CIRRELT

Centre interuniversitaire de recherche
sur les réseaux d'entreprise, la logistique et le transport

Interuniversity Research Centre
on Enterprise Networks, Logistics and Transportation

Solving Multi-Activity Personalized Shift Scheduling Problems with a Hybrid Heuristic

Sana Dahmen
Monia Rekik

July 2012

CIRRELT-2012-30

Document de travail également publié par la Faculté des sciences de l'administration de l'Université Laval, sous le numéro FSA-2012-007.

Bureaux de Montréal :
Université de Montréal
C.P. 6128, succ. Centre-ville
Montréal (Québec)
Canada H3C 3J7
Téléphone : 514 343-7575
Télécopie : 514 343-7121

Bureaux de Québec :
Université Laval
2325, de la Terrasse, bureau 2642
Québec (Québec)
Canada G1V 0A6
Téléphone : 418 656-2073
Télécopie : 418 656-2624

www.cirrelt.ca

Solving Multi-Activity Personalized Shift Scheduling Problems with a Hybrid Heuristic

Sana Dahmen, Monia Rekik*

Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation (CIRRELT) and Department of Operations and Decision Systems, 2325, de la Terrasse, Université Laval, Québec, Canada G1V 0A6

Abstract. This paper addresses a multi-activity shift scheduling problem in a continuous and flexible environment including a heterogeneous workforce. Given days-off schedules associated with each employee, our objective is to construct and assign admissible multi-activity shifts to employees on their work days in a way that minimizes under-staffing and over-staffing with a restricted budget on workforce cost. A hybrid heuristic which combines tabu search and a branch-and-bound procedure is proposed to solve the problem. The computational experiments prove that our method provides good schedules in relatively short computing times.

Keywords. Personnel scheduling, multi-activity context, heterogeneous workforce, hybrid heuristic, tabu search, integer programming.

Acknowledgements. This work was supported by the Fonds de recherche du Québec - Nature et technologies (FRQNT) through its new researchers start-up grant. This support is gratefully acknowledged.

---

* Corresponding author: Monia.Rekik@cirrelt.ca

Dépôt légal – Bibliothèque et Archives nationales du Québec
Bibliothèque et Archives Canada, 2012

1. Introduction

For many companies, constructing admissible and optimized schedules for employees is a difficult task. This is mostly due to the multiple and conflicting scheduling constraints defining work environments. Schedules must indeed guarantee a certain number of employees to be present at different periods of the planning horizon to ensure a minimally acceptable service quality. At the same time, over-assigning employees to work periods would yield monetary losses. Schedules must also satisfy collective agreement rules and other constraints related to employees qualifications and preferences. The scheduling task becomes even more complex for work environments operating 24 hours a day, seven days a week and offering a high level of flexibility in schedules definition.

Nowadays, the industrial landscape is moving towards diversification of commodities and services. Consequently, recent studies in personnel scheduling field consider *multi-activity* rather than single-activity operating environments. In this case, schedules are no longer constructed by specifying only the work and rest hours for employees over the planning horizon but also by defining the activities assigned to them during their work periods. When compared to single-activity environments, multi-activity contexts incorporate additional constraints related to employees qualifications, minimum and maximum activity durations within worked shifts, transitions between activities within a shift, etc.

This paper addresses a multi-activity shift scheduling problem in a continuous and flexible environment including a heterogeneous workforce. Given days-off schedules associated with each employee, our objective is to construct and assign admissible multi-activity shifts to employees on their work days in a way that minimizes under-staffing and over-staffing costs. Workforce cost is implicitly handled by adding a constraint on the total budget allocated to it. A heterogeneous workforce means that employees have different qualifications and cannot work all activities. In this case, a schedule is admissible

for an employee if all the activities composing it are those for which the employee is qualified. Considering continuous environments implies that shifts are permitted to overlap on two consecutive days of the planning horizon.

To the best of our knowledge, there are a few papers that addressed the personnel scheduling problem in a multi-activity context. The proposed approaches generally separate multi-activity shift scheduling problems into two sub-problems: shift construction and activity assignment. The first problem determines work start and end times where as the second one assigns activities to the shifts already defined. In this paper, we propose a hybrid heuristic to solve both problems simultaneously. The proposed heuristic combines the well-known tabu search technique and the exact branch-and-bound procedure of CPLEX. We prove through a large set of experiments that our heuristic gives near-optimal solutions in relatively short computing times.

The remainder of this paper is organized as follows. Next section is an overview of the personnel scheduling literature in multi-activity contexts. Section 3 defines the multi-activity shift scheduling problem addressed and proposes a generalized set covering formulation to model it. Section 4 describes in detail the proposed hybrid heuristic. In section 5, we report and discuss the experimental results obtained for a large set of generated instances. Section 6 concludes the paper and opens on possible extensions.

2. Literature review

Personnel scheduling problems are widely studied in the literature (Ernst et al., 2004). Two main approaches can be underlined: the explicit and the implicit approaches. The explicit approach uses a generalized set covering formulation (Dantzig, 1954) in which a binary decision variable is defined for each feasible schedule. This decision variable is equal to 1 if the corresponding schedule is chosen, 0 otherwise. Unlike the explicit approach, the implicit approach does not enumerate all feasible schedules explicitly. It models some forms of flexibility in an implicit way. The most common exam-

ple is the use of forward and backward constraints to model break placement flexibility (Bechtold and Jacobs, 1990). Recently, Rekik et al. (2010) generalize the concept of forward and backward constraints to model minimum and maximum pre- and post-break work stretch duration restrictions.

The explicit approach has the advantage of modelling scheduling problems with complex constraints. However, when the level of flexibility is relatively high, the number of feasible schedules increases considerably making the resulting set covering formulation impossible to solve to optimality in reasonable times. On the opposite, the implicit approach results in smaller models that can be efficiently handled with exact solution approaches. Such an approach can however be used only in restricted contexts having specific properties. Rekik et al. (2004, 2010) define the contexts in which forward and backward constraints can be used in personnel scheduling problems.

To the best of our knowledge, all the published work on multi-activity personnel scheduling considers only explicit models. This is in part due to the complexity of the constraints to be handled in these contexts.

