RAIRO-Oper. Res. 42 (2008) 229–258
DOI: 10.1051/ro:2008006

RAIRO Operations Research
www.rairo-ro.org

SOLUTION APPROACHES TO LARGE SHIFT SCHEDULING PROBLEMS

Monia Rekik[1], Jean-François Cordeau[2] and François Soumis[1]

Abstract. This paper considers large shift scheduling problems with different shift start times and lengths, fractionable breaks and work stretch duration restrictions. Two solution approaches are proposed to solve the problems over a multiple-day planning horizon. The first approach is based on a local branching strategy and the second one is based on a temporal decomposition of the problem. Local branching is very efficient in finding good feasible solutions when compared to a classical branch-and-bound procedure. However, the decomposition approach has the advantage of yielding feasible solutions in short computing times, even for difficult instances.

Keywords. Shift scheduling, flexibility, fractionable breaks, work stretch restrictions, forward and backward constraints, local branching, heuristic.

Mathematics Subject Classification. 90C10, 90C11, 90C29.

1. Introduction

Shift scheduling problems have received a lot of attention in the last decades. Constructing shifts typically consists of specifying the daily work start and finish times as well as the number, the duration and the position of the breaks. It is well known that incorporating flexibility alternatives into operating environments may

Received May 31, 2007. Accepted November 28, 2007.

[1] École Polytechnique de Montréal and GERAD, C.P. 6079, succ. Centre-Ville Montréal, H3C 3A7, Canada

[2] HEC Montréal and GERAD, 3000, chemin de la Côte-Sainte-Catherine, Montréal H3T 2A7, Canada; jean-françois.cordeau@hec.ca

Article published by EDP Sciences

considerably complicate the problem. However, considering some forms flexibility has been proved to also yield important savings in labor costs.

In this paper, we consider a continuous shift scheduling problem that includes a high degree of flexibility in terms of shift start time, shift length, break length and break placement. Two types of breaks are considered: the standard (indivisible) breaks and the *fractionable* breaks, recently introduced by Rekik *et al.* [18]. Standard breaks have a fixed length and are permitted to start within a time window, generally referred to as a break window. Unlike a standard break, a *fractionable break* is not restricted to be taken as a whole. It can be separated into several subbreaks under some conditions. In other words, one needs only to specify the total duration of the break to be attributed to a shift: the number, the length and the position of the associated subbreaks are optimally determined by the scheduling model. Some restrictions may however be imposed on the way a fractionable break is divided. They may concern, for example, the number of subbreaks, subbreak lengths, etc. The scheduling environment considered also incorporates work stretch duration restrictions. These constraints fix minimal and maximal periods of continuous work before and after each break. They ensure a convenient mix of work and rest periods within a shift. Our objective is to minimize the total workforce size over a multiple-day planning horizon given the flexible environment described above.

Two solution approaches are proposed. The first one is based on the local branching method initially introduced by Fischetti and Lodi [12] to solve general integer programming problems. We particularly propose some local branching cuts that are specific to shift scheduling problems. The second approach is based on a temporal decomposition of the planning horizon into time windows. Each restricted shift scheduling problem associated to a time window is solved to optimality by a classical branch-and-bound procedure. Some parts of the solution obtained for a time window are re-optimized in order to consider overlapping shifts arising in continuous operating environments. Both approaches are compared to the branch-and-bound procedure of CPLEX 9.0. The computational experiments show that the proposed local branching yields better solutions than the branch-and-bound procedure of CPLEX 9.0 and the windowing approach for relatively small problems. For larger problems, the windowing approach is superior and yields good feasible solutions in relatively short times.

The remainder of the paper is organized as follows. In the next section we give an overview of the shift scheduling literature by focusing on solution methods. In Section 3, we define the operating environments and expose the implicit formulations used to model the problem as proposed by Rekik *et al.* [18]. Local branching and time windowing approaches are then described in Section 4. In Section 5, we illustrate model characteristics for a set of real-life instances from an air-traffic control agency as well as a set of generated instances. The computational performance of the proposed models is finally assessed and compared for each of the three solution approaches.

2. Literature review

Although scheduling flexibility adds a lot of complexity to the shift scheduling problem, it has been shown to help reduce labor costs. For example, Bechtold and Jacobs [6] evaluated the reduction in workforce size obtained by increasing the number of possible shift starts, by varying shift lengths, and by considering break windows. Brusco and Jacobs [8] demonstrated through a large experimental study that just a small subset of shift start times is enough to provide a minimal workforce size. However, the quality of the solution may rapidly deteriorate when the adequate subset is not considered. Moreover, finding the suitable subset is a difficult and time consuming task. Aykin [2] studied the impact of break placement flexibility in a multiple break context. He concluded that using large break windows for all breaks may considerably decrease the number of employees required. Recently, Topaloglua and Ozkarahan [21] compared the workforce size needed for a scheduling environment using only eight-hour shifts to that required for an environment allowing three different shift lengths (eight-, ten- and twelve-hour shifts). An average reduction of 20% was obtained for the environment offering greater shift length variability. Furthermore, the large set of instances considered by Topaloglua and Ozkarahan [21] confirm the positive impact of using large break windows on the total workforce size. Finally, Rekik *et al.* [18] recently introduced the concept of fractionable breaks. They reported that the use of such "dividable" breaks may yield up to 8% savings in terms of the number of employees needed when compared to a context where only standard breaks (*i.e.*, that cannot be divided) are used.

Two main approaches have been proposed to handle highly flexible and thus complex shift scheduling problems: *explicit* approaches and *implicit* approaches.

The explicit approach uses the classical set covering formulation initially introduced by Dantzig [11] and tries to handle the large size of the resulting models by developing efficient heuristics. In fact, a typical set covering model defines a separate integer variable for each explicit work shift (*i.e.*, daily work start and finish times with assigned rest periods). Clearly, a scheduling problem with a high degree of flexibility yields a large number of shift alternatives and thus large set covering models that are very difficult to solve with exact methods. For example, Henderson and Berry [13] have modeled a shift scheduling problem including up to 15 000 explicit work shifts. The problem is solved heuristically in two phases. A subset of explicit shifts is first selected with the so-called *maxdif* heuristic. The resulting shift scheduling problem with a restricted subset is then solved using some heuristic procedures based on the solution of the LP relaxation of the problem. The *maxdif* procedure adds shifts iteratively to the subset of selected shifts, one shift at a time. At each iteration, all remaining (*i.e.*, not yet selected) shifts are compared to all shifts already selected. The shift that maximizes the number of periods that are not covered by the selected shifts is chosen. Three heuristics are proposed for the second phase to solve the restricted shift scheduling problems. Each of these heuristics converts the fractional LP solution into an integer one and

then tries to improve it by local search. The local search consists in permuting employee shifts. In another context, Thompson [20] considered the explicit approach for a shift scheduling problem where employees are not continuously available. A simulated annealing procedure is used to solve the problem. Recently, Hochbaum and Levin [14] studied the complexity of three shift scheduling problems modeled as generalized set covering formulations with bounded variables. These problems are classified according to the column structure in the constraint matrix: column of consecutive ones, column of cyclical ones and column of k consecutive blocs of ones. Hochbaum and Levin [14] proved that the k-bloc shift scheduling problem is NP-hard for all $k \geq 2$. They proposed a k-approximation algorithm that reduces the k-bloc problem to a one-bloc problem, which is known to be polynomially solvable. In the proposed algorithm, each column composed of k_i blocs of ones in the original problem is conveniently split into k_i columns, each including a unique bloc of consecutive ones. A feasible solution is then obtained by fixing the value of each original variable (column) to the maximum value of the associated bloc variables.

