	-1.94	10	172.14	1369.66
Average:	2.19	-1.8	9.8	96.89	521.07

Table 6 – Comparison between MH and CGH for the Demassey instances.

Instances	**Gap**(CGH/CG) (%)	**Gap**(MH/CGH) (%)	**NbS** ($MH \leq CGH$)	**Time MH (s)**	**Time CGH (s)**
Class 1	2.99	-0.97	5	49.72	112.78
Class 2	1.83	-0.25	4	9.06	8.81
Class 3	4.11	-1.88	5	2447.56	1558.66
Class 4	0.47	-0.08	5	94.81	15.97
Class 5	0.00	0.00	5	12.79	2.78
Class 6	0.20	-0.07	5	231.07	50.06
Class 7	0.65	-0.34	5	1209.29	1055.64
Class 8	87.12	-26.26	5	2320.83	3018.39
Average:	12.17	-3.73	4.87	796.89	727.89

Table 7 – Comparison between MH and CGH for the Lequy instances.