Jin (2009) considers a problem where a number of tasks and activities must be assigned to personalized pre-scheduled shifts in a work environment including a heterogeneous workforce. Tasks must start within specific time windows, have constant durations and are not pre-empted; whereas activities are interruptible, may start at any period and have variable durations. The proposed solution approach involves two stages. The first stage aims at allocating tasks and activities to pre-defined shifts with a first objective of minimizing tasks under-staffing and a second objective of minimizing activities under-staffing. In the second phase, tasks are assigned to shifts as suggested in the first phase and the activities assignment is re-optimized.

Lequy et al. (2010) addressed a multi-activity assignment problem similar to that of Jin (2009), but in which only interruptible activities are considered. Three integer programming models are proposed and solved with mathematical programming based solution approaches: an exact branch-and-bound,

a truncated branch-and-bound heuristic and a column generation heuristic. All these models consider a weighted sum objective function which minimizes under-staffing, over-staffing and transition costs. A rolling horizon procedure is proposed to tackle large-size instances.

Demassey et al. (2005) propose a hybrid constraint-linear programming approach to solve the multi-activity shift scheduling problem. Column generation is used to solve the linear relaxation of the problem. In each iteration, a subset of valid schedules are generated by a constraint satisfaction model and added to the master problem. The authors introduce a new optimization constraint, *the cost-regular global constraint*, within the constraint satisfaction model to generate only the negative reduced cost schedules. Later,Demassey et al. (2006) propose a variable-value ordering heuristic based on cost-regular constraints to solve the constraint satisfaction model. Instead of computing the minimal reduced cost as in Demassey et al. (2005), the authors compute near optimal solutions with low reduced costs.

Quimper and Rousseau (2010) use regular languages and context-free grammars to model a number of scheduling rules. From these languages they derive two large neighbourhood operators and use them in a large neighbourhood search process to solve the the multi-activity shift scheduling problem. Considering the same context, Côté et al. (2007) introduce MIP regular constraints. They use a global *regular* constraint and a network flow problem to model a number of union rules. Then, the MIP version of the regular constraint is used to transform a classical MIP model to a MIP regular model. Experimental results show good computational times. Later, Côté et al. (2011) introduce the MIP grammar constraint for the multi-activity shift scheduling problem. Context-free grammars are used to model union rules defining admissible shifts.

Recently, Côté et al. (2010) propose a branch-and-price algorithm to solve a personalized multi-activity shift scheduling problem. The restricted master problem is a set covering model. For each employee, a pricing sub-problem

is formulated using the MIP grammar constraint, and solved with dynamic programming. The branching rule is adapted from the B&P algorithm of Barnhart et al. (2000).

3. Problem description

3.1. Context and assumptions

We consider a multiple-day planning horizon divided into periods of equal length. Let J denote the set of all days of the planning horizon and I be the set of all its periods. We also denote by I_j the set of periods corresponding to day j.

The scheduling environment we consider includes multiple activities $a \in A$, for which a target demand needs to be satisfied at each period of the planning horizon. Formally, we assume to know for each activity $a \in A$ and each period $i \in I$, the number of employees, denoted $d_{a,i}$, needed to maintain a target activity level. Under-covering and over-covering are permitted but are penalized in the objective function. We denote by $c_{\alpha,a}$, respectively, $c_{\beta,a}$, the unit penalty cost paid for under-covering, respectively, over-covering, a one unit demand of activity a during a period. Notice that the definition of these costs can straightforwardly be generalized to be dependent not only on the activity type but also on the period i in which under-covering or over-covering occurs. In service companies like call centres, banks, or retail stores, under-covering costs can be set according to the loss of service quality (the dissatisfaction of customers due to the waiting time, for example). Over-covering costs can be seen as a penalization related to unproductive time. Generally, companies aim at having schedules that avoid under-covering first and then over-covering, if possible.

Employees are assumed heterogeneous in the sense that they are not all qualified to perform all the activities. In the following, we denote by E the set of all employees and by A_e the set of activities for which employee $e \in E$ is qualified. We also define E_a as the subset of employees that can work

activity a. Each employee is assumed to have a pre-assigned sequence of work and rest days over the planning horizon. We denote by J_e the set of work days of employee e.

The scheduling environment considered includes a high level of flexibility in shifts definition. Different shift types are considered. We denote by T the set of all shift types. A shift type $t \in T$ is defined by a starting time, a duration, a break duration and a break window during which the break must begin. Explicit shifts must be generated from the set T of shift types. Furthermore, when assigning shifts to employees, one must ensure a minimum rest duration D_{min} between two consecutive work days.

Besides, the assignment of activities to admissible shifts must respect minimum and maximum duration restrictions on activities work stretch. The minimum, respectively, maximum, duration restriction of an activity a, denoted respectively L_a^{min} and L_a^{max}, implies that an employee cannot work activity a less than L_a^{min} periods, respectively, more than L_a^{max} periods, consecutively within a shift. The minimum duration restriction is used to limit the number of transitions between activities within a shift avoiding thus a loss of productivity. The maximum duration restriction yields better quality shifts for employees especially when activities require great physical or mental efforts and are stressful.

In the following, we denote by Q_e^j the set of all explicit shifts with assigned activities that are admissible for employee e on day j. A shift q is admissible for e on day j if: (1) its starting and ending times and the break it includes derive from a shift type in T, (2) the activities assigned respect the minimum and maximum duration restrictions, and (3) employee e is qualified for each of the activities composing the shift.

Finally, different objective functions can be considered in practice when designing multi-activity shifts such as workforce costs, under- and over-covering costs, transition costs between activities within a shift, etc. In our case, we consider an objective that minimizes under-covering and over-

covering costs. For labour costs, we assume that there is a maximum budget G_{max} that cannot be exceeded. In the following, we denote by c_q the labour cost of shift $q \in \cup_{e \in E} \cup_{j \in J^e} Q_e^j$. This cost may depend on the shift length, the employee identity, the activities covered by the shift, etc.