The *implicit* approach focuses more on the modeling aspect and tries to reduce the problem size by implicitly representing some forms of flexibility. Exact branch-and-bound or branch-and-cut procedures are generally used to solve these implicit models. Moondra [16] was the first to implicitly represent shift length. His model considers shift start time and shift finish time variables, and uses a set of constraints to impose limits on the shift duration. Bechtold and Jacobs [5] implicitly model break placement flexibility for a discontinuous shift scheduling problem in a single break context. They define separate shift and break variables and ensure a correct match between them by using the so-called *forward* and *backward* constraints. This set of constraints guarantees that an eligible break can be assigned to each shift in its associated break window, without specifying its actual start time. Complete shift schedules with appropriate break assignments are constructed *a posteriori* using a break allocation algorithm. The allocation procedure arranges shifts in a nondecreasing order with respect to the latest period in which a break may occur and assigns shifts to the earliest available breaks. In fact, using forward and backward constraints assumes the absence of *extraordinary overlap*. Bechtold and Jacobs [5] define extraordinary overlap (E.O.) as the situation where there exists two shifts such that the break window for one shift begins strictly earlier and ends strictly later than the break window for the other shift. A branch-and-bound procedure is then used to solve the problem. Bechtold and Jacobs [5] show, through computational experiments, that their model is superior to the traditional set covering formulation with respect to computing time and memory. Recently, Addou and Soumis [1] proved that extraordinary overlap may be handled by an extension of the formulation of Bechtold and Jacobs [5] which incorporates a small set of additional constraints. One constraint is in fact added for each observed E.O.

Thompson [19] later combined the work of Moondra [16] to implicitly represent shifts and the work of Bechtold and Jacobs [5] to implicitly represent break

placement. Each shift is assumed to receive at most one break with some restrictions on pre- and post-break work stretch durations. A branch-and-bound procedure followed by a post-processor are used to generate actual schedules. The post-processor constructs explicit shifts and assigns breaks to them by using a first-in-first-out (FIFO) procedure. Thompson [19] reported that the FIFO matching can alleviate the E.O. problem.

Aykin [2] proposed another implicit approach to model break placement flexibility for a shift scheduling problem with multiple breaks and multiple break windows. He defines a break variable for each shift and each possible start time within the associated break window. Contrarily to Bechtold and Jacobs [5], Aykin [2] does not impose any restriction on the definition of break windows. Moreover, Aykin [2] considered a continuous operating environment where shifts are permitted to overlap from one day to the next. Aykin [2] use the demand constraints to compute upper bounds for shift variables. These upper bounds are included in the proposed formulation and are proved to be helpful in reducing the solution time in the branch-and-bound process. In a subsequent paper, Aykin [3] proposed a more elaborate branch-and-cut algorithm that uses objective value cuts and dynamically computed upper bounds on shift variables. The linear relaxation of the implicit formulation proposed in [2] is first solved to obtain a lower bound. A rounding heuristic is then applied to the fractional solution to get a feasible schedule and thus an upper bound. Cuts are added and iteratively updated while using the rounding heuristic and a limited branch-and-bound search to find an optimal schedule. Good results are obtained for large instances. An optimal solution was found in several cases; in the remaining ones, the "best non-optimal" solution (*i.e.*, superior by one to the lower bound) was identified.

Recently, Rekik *et al.* [18] introduced the concept of fractionable breaks and focused on the modeling aspect of the problem. Two implicit models were proposed. The first model extends the approach of Aykin [2] and the second one extends that of Bechtold and Jacobs [5]. Adapted forward and backward constraints are used in both formulations to model the work stretch duration restrictions. We also proposed a reformulation of general forward and backward constraints and showed that it considerably reduces model density and solution times. The computational results reported in [18] were for a single-day planning horizon and use the CPLEX solver with most of its parameters set to default values. In this paper, we focus on solution methods for longer planning horizons including one or several weeks. The solution approaches we present are in fact inspired by classical decomposition methods and local search concepts recently proposed for difficult MIP problems.

In this field, Fischetti and Lodi [12] introduced a local branching strategy that integrates both local search and metaheuristics with MIP. It constructs MIP subproblems that consider only a small neighborhood of a current incumbent solution. Neighborhoods are obtained through the introduction in the MIP of the so-called *local branching* constraints (the local branching strategy is explained in more details in Sect. 4.1). More recently, Danna *et al.* [10] proposed two approaches for exploring interesting, domain-independent neighborhoods for the MIP problems: the so-called *Relaxation Induced Neighborhood Search (RINS)* and *guided dives*. In

RINS, a neighborhood is defined by using the information contained in the continuous relaxation of the MIP model. More specifically, variables that have the same values in the incumbent and in the linear relaxation of the current node of the branch-and-bound tree are fixed. Neighborhood exploration is then formulated as a restricted MIP model and solved recursively. The guided dives approach is a simple modification of the default MIP tree traversal strategy. The choice of the child node to be explored first in the branch-and-bound process is made with regard to the value of the branching variable in the current incumbent solution. That is, for the MIP including binary variables, for example, the first child node to be explored is the one in which the binary branching variable is fixed to the value that it takes in the incumbent. Danna *et al.* [10] reported that their approaches outperform a slightly different implementation they propose for the local branching strategy of Fischetti and Lodi [12] and default CPLEX 8.1., in terms of the quality of the solution obtained within a specified time limit, of the ability to improve a given integer solution, and of the time required to diversify the search in order to find a new solution.

3. Mathematical models

This section extends to a continuous w-day planning horizon the two formulations we previously proposed [18] for the shift scheduling problem with fractionable breaks, standard breaks, break windows and work stretch duration restrictions. In both formulations generalized forward and backward constraints are used to model either work stretch duration restrictions (for the first formulation) or both work stretch duration and break window constraints (for the second formulation). The concept of forward and backward constraints was in fact introduced by Bechtold and Jacobs [5] to model break placement flexibility. It was recently generalized by Çezik and Günlük [9] and Rekik *et al.* [17] who proved that the feasibility of some transportation problems can be ensured by the use of adequate forward and backward constraints. In this context, Rekik *et al.* [18] proved that satisfying work stretch duration restrictions also corresponds to ensuring the feasibility of some transportation problems, and that these transportation problems possess the structure required to apply the generalized forward and backward constraints.

We first introduce the parameters and the sets needed to describe generalized forward and backward constraints.

3.1. Generalized forward and backward constraints

Consider a transportation problem, $T(N_1, N_2)$, represented by a bipartite network $G = (N_1 \cup N_2, A)$ where N_1 and N_2 are the sets of supply and demand nodes, respectively. The supply of each node $i \in N_1$ is denoted by O_i and the demand of each node $j \in N_2$ is denoted by D_j. Assume that the structure of $T(N_1, N_2)$ is such that a total order relation $\prec$ can be defined on set N_2. Assume also that each supply node $i \in N_1$ is connected to a set of consecutive demand nodes $j \in N_2$,

denoted by P_i. Finally, there exist no supply nodes i_1 and i_2 in N_1 such that $\min(P_{i_1}) \prec \min(P_{i_2})$ and $\max(P_{i_2}) \prec \max(P_{i_1})$.

To describe forward and backward constraints, we define the following sets:

$$\begin{aligned} N_2^s &= \cup_{i\in N_1}\{\min(P_i)\} \\ N_2^e &= \cup_{i\in N_1}\{\max(P_i)\} \\ N_2^B(j) &= \{j' \in N_2 | j \preceq j'\} \qquad \forall j \in N_2^s \\ N_2^F(j) &= \{j' \in N_2 | j' \preceq j\} \qquad \forall j \in N_2^e \\ N_1^B(j) &= \{i \in N_1 | P_i \subseteq N_2^B(j)\} \qquad \forall j \in N_2^s \\ N_1^F(j) &= \{i \in N_1 | P_i \subseteq N_2^F(j)\} \qquad \forall j \in N_2^e. \end{aligned}$$

Finally, let $n_2^s = \min(N_2)$ and $n_2^e = \max(N_2)$. Forward and backward constraints are then given, respectively, by:

$$\sum_{j'\in N_2^F(j)} D_{j'} \geq \sum_{i\in N_1^F(j)} O_i \qquad \forall\, j \in N_2^e \setminus \{n_2^e\}$$

and

$$\sum_{j'\in N_2^B(j)} D_{j'} \geq \sum_{i\in N_1^B(j)} O_i \qquad \forall\, j \in N_2^s \setminus \{n_2^s\}.$$

Under the assumptions made at the beginning of this section, the feasibility of $T(N_1, N_2)$ is ensured by the use of forward and backward constraints and an equality constraint between the total supply and total demand (see [9]; or [17]):

$$\sum_{i\in N_1} O_i - \sum_{j\in N_2} D_j = 0.$$

Figure 1 illustrates an example of a transportation problem having the appropriate structure. In this example, $N_1 = \{1,2,3,4\}$, $N_2 = \{5,6,7,8,9,10,11\}$, $P_1 = \{5,6,7\}$, $P_2 = \{7,8,9,10\}$, $P_3 = \{8,9,10\}$, $P_4 = \{10,11,12\}$, $N_2^e = \{7,10,12\}$, and $N_2^s = \{5,7,8,10\}$. Relying on the results stated above, the feasibility of this transportation problem is ensured by three sets of constraints. The first set includes an equality constraint between the total supply and the total demand. The second set includes two forward constraints (for $j = 7$ and $j = 10$). The third set incorporates three backward constraints (for $j = 7$, $j = 8$ and $j = 10$). We give hereafter the forward constraint corresponding to node $j = 10$ (Inequality (1)) and the backward constraint corresponding to node $j = 7$ (Inequality (2)).