3.2. Mathematical formulation

The proposed formulation uses a generalized set covering model. For each employee $e \in E$, each work day $j \in J_e$ of employee e and each explicit shift $q \in Q_e^j$, we define a binary decision variable $x_{e,j,q}$ that equals 1 if shift q is assigned to employee e on day j; and $x_{e,j,q} = 0$; otherwise. We also define for each activity $a \in A$ and each period $i \in I$, two integer decision variables $\alpha_{a,i}$ and $\beta_{a,i}$ that represent the under-covering and over-covering, respectively, of activity a in period i.

The model also uses constant binary parameters $\delta_{q,a,i}$ defined for each shift $q \in \cup_{e \in E} \cup_{j \in J^e} Q_e^j$, each activity $a \in A$ and each period $i \in I$ to indicate whether activity a is covered by shift q in period i (in this case, $\delta_{q,a,i} = 1$), or not ($\delta_{q,a,i} = 0$, in this case). Finally, to model the minimum rest duration restrictions (D_{min}), we denote by s_q and f_q, the starting and finishing times of a shift q.

The mathematical model ($GSCM$) can be written as follows:

$$\min \sum_{a \in A} \sum_{i \in I} (c_{\alpha,a} \alpha_{a,i} + c_{\beta,a} \beta_{a,i}) \tag{1}$$

$$\text{s.t.} \sum_{e \in E_a} \sum_{j \in J_e} \sum_{q \in Q_e^j} \delta_{q,a,i} x_{e,j,q} + \alpha_{a,i} - \beta_{a,i} = d_{a,i} \quad \forall \quad a \in A, i \in I \tag{2}$$

$$\sum_{q \in Q_e^j} x_{e,j,q} = 1 \quad \forall \quad e \in E, j \in J_e \tag{3}$$

$$\sum_{q' \in Q_e^{j+1}} s_{q'} x_{e,j+1,q'} - \sum_{q \in Q_e^j} f_q x_{e,j,q} \geq D_{min} \quad \forall \quad e \in E, (j, j+1) \in J_e^2 \tag{4}$$

$$\sum_{e\in E}\sum_{j\in J_e}\sum_{q\in Q_e^j} c_q x_{e,j,q} \leq G_{max} \tag{5}$$

$$x_{e,j,q} \in \{0,1\} \qquad \forall \quad e \in E, j \in J_e, q \in Q_e^j \tag{6}$$

$$\alpha_{a,i}, \beta_{a,i} \geq 0 \text{ and integers} \qquad \forall \quad a \in A, i \in I \tag{7}$$

Objective function (1) minimizes the total under-covering and over-covering costs. Demand constraints (2) ensure that the total number of qualified employees performing each activity a during each period i of the planning horizon must be equal to the required number (the target demand) subject to some adjustments related to under- and over-covering. Constraint (3) ensures that an admissible shift is assigned to each employee $e \in E$ on every work day $j \in J_e$. Constraint (4) ensures the respect of the minimum rest duration restriction between two shifts assigned to two consecutive work days for each employee schedule. Constraint (5) ensures that the total workforce cost is less than the maximum budget allowed. Finally, constraints (6) and (7) define the nature of the decision variables used in the model.

As already mentioned, considering a model that explicitly enumerates all admissible shifts may be intractable in practice with exact solution methods like those embedded in commercial solvers. The major contribution of this paper is to propose a heuristic that provides good-quality solutions in acceptable computing times for the personnel scheduling problem described above.

4. Hybrid heuristic

This section presents our solution approach. It is based on a hybrid heuristic that combines an adaptation of the well-known tabu search metaheuristic (Glover and McMillan, 1986) with a branch-and-bound procedure (B&B).

4.1. Overview of the hybridization scheme

Tabu Search (TS) is known to be one of the most popular meta-heuristic for approximatively solving large problems. Like other local search methods, at each iteration t, TS moves from a current solution S^t to a next one S^{t+1} by exploring a neighbourhood of S^t. A neighbourhood is generated by a move operator that performs a local perturbation to the current solution S^t. The next solution S^{t+1} is the best neighbour of S^t, even if this neighbour is not improving S^t. This non-improving move permits the search to escape from local optimum but can lead to cycling. TS uses thus the well-known concept of tabu lists to band revisiting previous solutions for a given number of iterations defined by the tabu list size.

In addition to the tabu list, two other memory techniques, operating in a complementary way, are introduced in TS: intensification and diversification. Both techniques are used to store some information collected during the search process. While intensification is used to give the priority to good attributes of elite solutions, diversification is used to discourage considering visited attributes in order to direct the search to unexplored areas of the search space. We refer the reader to the book of Talbi (2009) for more details on tabu search methods.

When designing our heuristic, one of the major challenges we faced is managing the trade off between neighbourhood sizes and neighbourhood exploration complexity. Indeed, considering large neighbourhoods certainly leads to large improvements in solution quality but at the same time requires large computing time to find the best neighbour. To overcome this difficulty, we use the $B\&B$ procedure of the commercial solver CPLEX to efficiently explore large neighbourhoods. The hybridization scheme we propose belongs in fact to a class of hybrid methods known as $Low - LevelRelayHybrid$ (Talbi, 2009). Figure 1 describes the main steps of the proposed hybridization scheme. As one can see, the $B\&B$ algorithm is embedded into the TS at three stages: the neighbourhood exploration, the intensification, and

the diversification stages. Next subsections give explicit details on how the hybridization is done.

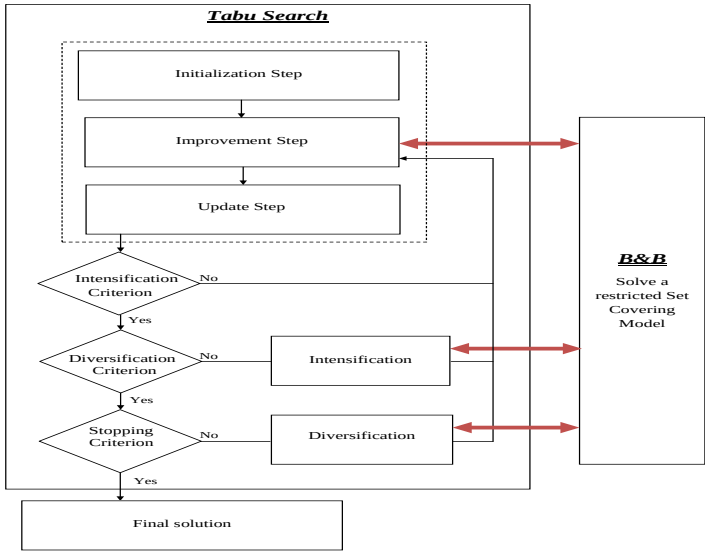

Figure 1: Hybridization scheme

4.2. Initial solution

To start the tabu search process, one needs a feasible initial solution that respect minimum and maximum activity work stretch restrictions, break placement constraint and minimum time separation between shifts on consecutive work days. To this end, we consider the following constructive algorithm:

Algorithm 1 Constructive algorithm

FOR each employee $e \in E$
FOR each working day $j \in J_e$
IF (j is the first day of the planning horizon) or (($j-1$) is a day-off)
THEN:
STEP 1 : Choose randomly a shift type $t \in T$. Go to STEP 2.
STEP 2 : Place randomly the break within the time window of the shift type t. Go to STEP 3.
STEP 3 : Choose randomly a sequence of activities $(a(1), ..., a(m))$. Start the first activity $a(1)$ at the beginning of the shift. Let $p = 1$. Go to STEP 4.
STEP 4 Enumerate all admissible end times for activity $a(p)$
IF (no admissible end time exists) **THEN** Go to STEP 6.
ELSE, Go to STEP 5.
STEP 5 : For each end time obtained for $a(p)$
IF ($p < m$), **THEN** start the next activity $a(p+1)$ at the current end time. $p = p + 1$. Go to STEP 4.
ELSE, go to STEP 6.
STEP 6 : **IF** (a subset of admissible shifts is obtained)
THEN choose randomly an eligible shift $q_{j,e}$,
ELSE
IF (all possible break placements are tested),
THEN choose another shift type and go to STEP 2.
ELSE choose another break placement and go to STEP 3.
ELSE
IF (($j-1$) is a day-on), **THEN**
Determine the set $T(q_{j-1})$ of admissible shift types for the shift $q_{j-1,e}$ assigned to employee e on day $(j-1)$, (a shift type is admissible for $q_{j-1,e}$ if its starting time and the ending time of $q_{j-1,e}$ respect the minimum separation rest duration restriction).
Replace the set of all shift types T by the subset $T(q_{j-1})$ (in STEP 1).
Repeat the steps (1)-(6).

4.3. Neighbourhood definition and exploration

In our case, a solution H corresponds to a set of $|E|$ personalized feasible schedules over the planning horizon, one schedule for each employee $e \in E$. Formally, $H = (h_e)_{e \in E}$ where h_e is defined by a tuple of explicit shifts $(q_j)_{j \in J^e}$.

Since under-covering is generally much less likeable (results in higher costs) than over-covering, our primary objective when designing our heuristic is to ensure that the target demand is satisfied (i.e., under-covering is minimized). Hence, a neighbourhood of a solution $\hat{H}$ is defined by a subset of feasible schedules that minimize the largest under-covering yielded by $\hat{H}$, denoted $\hat{\alpha}$ in the following.

More specifically, we first determine the activity-period couple ($\hat{a}$,$\hat{i}$) for which the under-covering in $\hat{H}$ is maximum. We denote by $\hat{j}$, the day of the planning horizon to which period $\hat{i}$ belongs. Then, a pre-processor is used to generate for each employee e working on day $\hat{j}$ (according to its pre-assigned days-off schedule) a set $Q_e^{\hat{j}}(\hat{H}) \subset Q_e^{\hat{j}}$ of all admissible shifts on day $\hat{j}$ in which activity $\hat{a}$ is worked during $\hat{i}$. Finally, complete schedules for each employee e are enumerated by considering all possible combinations of a shift in $Q_e^{\hat{j}}(\hat{H}) \cup \{\hat{q}_{\hat{j}}\}$ and the shifts already assigned to employee e in solution $\hat{H}$ for the remaining work days (that is,$(\hat{q}_j)_{j \in J^e \setminus \{\hat{j}\}}$). This set of complete schedules is denoted $S_e(\hat{H})$. Notice that including shift $\{\hat{q}_{\hat{j}}\}$ implies that the schedule of employee e in the current solution $\hat{H}$ is kept as an alternative schedule avoiding thus to impose to all employees to work activity $\hat{a}$ on period $\hat{i}$. The neighbourhood of solution $\hat{H}$, denoted $N(\hat{H})$, is then defined by all possible combinations of employees schedules $S_e(\hat{H})$, that is, $N(\hat{H}) = \prod_{e \in E} S_e(\hat{H})$.

To find the best neighbour of $\hat{H}$ in $N(\hat{H})$, we consider a restricted set covering model (as the one described in section (3)) in which we limit the sets of personalized shifts associated with employee e and day j to $Q_e^{\hat{j}}(\hat{H})$ if $j = \hat{j}$ and $\{\hat{q}_j\}$ for all $j \in J^e$. The resulting model is much smaller than the one yielded by all admissible schedules and is expected to be solved to optimality by the classical $B\&B$ procedure of CPLEX. However, in case the resulting restricted model is still difficult to solve to optimality in reasonable times, we reduce its size by randomly considering subsets of $Q_e^{\hat{j}}(\hat{H}), e \in E$.

4.4. short, medium and long-term memories

4.4.1. Tabu list (short-term memory)

To avoid cycling, tabu search employs an adaptive memory that stores recent moves to be banned for a finite number of iterations. In our case, we define a tabu list, denoted CT, as a FIFO queue which stores the last activity-period couples considered in neighbourhoods definition. A dynamic tabu list size is used to improve the robustness of the heuristic. To this end, we define a parameter, denoted ψ, and randomly change its value during the search process. This parameter is used to update the tabu list by removing the tabu status for all couples $(a, i) \in CT$ that are present in CT for more than ψ successive iterations. More details on the value considered for the ψ parameter are given in Section 5.2.

4.4.2. Intensification (medium-term memory)

The intensification is used to exploit promising regions by storing good properties in a medium-term memory (Talbi, 2009). A property is considered good when it often appears in the best neighbour solutions (called also elite solutions). In our case, an intensification is employed when the number of iterations without improvement reaches a pre-fixed number, denoted N_{int}.