$$D_5 + D_6 + D_7 + D_8 + D_9 + D_{10} \geq O_1 + O_2 \tag{1}$$

$$D_7 + D_8 + D_9 + D_{10} + D_{11} + D_{12} \geq O_2 + O_3 + O_4. \tag{2}$$

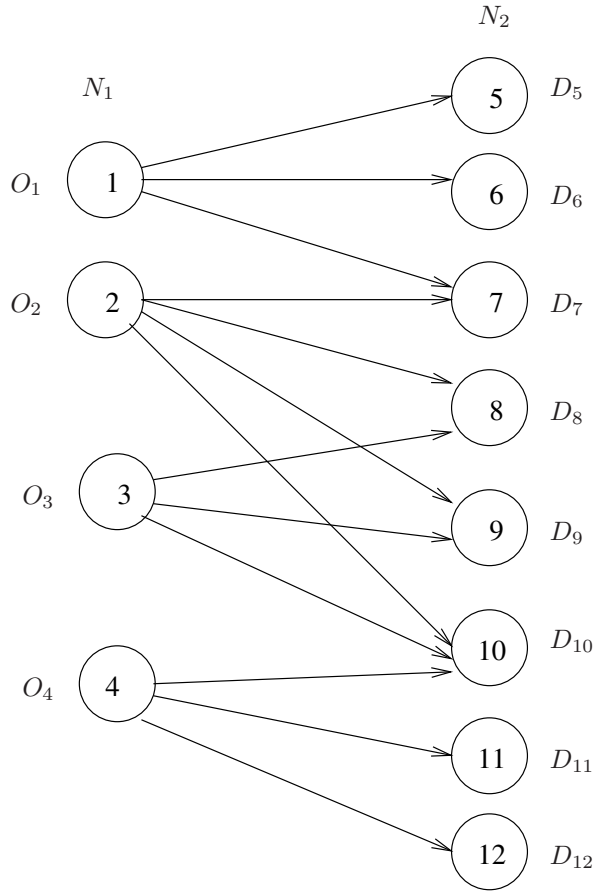

FIGURE 1. Example of a transportation problem with the required structure.

In fact, Rekik *et al.* [18] proposed a reformulation of forward and backward constraints that uses additional slack variables without increasing the total number of constraints. They prove that such a reformulation considerably reduces the density of the constraint matrix and solution times. A non-negative slack variable Z_{j^e} (respectively, Z_{j^s}) is defined for each forward (respectively, backward) constraint corresponding to node $j^e \in N_2^e \setminus \{n_2^e\}$ (respectively, $j^s \in N_2^s \setminus \{n_2^s\}$). Given the total order relation $\prec$ defined on set N_2, forward and backward constraints can be

rewritten as follows:

$$\sum_{j' \in N_2^F(j_1^e)} D_{j'} - \sum_{i \in N_1^F(j_1^e)} O_i = Z_{j_1^e}$$

$$\sum_{j' \in N_2^F(j_{(l+1)}^e) \setminus N_2^F(j_l^e)} D_{j'} - \sum_{i \in N_1^F(j_{(l+1)}^e) \setminus N_1^F(j_l^e)} O_i + Z_{j_l^e} = Z_{j_{(l+1)}^e} \quad \forall j_l^e \in N_2^e \setminus \{n_2^e\}$$

$$\sum_{j' \in N_2^B(j_1^s)} D_{j'} - \sum_{i \in N_2^B(j_1^s)} O_i = Z_{j_1^s}$$

$$\sum_{j' \in N_2^B(j_{(l+1)}^s) \setminus N_2^B(j_l^s)} D_{j'} - \sum_{i \in N_1^B(j_{(l+1)}^s) \setminus N_1^B(j_l^s)} O_i + Z_{j_l^s} = Z_{j_{(l+1)}^s} \quad \forall j_l^s \in N_2^s \setminus \{n_2^s\},$$

where $j_1^e = \min(N_2^e)$, $j_1^s = \max(N_2^s)$, $j_{(l+1)}^e$ is the successor of j_l^e in N_2^e, and $j_{(l+1)}^s$ is the predecessor of j_l^s in N_2^s. Hereafter, we illustrate the reformulation of forward constraints through the example of Figure 1 (we refer the reader to [18], for the proof in more general contexts). Recall that two forward constraints are needed for the example of Figure 1. These constraints are given by:

$$D_5 + D_6 + D_7 \geq O_1 \tag{3}$$

$$D_5 + D_6 + D_7 + D_8 + D_9 + D_{10} \geq O_1 + O_2. \tag{4}$$

Let $Z_1^e \geq 0$ be the slack variable associated with forward constraint (3) and $Z_2^e \geq 0$ be the one associated with forward constraint (4). These constraints can thus be rewritten as:

$$D_5 + D_6 + D_7 - O_1 = Z_1^e \tag{5}$$

$$D_5 + D_6 + D_7 + D_8 + D_9 + D_{10} - O_1 - O_2 = Z_2^e. \tag{6}$$

Constraint (6) can thus be reformulated as:

$$D_8 + D_9 + D_{10} - O_2 + Z_1^e = Z_2^e.$$

To simplify the presentation, we illustrate the models with a classical formulation of forward and backward constraints. However, in the computational experiments, we use the models with reformulated forward and backward constraints.

3.2. Problem definition

We consider a continuous planning horizon of w days and 24 h work days. As is common in shift scheduling models, the planning horizon is divided into *periods* of equal lengths. The set of all periods is denoted by I. The number of employees required in each period $i \in I$ is assumed to be known in advance and is denoted by d_i.

Rekik *et al.* [18] circumvented the complexity resulting from the fractionable break concept by using an enumerative pre-processor that generates the so-called

break profiles. A break profile represents a sequence of break lengths that must be attributed to a given shift. Thus, if a shift must receive a fractionable break, all feasible sequences of subbreak lengths associated with a fractionable break are enumerated. If the shift must also receive some standard breaks (*i.e.*, breaks that cannot be divided), all admissible permutations of subbreak and standard breaks are enumerated. The pre-processor also generates the so-called *window profile* for each shift. A window profile includes the break windows associated with the breaks of the break profile of the given shift. These break windows are determined by considering the pre- and post-break work stretch duration restrictions and the pre-existing break windows of standard breaks. The models proposed by Rekik *et al.* [18] handle the case where work stretch duration parameters (*i.e.*, minimum and maximum pre- and post-break work stretch durations) depend on the time interval (the so-called *day-part*) of the day. The minimum and maximum number of consecutive work periods associated with a day-part dp are denoted respectively by α_{dp} and μ_{dp}. The set of all day-parts is denoted by TDP (generally, there are three day-parts: day, evening and night). The work stretch parameters that apply to a given shift are those associated with the day-part covering the most periods in the shift.

A set J of all possible shifts is then generated by the pre-processor. A shift $j \in J$ is defined for each combination of start time, length, break profile and window profile. The break profile, respectively the window profile, associated with shift j is denoted by BP_j, respectively WP_j. The total number of breaks that must be attributed to a shift j is denoted by q_j (q_j is thus the number of elements in BP_j or WP_j). Parameter $BP_j(p)$ is used to denote the length of the break that must be attributed to shift j in the pth position whereas $WP_j(p)$ denotes the break window associated with it. The set of all break profiles, *i.e.*, $\cup_{j\in J} BP_j$, is denoted by TBP. Finally, the day-part that maximizes the number of periods covered by shift j is denoted by dp_j.

To determine the set K of all breaks, the break profiles and the window profiles of all shifts $j \in J$ are considered. A break k is defined for each break profile $bp \in TBP$, each position p of a break in bp, each possible start time t (with respect to the window profiles associated with break profile bp), and each admissible day-part dp of the operating day. A day-part dp is admissible for break k if break k can be matched with a shift j associated with dp (*i.e.*, such that $dp_j = dp$).