To perform intensification, we define a three-dimensional matrix (called the recency matrix) $R = (r_{e,a,i})$, where coefficient $r_{e,a,i}$ gives the maximum number of successive iterations in which employee e performs activity a during period i in the best neighbour solutions. The intensification we propose consists in constructing schedules that keep *active* as much as possible a pre-specified number Δ_{int} of triplets (e, a, i) that have the highest $r_{e,a,i}$ values. A triplet (e, a, i) is active in a solution if employee e is assigned activity a on period i. Notice that Δ_{int} is a parameter of our heuristic that depends on the size of the recency matrix R. In the following, we denote K_{int} the set of Δ_{int} triplets (e, a, i) retained for intensification.

To construct schedules that keep as much as possible triplets in K_{int} active, we partition K_{int} into subsets $K_{int}(e)$, one subset of each employee

e such that $(e, a, i) \in K_{int}$. A pre-processor is then used to generate for each employee e a subset of personalized shifts Q_e^{int} ($\subset \cup_{j \in J^e} Q_e^j$) for which employee e is assigned activity a on period i for the largest possible number of couples (a, i) in $K_{int}(e)$. Notice that some pairs (a, i) in $K_{int}(e)$ may be incompatible because of the minimum and maximum activity duration restrictions. The size of sets Q_e^{int} is restricted to a pre-specified value (a parameter of our heuristic).

Intensification is performed by solving a restricted set covering model (see Section 3) in which only shifts in $Q_e^{int}, e \in E$ are considered. Notice that since $K_{int}(e)$ may be empty for some employees, we add to the subset of shifts $Q_e^{int}, e \in E$ the best solution identified so far to ensure feasibility.

4.4.3. Diversification (long-term memory)

Diversification allows the search to move towards unexplored regions of the feasible space. This technique uses a long-term memory, called frequency memory, which stores the number of times a component appears in the visited solutions. In our case, we apply the so-called *restart diversification* meaning that we consider a new initial solution and restart the search process (Talbi, 2009). We perform diversification when a pre-specified number of iterations, denoted N_{div}, is reached without improvement of the best solution.

To obtain a new diversified solution, we construct a random feasible solution (with Algorithm 1) and make changes on it so that it includes properties that appear the least in the best neighbour solutions identified during the search process. This is done by considering a three-dimensional frequency matrix $F = (f_{e,a,i})$ where coefficient $f_{e,a,i}$ gives the total number of iterations such that employee e performs activity a during period i in the best neighbour solutions. As for intensification, we define a parameter Δ_{div} to determine the number of triplets (e, a, i) that are retained in the frequency matrix F. Then, we determine the set K_{div} of the (e, a, i) triplets that have the Δ_{div} lowest values $f_{e,a,i}$. Observe that K_{div} defines the triplets (e, a, i) that are active the least in the best visited solutions. The diversification we

propose consists in defining schedules that forces as much as possible these triplets to be active.
To this end, we define for each employee $e \in E$, a set $K_{div}(e)$ of pairs (a, i) such that $(e, a, i) \in K_{div}$. A pre-processor is designed to generate for each employee e a subset of personalized shifts Q_e^{div} ($\subset \cup_{j \in J^e} Q_e^j$) in which employee e performs activity a on period i for the largest possible number of couples belonging to $K_{div}(e)$. A restricted set covering model (see Section 3) in which only the subsets of shifts $Q_e^{div}, e \in E$ are considered is then solved. Notice that since $K_{div}(e)$ may be empty for some employees, we add to the subset of shifts $Q_e^{div}, e \in E$ a feasible solution randomly generated by the constructive algorithm (Algorithm 1) to ensure model feasibility. The solution of this model is taken as a new initial solution and tabu search is restarted.

5. Computational experiments

The objective of this section is to evaluate the performance of the proposed heuristic in terms of computing time and solution quality. To this end, 216 problem tests are generated by varying a number of work environment features (shift types, number of activities, number of employees, etc.). Next section describes the problems tests considered in our study.

5.1. Problem tests

Nine instances sets are generated by varying the number of employees $|E|$ (15, 20 and 30) and the set of activities A (a combination of 2, 3 and 4 activities). More precisely, the set of activities is set equal to $\{a_1, a_2\}$, $\{a_1, a_2, a_3\}$ or $\{a_1, a_2, a_3, a_4\}$. Activities a_1, a_2, a_3 and a_4 are ordered according to their stressful level, this stressful level being reflected by the maximum work stretch duration permitted for the activity within a shift. Hence, activity a_1 is the most stressful and imposes a maximum work stretch that is smaller than activity a_2, a_2 is more stressful than a_3, etc. The minimum, respectively, maximum, work stretch duration of activities are set between 1 and 2 hours, and 2 and 4 hours, respectively.

For each instance set $(A, |E|)$, 24 instances are generated by varying demand profiles and shift types yielding 216 instances in total. Six demand profiles, $p = 1, \ldots, 6$ are considered. These demand profiles differ on the degree of demand fluctuation throughout the planning horizon. More precisely, the demand profile $p = 1$ is characterized by a relatively low standard deviation implying that the demand variation across time periods tend to be very close to the mean demand. On the opposite, for demand profile $p = 6$, demand values are spread out over a large range with respect to the mean value.

On another hand, 120 shift types are generated by considering starting times at each hour of the day and 5 different durations (6, 7, 8, 9 and 10 hours). Break windows are defined in the middle of the shift work interval and the break duration is taken equal to one or two hours depending on the shift type length. These shift types are generated and stored in a pool. For each instance set $(A, |E|)$ and each demand profile p, four instances are generated by randomly selecting 36, 54, 96 or 120 shift types within the pool of shift types. Notice, that shift types are selected in a way to ensure that all periods of the day are covered.

For all generated instances, we consider a continuous 7-day planning horizon divided into periods of 60-minutes each ($|I| = 7 \times 24 = 168$ periods). We also assume that all employees are qualified for all activities. Finally, under-covering and over-covering costs are taken equal to 1 for all activities.

Notice that ,to evaluate the performance of the proposed heuristic, the 216 instances we generate are designed so that the corresponding optimal solution is known. More precisely, data are fixed in a way that ensures that a solution with no under-covering and no over-covering exists for each instance (that is the optimal solution yields a null objective value). This is done by conveniently tuning some parameters such as pre-assigned days-on sequences, demand profiles and activities work stretch restrictions.

5.2. Parameters settings

As already explained, the hybrid heuristic we propose considers a number of parameters related to short, medium and long-term memories that need to be fixed. After performing a number of tests with different values of these parameters, we find that the best trade off between solution time and solution quality is on average obtained with the following settings.

The tabu list size ψ is first set to ψ^0 and this value is randomly modified every 5 iterations within the interval $[\psi^0(1-\nu), \psi^0(1+\nu)]$ with $\nu \in \{0, 1\}$. The value of ψ^0 is set to $6\times|A|$ (recall that $|A|$ denotes the number of activities in the instance set). An intensification is performed when the number of successive iterations without improvement exceeds 3 (i.e., $N_{int} = 3$). A diversification is performed when the number of successive iterations without improvement exceeds 6 (i.e., $N_{div} = 6$). The algorithm is stopped when two diversifications are performed. The best solution identified is the solution output by the heuristic.

Table 1 reports the average, minimum and maximum size of the tabu list recorded for each instance set over the 24 instances it includes (Column "Tabu list size"). It also reports the average, minimum and maximum number of intensifications (Column "Int_{Best}"), respectively, diversifications (Column "Div_{Best}") required by the heuristic to locate the best solution.

	Av. Tabu list size			*# Int.$_{Best}$*			*# Div.$_{Best}$*		
$(\|A\|, \|E\|)$	Av.	Min	Max	Av.	Min	Max	Av.	Min	Max
(2. 15)	7.44	6.23	8.97	3.29	0	13	0.79	0	2
(2. 20)	7.73	6.21	8.76	3.88	0	9	0.83	0	2
(2. 30)	7.91	6.38	9.08	4.46	1	8	1.04	0	2
(3. 15)	9.15	7.00	11.32	4.33	0	11	0.83	0	2
(3. 20)	9.13	6.49	12.13	3.79	1	10	0.58	0	2
(3. 30)	10.12	7.35	13.03	4.13	1	9	0.71	0	2
(4. 15)	10.42	7.69	12.40	4.79	1	11	1.13	0	2
(4. 20)	10.68	6.46	15.10	5.75	1	14	0.96	0	2
(4. 30)	11.39	8.08	15.86	5.17	1	9	0.96	0	2

Table 1: Short, medium and long-term memories' parameters

Table 1 shows that the intensification phase is necessary to improve the quality of the heuristic solution. In fact, the average number of intensifi-

cations needed to reach the best solution varies from 3.29 (instance set (2, 15)) to 5.75 (instance set (4, 20)). Even if for some instances the best solution is reached without intensification, 14 intensifications are needed in some cases (instance set $(4, 20)$). One can also observe that the number of intensifications needed to identify the best solution increases with the number of employees and the number of activities defining the scheduling environment.

Regarding the diversification phase, the number of diversifications required to identify the best solution is lower than 1 for almost all instances sets. Recall that the heuristic stopping criteria is defined with respect to a maximum number of diversifications which is set to 2 for all instances. More precisely, over the 216 instances considered in this study, 39% locate the best solution with no diversification, 34% required one diversification and 26% needed the two diversifications.

5.3. Solution quality

This section presents the solutions obtained with the proposed algorithm with the parameters settings described in Section 5.2. In order to evaluate the quality of the solution obtained, we consider a number of performance indicators (P.I.) defined as follows.

5.3.1. Performance indicators

Two P.I., denoted respectively φ^{α} and φ^{β}, are defined to measure the total under-covering, respectively, over-covering, of all activities over all periods with respect to the total demand. That is,

$$\varphi^{\alpha} = \frac{\sum_{a \in A} \sum_{i \in I} \alpha_{a,i}}{\sum_{a \in A} \sum_{i \in I} d_{a,i}} \tag{8}$$

$$\varphi^{\beta} = \frac{\sum_{a \in A} \sum_{i \in I} \beta_{a,i}}{\sum_{a \in A} \sum_{i \in I} d_{a,i}} \tag{9}$$

where, $\alpha_{a,i}$, respectively, $\beta_{a,i}$, denotes the number of employees lacking, respectively in excess, for activity a on period i yielded by the heuristic solution.

Obviously, the lower the value of these P.I. is, the better the solution output by the heuristic meets the total target demand.

Two other P.I. are defined at the activity level to measure the average under-covering, denoted $\overline{\Gamma}^{under}$, respectively, the average over-covering, denoted $\overline{\Gamma}^{over}$, of the demand per activity and per period yielded by the heuristic solution. These P.I. are computed as follows:

$$\overline{\Gamma}^{under} = \frac{1}{|A||I|} \sum_{a \in A} \sum_{i \in I} \alpha_{a,i} \tag{10}$$

$$\overline{\Gamma}^{over} = \frac{1}{|A||I|} \sum_{a \in A} \sum_{i \in I} \beta_{a,i} \tag{11}$$

Unlike φ^{α} and φ^{β} which give an idea on the total uncovered and over-covered demand for all activities and all periods, $\overline{\Gamma}^{under}$ and $\overline{\Gamma}^{over}$ give details on under-covering and over-covering on both the activity and the period levels.

5.3.2. Results

Table 2 reports the min, max and average values of the four P.I. defined in Section 5.3.1 for the nine generated instances sets. Recall that each instance set $(|A|, |E|)$ includes 24 instances. Each line in Table 2 displays thus the minimum, maximum and average value over the 24 instances of the corresponding instance set.

Table 2 shows that our heuristic provides good quality solutions for all instances set. In fact, the total under-covering φ^{α} ranges between 0% and 5.30% but does not exceed 0.78% on average (for all the 216 instances). The local under-covering P.I. ($\overline{\Gamma}^{under}$) confirms the good performance of our heuristic since the average under-covering per activity per period is very small (between 0.0% and 0.02%) reaching 0.10% for the worst instance.