Extending the models proposed by Rekik *et al.* [18] from a one-work-day planning horizon to a w-work-day horizon consists in enlarging the set of periods I, the set of shifts J, and the set of breaks K (K is actually derived from J).

3.3. Implicit formulation ($P1$)

Rekik *et al.* [18] defined a network flow variable X_{jk} for each shift j and each break k that can be matched with shift j. This variable represents the number of breaks k attributed to shift j. A break k is eligible for shift j if its break profile corresponds to the same break profile BP_j of shift j, if it has a position p_k and a start time t_k such that $t_k \in WP_j(p_k)$, and, finally, if it is associated with the same

day-part as shift j. In the following, the set of breaks k that can be matched with shift j in the position p, $p \in Q_j$, $Q_j = \{1, \ldots, q_j\}$, is denoted by $K_{j(p)}$. Similarly, J_k will denote the set of shifts j that can receive a break k in position p_k.

The minimum and maximum number of periods that can be worked consecutively within shift $j \in J$ are denoted respectively by α_j and μ_j. The work stretch duration restrictions imply that two consecutive breaks attributed to shift j must be separated by at least α_j and at most μ_j periods. Rekik *et al.* [18] showed that these constraints can be modeled with adapted forward and backward constraints derived from a particular transportation problem $T(K_{j(p)}, K_{j(p+1)})$ (as defined in Sect. 3.1) where $p \in Q_j^-$, $Q_j^- = \{1, \ldots, q_j - 1\}$ and X_{jk} represents the supply/demand associated with nodes in $K_{j(p)}$ and $K_{j(p+1)}$. An arc exists between a node $k^1 \in K_{j(p)}$ and a node $k^2 \in K_{j(p+1)}$ if break k^2 starts at least α_j periods and at most μ_j periods after break k^1 ends.

To describe the model, the following parameters are introduced:

δ_{ij}: $= 1$ if shift $j \in J$ covers period $i \in I$; 0 otherwise;
ρ_{ik}: $= 1$ if break k covers period $i \in I$; 0 otherwise.

Model $(P1)$ is given by :

$$(P1) \quad \text{minimize} \quad \sum_{j \in J} S_j \tag{7}$$

subject to

$$\sum_{j \in J} \delta_{ij} S_j - \sum_{k \in K} \rho_{ik} \sum_{j \in J_k} X_{jk} \geq d_i \forall i \in I \tag{8}$$

$$\sum_{k \in K_{j(p)}} X_{jk} - S_j = 0 \forall j \in J, p \in Q_j \tag{9}$$

$$\sum_{k' \in K^F_{j(p+1)}(k)} X_{jk'} - \sum_{k' \in K^F_{j(p)}(k)} X_{jk'} \geq 0 \; \forall j \in J, p \in Q_j^-, k \in K^e_{j(p+1)} \setminus \{k^e_{j(p+1)}\} \tag{10}$$

$$\sum_{k' \in K^B_{j(p+1)}(k)} X_{jk'} - \sum_{k' \in K^B_{j(p)}(k)} X_{jk'} \geq 0 \forall j \in J, p \in Q_j^-, k \in K^s_{j(p+1)} \setminus \{k^s_{j(p+1)}\} \tag{11}$$

$$\sum_{k \in K_{j(p+1)}} X_{jk} - \sum_{k \in K_{j(p)}} X_{jk} = 0 \forall j \in J, p \in Q_j^- \tag{12}$$

$$S_j \geq 0 \text{ and integer } \forall \, j \in J \tag{13}$$

$$X_{jk} \geq 0 \text{ and integer } \forall \, j \in J, k \in \cup_{p \in Q_j} K_{j(p)}. \tag{14}$$

The objective function (7) minimizes the total number of employees over the planning horizon. Demand constraints (8) ensure that the number of employees working in a certain period, *i.e.*, those who are present and not on break,

is at least equal to the demand for that period. Break window constraints (9) ensure that each shift j receives q_j breaks, one in each position $p \in Q_j$, such that the break in position p has a length compatible with the break profile of shift j and starts within $WP_j(p)$. Work stretch duration constraints (10)–(12) are forward, backward and equality constraints corresponding to transportation problems $T(K_{j(p)}, K_{j(p+1)}), j \in J, p = 1, \ldots, q_j - 1$. It can be verified that equality constraints (12) are redundant in the presence of constraints (9) and can thus be removed.

3.4. Implicit formulation ($P2$)

Model ($P2$) uses shift variables S_j as defined for model ($P1$). A break variable B_k is defined for each break $k \in K$ and represents the number of employees receiving break k.

Rekik *et al.* [18] proved that each shift will receive its breaks within the associated break windows if the feasibility of some transportation problems is ensured. In fact, they defined a transportation problems $T\left(J_{(bp,dp)}, K_{(bp,dp,p)}\right)$ for each break profile $bp \in TBP$, each day-part $dp \in TDP$, and each possible position p of a break in bp, $p \in N_{bp}$, where $N_{bp} = \{1, \ldots, n_{bp}\}$ and n_{bp} denotes the total number of breaks in bp. The set of supply nodes $J_{(bp,dp)}$ contains all shifts j associated with break profile bp and day-part dp, *i.e.*, $J_{(bp,dp)} = \{j \in J | BP_j = bp \text{ and } dp_j = dp\}$. The supply associated with node j is S_j. The set of demand nodes $K_{(bp,dp,p)}$ contains all breaks k associated with day-part dp and occupying the pth position in break profile bp, *i.e.*, $K_{(bp,dp,p)} = \{k \in K | BP_k = bp \text{ and } dp_k = dp \text{ and } p_k = p\}$. The demand for node k is B_k. An arc (j, k) exists between a shift $j \in J_{(bp,dp)}$ and a break $k \in K_{(bp,dp,p)}$ if break k starts within the break window associated with the pth position of the window profile WP_j of shift j. Breaks in $K_{(bp,dp,p)}$ are sorted in ascending order with respect to their start time, which defines a total order relation on set $K_{(bp,dp,p)}$. The authors assume that there is no extraordinary overlap between breaks in $K_{(bp,dp,p)}$. Hence, for a given break profile bp, a given day-part dp and a given position p of a break in bp, the transportation problem $T\left(J_{(bp,dp)}, K_{(bp,dp,p)}\right)$ can be replaced by a set of forward, backward and equality constraints as described in Section 3.1.

To model pre- and post-break work stretch duration restrictions, Rekik *et al.* [18] consider a transportation problem $T\left(K_{(bp,dp,p)}, K_{(bp,dp,p+1)}\right)$ for each break profile $bp \in TBP$, each day-part $dp \in TDP$ and each position p of a break in bp, $p \in Nbp^-$ where $N_{bp}^- = \{1, \ldots, n_{bp} - 1\}$. The supply/demand of a node k is B_k. An arc (k^1, k^2) exists between a break $k^1 \in K_{(bp,dp,p)}$ and a break $k^2 \in K_{(bp,dp,p+1)}$ if break k^2 starts at least α_{dp} and at most μ_{dp} periods after break k^1 ends. Breaks in $K_{(bp,dp,p+1)}$ are sorted in ascending order with respect to their start time. It can be verified that the condition of no extraordinary overlap, as defined in Section 3.1, is satisfied. Thus, problems $T\left(K_{(bp,dp,p)}, K_{(bp,dp,p+1)}\right)$, $bp \in TBP, dp \in TDP, p \in N_{bp}^-$, can be replaced by adapted forward, backward and equality constraints.