In terms of over-covering, the solutions output by the heuristic remain also satisfactory ranging between 0% for the best case and 6.50% for the

	$\varphi^{\alpha}(\%)$			$\varphi^{\beta}(\%)$			$\overline{\Gamma}^{under}$			$\overline{\Gamma}^{over}$		
$(\|A\|, \|E\|)$	Av.	Min	Max	Av.	Min	Max	Av.	Min	Max	Av.	Min	Max
(2, 15)	1.23	0	5.04	1.16	0.00	3.29	0.02	0	0.07	0.02	0	0.04
(2, 20)	0.92	0	3.15	0.58	0.00	1.57	0.02	0	0.06	0.01	0	0.03
(2, 30)	1.59	0	3.63	1.30	0.00	3.78	0.04	0	0.10	0.04	0	0.10
(3, 15)	0.31	0	2.76	0.40	0.00	2.55	0.00	0	0.03	0.00	0	0.02
(3, 20)	0.60	0	2.53	0.66	0.00	2.48	0.01	0	0.03	0.01	0	0.03
(3, 30)	0.20	0	1.79	0.24	0.00	1.45	0.00	0	0.03	0.01	0	0.03
(4, 15)	1.06	0	2.85	1.08	0.00	3.11	0.01	0	0.02	0.01	0	0.02
(4, 20)	0.96	0	5.30	1.19	0.16	6.50	0.01	0	0.05	0.01	0	0.06
(4, 30)	0.15	0	0.54	0.22	0.00	0.97	0.00	0	0.01	0.00	0	0.01

Table 2: Quality of the solutions obtained with the proposed hybrid heuristicp

worst case for the global over-covering P.I. (φ^{β}) and 0% and 0.07% for the local over-covering P.I. ($\overline{\Gamma}^{over}$).

Table 2 also reflects the robustness of our hybrid heuristic since the quality of the solutions it yields is insensitive to the input data and the instances size.

5.3.3. Heuristic versus optimal solutions

The performance of a heuristic method is generally evaluated by measuring the minimum, the average and the maximum deviation of the heuristic solution with respect to the optimal solution if it is known, to an estimator of the optimal solution or to a pre-specified bound. In our case, our problem tests are generated in a way that the optimal solution is known and yields null under-covering and null over-covering of all activities and all periods. Hence, the deviation of our heuristic solution to the optimal solution can be straightforwardly derived from global P.I. φ^{α} and φ^{β}.

Figure 2 gives the percentage of instances for which our heuristic yields a deviation (φ^{α}) to the optimal under-covering and a deviation (φ^{β}) to the optimal over-covering that is less than or equal to a certain number Π (Π ranges between 0 and 6% with an incrementation step of $0,1\%$).

Figure 2 shows that the proposed heuristic identifies a solution that is 1.5% far from the optimal solution in terms of under-covering for 80% of the instances. For 33.8% of the instances (73 over the 216 considered), the

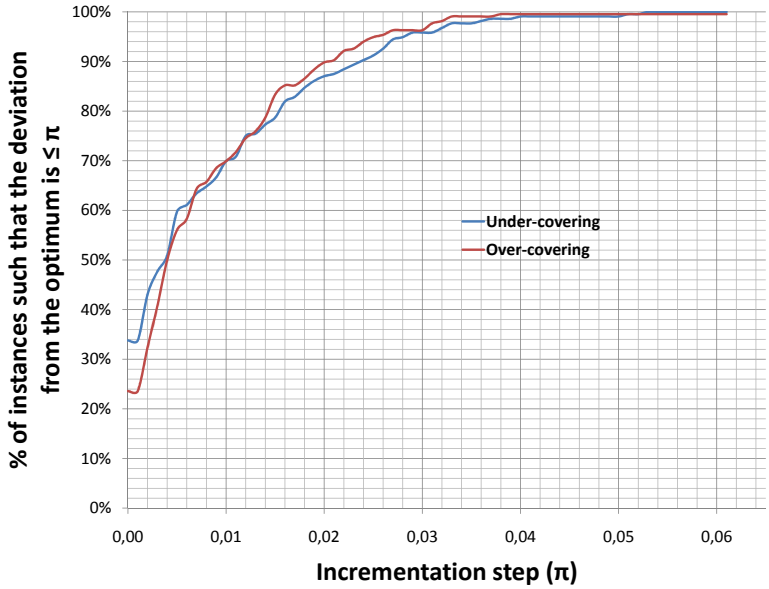

Figure 2: The percentage of instances for which φ^{α}, respectively, φ^{β} is $\leqslant \Pi$

heuristic identifies the optimal solution in terms of under-covering. The maximum deviation with respect to the optimal under-covering solution does not exceed 5.3%.

When considering over-covering, the optimal solution is identified for 23.61% of the instances (51 over the 216 considered). The under-covering yielded by our heuristic solution is below 1.4% for 80% of the instances and does not exceed 6.5% for the worst instance.

As a conclusion, our heuristic computes good quality solutions. Although the resulting schedules are not optimal for all instances, the deviation to the optimum remains relatively small for both under-covering and over-covering. Furthermore, in practice, it is difficult to determine with certainty the number of employees required for each activity and each period. There is generally a margin of error in demand forecasting of about 2% making our heuristic

solutions almost optimal in practice.

5.4. Solution time

This section reports the computational performance of the proposed heuristic in terms of solution times. Table 3 displays two time values: (1) the total computational time in minutes, CPU_{Tot}, required by the heuristic (the time required to reach the stopping criteria of two diversifications) and (2) the computational time in minutes needed to reach the best solution (CPU_{Best}).

	$CPU + Tot(min)$			$CPU_{Best}(min)$		
$(\|A\|, \|E\|)$	Av.	Min	Max	Av.	Min	Max
(2, 15)	5.59	0.29	12.61	3.37	0.29	12.61
(2, 20)	8.94	1.70	16.75	5.57	0.34	16.04
(2, 30)	12.36	3.04	33.65	8.22	0.68	20.94
(3, 15)	13.46	0.83	31.42	10.71	0.83	30.98
(3, 20)	20.87	2.54	46.95	10.75	1.76	31.76
(3, 30)	28.68	3.73	68.41	18.13	3.73	46.98
(4, 15)	28.16	2.98	62.65	19.85	2.97	55.21
(4, 20)	40.21	12.71	85.72	25.40	3.42	45.79
(4, 30)	38.93	9.34	71.96	28.00	5.66	70.68

Table 3: Computational Time of the hybrid heuristic

Table 3 shows that the total time required by the proposed heuristic to perform two diversifications varies between 5.59 minutes on average for instances set $(2, 15)$ to 38.93 minutes on average for instances set $(4, 30)$. This average total time is about 21.91 minutes when considering the 216 instances. The best-case instance requires 0.29 minutes (almost 18 seconds) in total whereas the worst-case instance needs more than 85.72 minutes. In all cases, one can assert that this total computing time remains reasonable for handling a multiple-day (one week in our case) shift scheduling problem given the quality of the resulting solutions.