Model $(P2)$ is thus written as follows:

$$(P2) \quad \text{minimize} \quad \sum_{j \in J} S_j \tag{15}$$

subject to

$$\sum_{j \in J} \delta_{ij} S_j - \sum_{k \in K} \rho_{ik} B_k \geq d_i \quad \forall\, i \in I \tag{16}$$

$$\sum_{k' \in K^F_{(bp,dp,p)}(k)} B_{k'} - \sum_{j \in J^F_{(bp,dp)}(k)} S_j \geq 0 \quad \forall\, bp \in TBP, dp \in TDP, p \in N_{bp} \quad \forall\, k \in K^e_{(bp,dp,p)} \setminus \{k^e_{(bp,dp,p)}\} \tag{17}$$

$$\sum_{k' \in K^B_{(bp,dp,p)}(k)} B_{k'} - \sum_{j \in J^B_{(bp,dp)}(k)} S_j \geq 0 \quad \forall\, bp \in TBP, dp \in TDP, p \in N_{bp} \quad \forall\, k \in K^s_{(bp,dp,p)} \setminus \{k^s_{(bp,dp,p)}\} \tag{18}$$

$$\sum_{k \in K_{(bp,dp,p)}} B_k - \sum_{j \in J_{(bp,dp)}} S_j = 0 \quad \forall\, dp \in TDP, bp \in TBP, p \in N_{bp} \tag{19}$$

$$\sum_{k' \in K^F_{(bp,dp,p+1)}(k)} B_{k'} - \sum_{k' \in K^F_{(bp,dp,p)}(k)} B_{k'} \geq 0 \quad \forall\, bp \in TBP, dp \in TDP, p \in N^-_{bp} \quad \forall\, k \in K^e_{(bp,dp,p+1)} \setminus \{k^e_{(bp,dp,p+1)}\} \tag{20}$$

$$\sum_{k' \in K^B_{(bp,dp,p+1)}(k)} B_{k'} - \sum_{k' \in K^B_{(bp,dp,p)}(k)} B_{k'} \geq 0 \quad \forall\, bp \in TBP, dp \in TDP, p \in N^-_{bp} \quad \forall\, k \in K^s_{(bp,dp,p+1)} \setminus \{k^s_{(bp,dp,p+1)}\} \tag{21}$$

$$\sum_{k \in K(bp,dp,p+1)} B_k - \sum_{k \in K(bp,dp,p)} B_k = 0 \quad \forall\, bp \in TBP, dp \in TDP, p \in N^-_{bp} \tag{22}$$

$$S_j \geq 0 \text{ and integer} \quad \forall\, j \in J \tag{23}$$

$$B_k \geq 0 \text{ and integer} \quad \forall\, k \in K. \tag{24}$$

Model $(P2)$ considers the same objective function (15) and the same demand constraints (16) as model $(P1)$. Constraints (17), (18) are forward, backward and equality constraints corresponding to transportation problems $T(J_{(bp,dp)},$

$K_{(bp,dp,p)}), bp \in TBP, dp \in TDP, p = 1, \ldots, n_{bp}$. They ensure that each shift receives the associated number of breaks and each attributed break is eligible with respect to break length (*i.e.*, its position in the break profile) and break start time (with respect to the associated break window). Pre- and post-break work stretch durations are modeled by forward, backward and equality constraints (20)–(22) corresponding to transportation problems $T(K_{(bp,dp,p)}, K_{(bp,dp,p+1)}), bp \in TBP$, $dp \in TDP, p = 1, \ldots, n_{bp} - 1$. One can verify that equality constraints (22) are redundant in the presence of equality constraints (19) and can thus be removed.

The shift scheduling problem addressed in this paper considers a multiple-break context and restrictions on the minimum and maximum work stretch duration before and after each break. The solution obtained by formulation $(P1)$ yields explicit shifts with specified start and finish times and break assignments. However, formulation $(P2)$ does not use such an explicit assignment variable to link shifts with associated breaks. In fact, it incorporates separate shift and break variables $S_j, j \in J$ and $B_k, k \in K$. The correct match between shifts and breaks is ensured by the forward and backward constraints. The optimal implicit solution, (S^*, B^*) determined by $(P2)$ guarantees that an explicit schedule with break assignments can always be obtained from it. In fact, the optimal non-null breaks $B^*_k, k \in K^*$, determined by $(P2)$ can be assigned to the optimal non-null shifts $S^*_j, j \in J^*$, in different ways depending on the scheduling problem. For a single break context, this can be done using a simple allocation procedure as that described in Bechtold and Jacobs [5], for example. In a multiple break context, although such a procedure assigns a break to a shift within the associated break window, it does not guarantee that these breaks are conveniently placed one with respect to another when work stretch duration restrictions must be satisfied. To handle the complexity of work stretch duration constraints, the assignment of optimal breaks to optimal shifts is done by solving an integer programming model denoted hereafter by $(P3)$. Model $(P3)$ incorporates integer variables X_{jk}, similar to those of model $(P1)$, for each shift j of a non-null optimal variable $S^*_j, j \in J^*$, and each break k of a non-null optimal variable $B^*_k, k \in K^*$, that is admissible for j. X_{jk} represents the number of breaks $k \in K^*$ attributed to shift $j \in J^*$. The objective function of model $(P3)$ has a constant value since the only purpose of considering this model is to determine a feasible solution to the break assignment problem. Model $(P3)$ incorporates the same set of constraints (9)–(14) of model $(P1)$ but only for shifts $j \in J^*$ and breaks in $k \in K^*$; they will also consider only the break and shifts variables of these sets. Furthermore, shift variables S_j of constraints (9) will have a constant value equal to S^*_j. Finally, the following set of constraints is added to $(P1)$ to ensure that the number of assignments of a break k does not exceed the total number of available breaks k:

$$\sum_{j \in J_k \cup J^*} X_{jk} = B^*_k \quad \forall k \in K^*.$$

Model $(P3)$ is thus given by:

$$(P3) \quad \text{minimize} \qquad 0$$

subject to

$$\sum_{k \in K_{j(p)} \cap K^*} X_{jk} = S^*_j \quad \forall j \in J^*, p \in Q_j$$

$$\sum_{j \in J_k \cap J^*} X_{jk} = B^*_k \quad \forall k \in K^*$$

$$\sum_{k' \in K^F_{j(p+1)}(k) \cap K^*} X_{jk'} - \sum_{k' \in K^F_{j(p)}(k) \cap K^*} X_{jk'} \geq 0 \quad \forall j \in J^*, p \in Q^-_j,$$
$$k \in (K^e_{j(p+1)} \setminus \{k^e_{j(p+1)}\}) \cap K^*$$

$$\sum_{k' \in K^B_{j(p+1)}(k) \cap K^*} X_{jk'} - \sum_{k' \in K^B_{j(p)}(k) \cap K^*} X_{jk'} \geq 0 \quad \forall j \in J^*, p \in Q^-_j,$$
$$k \in (K^s_{j(p+1)} \setminus \{k^s_{j(p+1)}\}) \cap K^*$$

$$\sum_{k \in K_{j(p+1)} \cap K^*} X_{jk} - \sum_{k \in K_{j(p)} \cap K^*} X_{jk} = 0 \quad \forall j \in J, p \in Q^-_j$$

$$X_{jk} \geq 0 \quad \text{and integer} \quad \forall\, j \in J^*,$$
$$k \in (\cup_{p \in Q_j} K_{j(p)}) \cap K^*.$$

4. Solution approaches

This section presents two approaches for solving complex shift scheduling problems over long planning horizons. The first approach is based on the local branching strategy initially introduced by Fischetti and Lodi [12] to solve difficult MIP problems. The local branching strategy presented here uses local branching cuts that are adapted for shift scheduling problems. The second approach uses a temporal decomposition of the problem into different sub-problems, one for each time window of the planning horizon. This latter approach is referred to as *time windowing*.

4.1. Adapted local branching strategy

The local branching strategy aims to improve the heuristic behavior of Mixed Integer Programming solvers. It provides good feasible solutions at early stages of the solution process and updates incumbent solutions frequently, thus accelerating the solution process.

Local branching strategies use two branching criteria. The first branching criterion is external and tends to direct the search into a promising search sub-space. The second branching criterion corresponds to the branching performed within the MIP solver. It is used to explore the sub-space resulting from the first branching. As noticed by Fischetti and Lodi [12], local branching is inspired by local neighborhood search methods (see, *e.g.*, [15]) in the sense that the search sub-space

represents in fact a neighborhood of a given feasible solution to the problem. This neighborhood is defined through the so-called *local branching constraints*.

More specifically, consider a MIP problem and let $\mathcal{B}$ denote the set of binary variable indices. For a given solution $\bar{x}$ of the problem, let $\bar{\mathcal{B}}$ denote the binary support of $\bar{x}$, *i.e.*, $\bar{\mathcal{B}} = \{j \in \mathcal{B} : \bar{x}_j = 1\}$. For a given positive integer parameter k, Fischetti and Lodi [12] define the $k - OPT$ *neighborhood* $\mathcal{N}(\bar{x}, k)$ of $\bar{x}$ as the set of feasible solutions of the problem satisfying the following local branching constraints:

$$\begin{aligned} \Delta(x, \bar{x}) &\leqslant k, \\ \text{where } \Delta(x, \bar{x}) &= \sum_{j \in \bar{\mathcal{B}}} (1 - x_j) + \sum_{j \in \mathcal{B} \setminus \bar{\mathcal{B}}} x_j. \end{aligned}$$

As observed by Fischetti and Lodi [12], for a given neighbor x of $\bar{x}$, the *distance function* $\Delta(x, \bar{x})$ represents the number of binary variables x_j flipping their value, with respect to $\bar{x}$, either from 1 to 0 or from 0 to 1.

A basic local branching scheme is illustrated in Figure 2 (we use a figure similar to that presented by Fischetti and Lodi [12]). In this figure, a triangle marked by the word "solver" corresponds to the branching subtrees to be explored with the standard branching criterion of the exact MIP solver at hand. As one can see from Figure 2, the local branching method starts with an initial incumbent solution $\bar{x}^1$ at the root node. This solution can be obtained heuristically or by the MIP solver. Two branches are then created to partition the initial feasible region, $\mathcal{N}$, into two disjoint sub-regions. The left branch corresponds to the local branching constraints and defines a neighborhood of $\bar{x}^1$, $\mathcal{N}(\bar{x}^1, k)$. The right branch defines all feasible solutions except those belonging to $\mathcal{N}(\bar{x}^1, k)$. The left branch node is explored with the MIP solver until an optimal solution, $\bar{x}^2$, is obtained (with respect to the neighborhood). The same scheme is then re-applied on the smallest feasible region $\mathcal{N} \setminus \mathcal{N}(\bar{x}^1, k)$ with the new incumbent solution $\bar{x}^2$. Obviously, updating the incumbent solution assumes that the optimal solution found in a given neighborhood improves the current incumbent one. In the case where no improved solution is obtained (such as for node 6), the right branch node (corresponding in the figure to sub-tree $\mathcal{N} \setminus (\mathcal{N}(\bar{x}^1, k) \cup \mathcal{N}(\bar{x}^2, k) \cup \mathcal{N}(\bar{x}^3, k))$) is totally explored with the MIP solver.

Fischetti and Lodi [12] reported that the local branching method they propose may be used either as an exact solution method or as a heuristic method that yields good feasible solutions that are not necessarily optimal. With the basic scheme described above, local branching explores all the branching tree while giving the priority to some promising search sub-spaces. It acts then as an exact solution method. For the heuristic aspect, Fischetti and Lodi [12] impose a time limit for the left branch computation to handle the cases where exploring all the left subtree is very time consuming. If the time limit is reached with no improved solution (with respect to the incumbent), a backtrack to the father node is done and a new left branch node is created by reducing the size of the neighborhood (*i.e.*, reducing the right hand side of the local branching cut). If an improved solution is obtained,

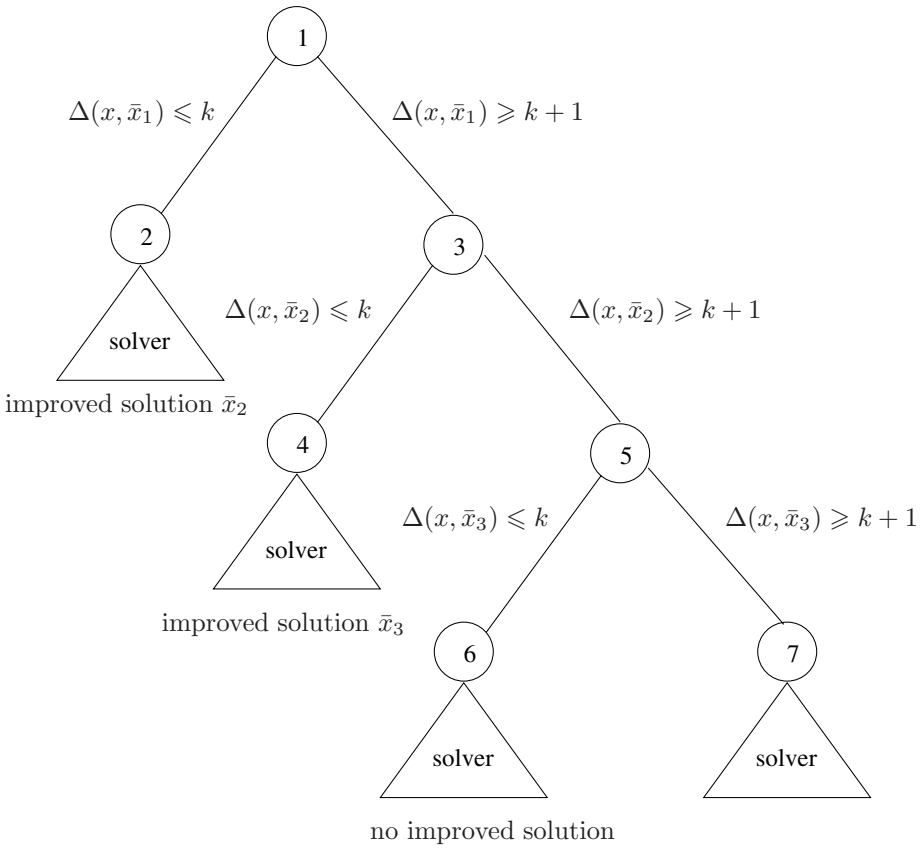

FIGURE 2. Basic local branching strategy.

it becomes the new incumbent, a backtrack to the father node is done, and a new left branch node associated with the new incumbent solution is created.

Furthermore, Fischetti and Lodi [12] used some diversification mechanisms to handle the case where the exploration of the left branch node terminates with no improved solution. Two types of diversifications are applied: the so-called *soft* and *strong* diversifications. A soft diversification consists in enlarging the current neighborhood. A strong diversification not only considers larger neighborhoods but also accepts solutions that are worse than the incumbent. This worse solution is then used as the new reference solution and the method is re-applied in an attempt to improve it and eventually improve the incumbent.

It is worth mentioning that local branching constraints were initially defined with respect to binary variables only. This was done intentionally since, according to [12], the difficulty of MIP problems comes essentially from these variables. The authors however suggest, as an extension of their paper, some local branching

constraints including general variables that can be used for MIPs that incorporate only general integer variables.

The shift scheduling problem discussed in this paper is in fact a MIP with only general integer variables. Inspired by the suggestions of Fischetti and Lodi [12], we propose hereafter some local branching constraints that are specific to shift scheduling problems.

Let $(\bar{S}, \bar{X})$ (respectively $(\bar{S}, \bar{B})$), denote a feasible *reference* solution of a shift scheduling problem modeled with model $(P1)$ (respectively model $(P2)$). The distance function of the local branching cut we propose here is defined with respect only to the shift variables of the reference solution. Moreover, only non zero variables are considered. Let $\bar{J}(\bar{S})$ denotes the set of these variable indices. Thus $\bar{J}(\bar{S}) = \{j \in J : \bar{S}_j \neq 0\}$.

To describe the local branching constraints, one has to introduce two additional integer variables, S_j^+ and S_j^- for each $j \in \bar{J}(\bar{S})$. The value of these variables measures the distance between the value of a current shift variable S_j and the corresponding reference value $\bar{S}_j$. Thus:

$$S_j = \bar{S}_j + S_j^+ - S_j^- \qquad \forall j \in \bar{J}(\bar{S}).$$

Thus, for a given reference solution $(\bar{S}, \bar{X})$ of $(P1)$ or $(\bar{S}, \bar{B})$ of $(P2)$, the local branching cut is defined as:

$$\sum_{j \in \bar{J}(\bar{S})} (S_j^+ + S_j^-) \leqslant k,$$

where k is a fixed integer parameter.

Except for the definition of the local branching constraints, the local branching strategy scheme used is the same as that presented by Fischetti and Lodi [12] with the enhanced heuristic solution scheme described above.

4.2. Time windowing approach

As its name suggests, the time windowing approach is based on a temporal decomposition of the shift scheduling problem into several sub-problems, one for each time window of the planning horizon. Each shift scheduling sub-problem is then solved to optimality over the associated time window. The final solution for the overall planning horizon is obtained by combining all the partial solutions of the sub-problems. Thus, time windowing appears as a pure heuristic approach since there is no guarantee that the final solution is optimal.