Furthermore, the results displayed under the $"CPU"_{Best}$ column show that identifying the best solution does not necessarily require all this computing time (no need to perform two diversifications). The average time required to locate such best solutions ranges between 0.29 and 70.68 minutes with an average of 14.44 minutes for all the 216 instances. In sum, the algorithm

could be stopped after about 40% of the total CPU and would yield a good quality solution. For example, for the worst case instance in terms of total CPU (85.72 minutes), the best solution was identified in less than 32.15 minutes.

As a conclusion, our experimental study proves that the proposed heuristic shows good performances in terms of solution quality and solution time. We believe that it can be implemented in real-life contexts and results in appreciable schedules for both employees and managers.

It is worth mentioning that we tested the relevance of the proposed neighbourhood and compare the performance of the heuristic with a neighbourhood as defined in Section 4 and a neighbourhood where couples $(a, i))$ are randomly selected. The results we obtained prove that considering the couple (a, i) that yields the maximum under-covering gives better results for both solution quality and solution times.

6. Conclusion and futures extensions

This paper considers a continuous personnel scheduling problem in multi-activity work environments with heterogeneous workforce. To the best of our knowledge, there are a few papers that addressed the personnel scheduling problem in a multi-activity context. The proposed approaches generally separate the problem into shift construction and activity assignment. The major contribution of this paper is to propose an efficient hybrid heuristic to solve both problems simultaneously on relatively long planning horizons (a week). It is assumed that days-off schedules associated with each employee are known and our objective is to construct and assign admissible multi-activity shifts to employees on their work days in a way that minimizes under-staffing and over-staffing with a budget restriction on total workforce cost.

The proposed heuristic combines the well-known tabu search technique and the exact branch-and-bound procedure of CPLEX. Branch and bound is

used at three stages: neighbourhood exploration, intensification and diversification. Neighbourhoods are defined based on the couples (activity, period) for which under-covering is maximized. We prove through a large set of experiments that our heuristic gives near-optimal solutions in relatively short computing times.

Many research directions can be explored in the future. As already mentioned, the few papers that addressed multi-activity scheduling problems consider generalized set covering formulations. However, implicit modelling was proven to be efficient in solving single-activity scheduling problems having a particular structure (Bechtold and Jacobs, 1990; Aykin, 2000; Rekik et al., 2004). It results in smaller models especially when a high degree of flexibility has to be handled. A future trend would be to consider implicit approaches to solve multi-activity shift scheduling problems. We are currently working on that.

Acknowledgments

This work was supported by the Fonds de recherche du Québec Nature et Technologie (FQRNT) through its new researchers start-up grant. This support is gratefully acknowledged.

References

Aykin, T., 2000. A Comparative Evaluation of Modeling Approaches to The Labor Shift Scheduling Problem. European Journal of Operational Research 125, 381–397.

Barnhart, C., Hane, C., Vance, P., 2000. Using branch-and-price-and-cut to solve origin-destination integer multicommodity flow problems. Operations Research, 318–326.

Bechtold, S. E., Jacobs, L. W., 1990. Implicit Modeling of Flexible Break Assignments in Optimal Shift Scheduling. Management Science 36 (11), 1339–1351.

Côté, M., Gendron, B., Quimper, C., Rousseau, L., 2011. Formal languages for integer programming modeling of shift scheduling problems. Constraints 16 (1), 54–76.

Côté, M., Gendron, B., Rousseau, L., 2007. Modeling the regular constraint with integer programming. Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems, 29–43.

Côté, M., Gendron, B., Rousseau, L., Centre interuniversitaire de recherche sur les réseaux d'entreprise, l. l. e. l. t., 2010. Grammar-Based Column Generation for Personalized Multi-Activity Shift Scheduling. CIRRELT.

Dantzig, G. B., 1954. Traffic Delays at Toll Booths. Journal of the Operations Research Society of America 2 (3), 339–341.

Demassey, S., Pesant, G., Rousseau, L., 2005. Constraint programming based column generation for employee timetabling. Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems, 830–833.

Demassey, S., Pesant, G., Rousseau, L., 2006. A cost-regular based hybrid column generation approach. Constraints 11 (4), 315–333.

Ernst, A. T., Jiang, H., Krishnamoorthy, M., Sier, D., 2004. Staff Scheduling and Rostering: A Review of Applications, Methods and Models. European Journal of Operational Research 153, 3–27.

Glover, F., McMillan, C., 1986. The General Employee Scheduling Problem: An Integration of MS and AI. Computers & Operations Research 13 (5), 563–573.

Jin, J., 2009. Pré-affectation des Taches aux Employés Effectuant des Taches Non-interruptibles et des Activités Interruptibles. Master's thesis, école Polytechnique, Montréal, CANADA.

Lequy, Q., Bouchard, M., Desaulniers, G., Soumis, F., Tachefine, B., 2010. Assigning Multiple Activities to Work Shifts. Journal of Scheduling, 1–13.

Quimper, C., Rousseau, L., 2010. A large neighbourhood search approach to the multi-activity shift scheduling problem. Journal of Heuristics 16 (3), 373–392.

Rekik, M., Cordeau, J.-F., F., S., 2004. Using Benders Decomposition to Implicitly Model Tour Scheduling. Annals of Operations Research 128, 111–133.

Rekik, M., Cordeau, J.-F., F., S., 2010. Implicit Shift Scheduling with Multiple Breaks and Work Stretch Duration Restrictions. Journal of scheduling 13 (1), 49–75.

Talbi, E.-G., 2009. Metaheuristics From Design to Implementation. John Wiley & Sons, Inc.