The definition of the time windows has a major impact on the performance of the time windowing approach in terms of solution quality and computing times. In fact, time windows must be sufficiently small to yield shift scheduling subproblems that are easy to solve within reasonable times, but also still large enough to yield a global solution that is not too far from the optimal one. Furthermore, one can decompose the planning horizon into disjoint time windows so that each

period belongs to only one time window. However such a decomposition does not consider overlapping shifts that arise in continuous scheduling problems. In fact, it assumes a discontinuous formulation of the problem, *i.e.*, a formulation where each shift is entirely contained in a single time window. As mentioned by several researchers (see, *e.g.*, [7]), this assumption may result in considerable labor losses when compared to a continuous formulation of the problem.

The windowing procedure proposed in this paper considers overlapping time windows thus addressing a continuous formulation of the shift scheduling problem. A shift scheduling subproblem associated with a given time window thus incorporates all shifts that start within the time window and the corresponding breaks. The demand constraints considered are those associated with the periods belonging to the corresponding time window.

Let T denote the set of time windows resulting from a temporal decomposition of the planning horizon. One can easily define a total order relation $\prec$ on set T. In this order a time window t_l is smaller than a time window t_m if t_l starts before t_m. One can easily index the elements of T with respect to this total order relation so that $T = \{t_1, t_2, \ldots, t_n\}$ and $t_1 \prec t_2 \prec \cdots \prec t_n$, where n denotes the number of generated time windows. It is worth mentioning that $t_1 \cup t_2 \cdots \cup t_n = I$, the set of all periods of the planning horizon. In order to consider overlapping shifts present in continuous scheduling problems, time window t_{l+1} overlaps with its predecessor t_l (if this predecessor exists, *i.e.*, for $l \geqslant 2$) on all the periods that are covered by all shifts that overlap from t_l to t_{l+1}. All these overlapping shifts are then reconsidered in the shift scheduling sub-problem associated with time window t_{l+1}. Obviously, the demand constraints associated with periods in $t_l \cap t_{l+1}$ were satisfied when solving the shift scheduling problem over time window t_l. Since these periods are reconsidered in the shift scheduling sub-problem associated with t_{l+1}, one must update the right hand side of the corresponding demand constraints. Indeed, this update consists in subtracting from the original value of each of these demands the sum of all shifts that start and finish within t_l and for which the demand period is a work period (not a break). For cyclic problems, the last time window t_n not only overlaps with its predecessor but also with the first time window t_1. The periods covered in common by time windows t_n and t_1 are those covered by all shifts that overlap from t_n to t_1. Thus, for the shift scheduling subproblem associated with time window t_n, the right hand sides of demand constraints associated with periods in $t_n \cap t_1$ also need to be updated. This update consists in subtracting from the original value of each of these demands the sum of the all shifts that begin and end within t_1 and for which the demand period is a work period.

The final solution is obtained by combining all shift and break variable values yielded by the solution of all sub-problems. In fact, each shift and each break variable associated with shifts overlapping from one time window, t_l, to the next (t_{l+1}) will have two values. The value considered in the final solution is that obtained for time window t_{l+1}. In a cyclic case, the values considered for the variables associated with shifts that overlap from t_n to t_1 are those obtained for t_n.

The time windowing method presented above is general enough to be applied to any shift scheduling problem. The shift scheduling subproblems involve the

same types of variables and constraints as the original one, but on a smaller planning horizon. The only pre-processing needed for a given subproblem consists in updating the right-hand side of some demand constraints when overlapping time windows are considered. Such a pre-processing assumes that explicit shifts are obtained after the solution of the previous sub-problem. Hence, the difficulty of the pre-processing task depends on the formulation used to model the scheduling problem. In the case where only implicit solutions are yielded, one needs to define some procedures to obtain explicit solutions. Hence, for the formulations we propose in this paper and as explained in Section 3, formulation $(P1)$ yields explicit shifts $S_j, j \in J$ with explicit break assignments $X_{jk}, j \in J, k \in K$. Thus the update of demand constraints is straightforward with formulation $(P1)$. For formulation $(P2)$ a particular assignment problem $(P3)$ must be considered to derive explicit shifts from the implicit optimal solution obtained with model $(P2)$.

5. Computational experiments

The purpose of this section is to first compare different solution approaches for real-life shift scheduling problems arising from an air-traffic control agency. Three solution approaches are considered: the adapted local branching method described in Section 4.1, the windowing heuristic described in Section 4.2, and the branch-and-bound procedure of CPLEX 9.0. A second set of instances is then considered to show the performance of each of these procedures for generated shift scheduling problems traditionally considered in the shift scheduling literature.

5.1. Scheduling environments and problem size

The scheduling environments considered in the analysis operate 24 h a day and seven days a week. The operating day is divided into 96 periods of 15 min. Demand profiles are cyclic with a cycle length equal to one week. Thus, one need only determine the schedules on a one-week period.

The first set of instances includes three different operating environments, denoted hereafter by $ENV1$, $ENV2$ and $ENV3$, respectively. Each environment is characterized by a list of admissible shift types and work stretch parameters. A shift type is specified by a start time, a length and the fractionable and/or unfractionable breaks it must receive. Instances provided by the company involve up to 21 shift types and almost all shift types include a unique fractionable break. For $ENV1$, all shift types include only fractionable breaks. For $ENV2$ and $ENV3$, only one shift type incorporates a fixed break. In terms of work stretch parameters, all the instances consider three day-parts and thus three different values of minimum and maximum work stretch parameters during the day. Six demand patterns are specified for the first and third environments and two are specified for the second environment, resulting in 14 real-life instances.

The second set of generated instances includes two operating environments, denoted hereafter by $ENV4$ and $ENV5$, respectively. Environment $ENV4$ considers nine-hour shifts starting every hour. Each shift must receive two 15 min

relief breaks and a 90 min meal break. A break window is defined for each break and the minimum and maximum work stretch duration restriction parameters are set to 60 min and 180 min, respectively. Except the pre- and post-break work stretch duration constraints, the first hypothetical environment is a typical shift scheduling environment with multiple standard breaks and is often considered in the literature on shift scheduling problems (*e.g.*, [4]). The second hypothetical environment, $ENV5$, offers more flexibility in the definition of shifts and breaks. It considers nine-hour shifts starting every 30 min. Each shift must receive a 90 min fractionable break with a minimum and a maximum subbreak length of 15 min and 60 min, respectively. In order to ensure realistic operating conditions, the break profiles associated with shifts are restricted to those having exactly three subbreaks such that the length of the subbreak in the middle is strictly greater than the first and last ones, thus resulting in three different break profiles $(15, 60, 15)$, $(30, 45, 15)$ and $(15, 45, 30)$. Finally, the minimum and maximum work stretch duration restrictions are set to 60 min and 180 min, respectively. For each of these operating environments, the six demand patterns of the air-traffic control agency are considered.

It is worth recalling that the implicit models used in this experimental analysis correspond to models $(P1)$ and $(P2)$ described in Sections 3.3 and 3.4 for which all forward and backward constraints are reformulated with slack variables as proposed by Rekik *et al.* [18]. Table 1 reports the number of variables and constraints as well as the density of the constraint matrix of models $(P1)$ and $(P2)$ for the real-life scheduling environments of SET1 and SET2.

TABLE 1. Model size for SET1 and SET2.

	Number of variables		Number of constraints		Matrix density (%)	
ENV	(P1)	(P2)	(P1)	(P2)	(P1)	(P2)
1	22,274	21,481	12,558	14,486	0.04	0.04
2	16,996	15,624	9,065	10,100	0.05	0.06
3	9,639	7,943	5,670	5,801	0.09	0.10
4	9,408	5,901	5,208	4,373	0.09	0.12
5	62,496	21,291	31,920	14,835	0.01	0.04

As one can see from Table 1, the experimental study proposed here includes small (environment 3 and 4), medium (environment 2) as well as large scheduling problems (environments 1 and 5). Except for environment 5, models $(P1)$ and $(P2)$ are almost equivalent in terms of the number of variables and constraints, and the matrix density. It is worth noticing that the density of both models is very low (no more than 0.12%). For environment 5, model $(P1)$ has up to 62 496 variables and 31 920 constraints compared to only 21 291 variables and 14 835 constraints for model $(P2)$.

TABLE 2. Computational results for model ($P1$).

ENV	Demand pattern	LP	LB	Local Branching			CPLEX 9.0			Windowing		
				Value	Time (s)	Gap (%)	Value	Time (s)	Gap (%)	Value	Time (s)	Gap (%)
1	1	192.5	193	196	173.47	1.55	197	56.41	2.07	199	2701.70	3.11
	2	192.5	193	196	11.71	1.55	196	41.37	1.55	199	1807.90	3.11
	3	141.61	147	147	124.02	0.00	147*	40.66	0.00	153	36.80	4.08
	4	125.25	132	133	635.01	0.76	133	178.69	0.76	139	198.90	5.3
	5	183.38	186	189	21.47	1.61	190	368.47	2.15	192	2702.20	3.23
	6	199.91	202	204	2141.59	0.99	206	52.88	1.98	207	945.40	2.48
2	3	136.25	139	140	372.38	0.72	141	23.12	1.44	143	906.10	2.88
	4	119.00	122	126	4.84	3.28	126	65.05	3.28	131	2700.60	7.38
3	1	185.50	186	189	1.61	3.28	190	2.15	3.28	189	1803.00	1.61
	2	186.50	188	189	1973.24	0.53	190	15.62	1.06	189	3.30	0.53
	3	141.25	146	146	3.12	0.00	146*	10.88	0.00	146	3.30	0.00
	4	124.25	133	133	2.57	0.00	133*	1.51	0.00	133	2.60	0.00
	5	185.50	189	189	4.00	0.00	190	2.11	0.53	189	2.00	0.00
	6	200.57	203	203	8.24	0.00	203*	8.88	0.00	203	4.00	0.00
4	1	151.95	152	161	614.21	5.92	166	197.55	9.21	164	2553.00	7.89
	2	153.92	154	163	2321.72	5.84	167	446.25	8.44	162	1614.20	5.19
	3	117.32	118	124	42.00	5.08	129	105.83	9.32	124	36.20	5.08
	4	102.31	103	105	1775.72	1.94	114	26.83	10.68	109	1139.00	5.83
	5	163.66	164	170	551.86	3.66	168	1480.97	2.44	169	1911.10	3.05
	6	177.85	178	183	121.49	2.81	193	54.72	8.43	185	2700.80	3.93
5	1	144.84	145	154	604.17	6.21	158	2101.28	8.97	149	203.20	2.76
	2	145.48	146	153	3106.53	4.79	160	3107.79	9.59	148	1202.70	1.37
	3	112.00	112	-	-	-	-	-	-	115	1403.76	2.68
	4	98.00	98	-	-	-	-	-	-	108	3391.50	10.20
	5	148.40	149	-	-	-	-	-	-	156	3600.00	4.70
	6	161.66	162	-	-	-	-	-	-	171	3600.00	5.56

TABLE 3. Computational results for model ($P2$).

ENV	Demand pattern	LP	LB	Local Branching			CPLEX 9.0			Windowing		
				Value	Time (s)	Gap (%)	Value	Time (s)	Gap (%)	Value	Time (s)	Gap (%)
1	1	192.50	193	197	1164.90	2.07	199	374.77	3.11	196	2704.40	1.55
	2	192.50	193	198	735.45	2.59	199	541.58	3.11	200	2705.00	3.63
	3	141.61	147	150	3431.28	2.04	147	1963.87	0.00	151	224.90	2.72
	4	125.25	132	135	2103.41	2.27	135	639.36	2.27	134	287.10	1.52
	5	183.36	186	190	1358.99	2.15	189	1978.77	1.61	189	2704.50	1.61
	6	199.91	202	205	1131.13	1.49	209	757.97	3.47	204	963.60	0.99
2	3	136.25	139	140	227.45	0.72	140	457.97	0.72	140	2748.50	0.72
	4	119.00	122	126	318.97	3.28	126	265.10	3.28	128	2701.60	4.92
3	1	185.50	186	189	96.50	1.61	190	104.25	2.15	196	17.10	5.38
	2	186.50	188	189	96.50	0.53	191	35.14	1.60	199	13.10	5.85
	3	141.25	146	146	1092.46	0.00	146	39.82	0.00	146	3.30	0.00
	4	124.25	133	133	155.53	0.00	133	53.29	0.00	135	1804.00	1.50
	5	185.50	189	189	241.99	0.00	189*	116.39	0.00	189	17.90	0.00
	6	200.57	203	203	115.99	0.00	204	116.39	0.49	204	14.00	0.49
4	1	151.95	152	159	1527.26	4.61	166	123.43	9.21	167	2095.90	9.87
	2	153.92	154	162	1995.79	5.19	165	313.30	7.14	168	1822.60	9.09
	3	117.32	118	124	1321.70	5.08	131	178.08	11.02	129	92.3	9.32
	4	102.31	103	107	3265.78	3.88	111	777.79	7.77	112	104.00	8.74
	5	163.66	164	168	1480.97	2.44	175	175.55	6.71	177	3600.00	7.93
	6	177.85	178	184	1510.93	3.37	188	146.28	5.62	194	1108.60	8.99
5	1	144.84	145	-	-	-	-	-	-	154	1171.30	6.21
	2	145.48	146	-	-	-	-	-	-	156	2541.80	6.85
	3	112.00	112	-	-	-	-	-	-	115	1770.30	2.68
	4	98.00	98	-	-	-	-	-	-	104	1586.60	6.12
	5	148.40	149	-	-	-	-	-	-	162	2714.40	8.72
	6	161.658	162	-	-	-	-	-	-	177	2714.70	9.26

5.2. Comparison of the solution approaches

The three solution approaches were applied to both formulations ($P1$) and ($P2$) on a 2.66 GHz Xeon PC with a time limit of 3600 s. The local branching strategy and the windowing approach use CPLEX 9.0 with most parameters set at their default values. A branching strategy giving higher priority to shift variables S_j was considered for the three approaches. After a series of tests, we concluded that, on average, better solutions were obtained, for the three methods, with the "emphasis" parameter of CPLEX 9.0 set to "feasibility" (*i.e.*, when the search concentrates on finding feasible solutions rather than optimal ones).

For the local branching strategy, the size of the neighborhood (k) is set to 10. As illustrated at the end of the section, this value was found to be the most effective, on average. The time allowed to explore a local branching node is set to 300 s (this parameter corresponds in fact to the *node time limit* parameter defined by Fischetti and Lodi [12]). Finally, no more than five strong diversifications are permitted (this parameter corresponds to the *dv-max* parameter described in [12]). Recall that a strong diversification consists in restarting the local branching process from a new reference solution. The initial reference solution corresponds to the first feasible one given by CPLEX 9.0.

For the windowing approach, the seven-day planning horizon was divided into three time windows of two days each and a time window of one day. In fact, depending on the solution obtained for its predecessor, the current time window was enlarged as described in Section 4.2 to consider overlapping shifts. Each of these subproblems was solved within a time limit of 900 s.

The results obtained with the three approaches are reported in Tables 2 and 3. These tables display the LP and IP solutions obtained with each formulation for each instance as well as the time (in seconds) needed to find the IP solutions. In fact, Rekik *et al.* [18] proved that the LP relaxation of models ($P1$) and ($P2$) are equivalent. However, to clarify the presentation, we choose to report the LP column for each formulation.

It is worth recalling that the local branching method and the branch-and-bound procedure of CPLEX 9.0 are applied for the whole shift scheduling problem over the week. An optimal solution was thus very difficult to obtain in 3600 s. Hence, we report in the Value column the best feasible solution obtained (which represents an upper bound on the optimal integer solution) and mark it with an asterisk when it is proved to be optimal. The Time columns of the local branching strategy and the branch-and-bound procedure of CPLEX 9.0 display the time (in seconds) needed to find the best feasible solution during the search process. A dash ($'-'$) in a column indicates that no feasible solution was identified within the time limit. For the windowing approach, the reported solution values and times are those obtained at the end of the whole process. If no optimal solution was found over a given time window, a solution time equal to 900 s, *i.e.*, the time limit, is considered for the corresponding sub-problem. We also add in Tables 2 and 3 a column, denoted by "LB", which displays the best lower bound deduced from the classical branch-and-bound search of CPLEX (*i.e.*, by rounding up the "best node" value) as well

as a column denoted by "Gap" that illustrates the relative gap between the lower and upper bounds obtained for each approach and each formulation.

When compared to the branch-and-bound procedure of CPLEX, the local branching strategy applied to model ($P1$) yielded a better feasible solution for 15 instances over the 22 (68.18%) for which an initial feasible solution was obtained within the time limit. This ratio reached 60% for model ($P2$) (12 instances over 20). The difference between objective values reached 8.57% (see demand pattern 4 for ENV4) for model ($P1$) and 5.65% for model ($P2$) (see demand pattern 3 for ENV4). For the seven remaining instances formulated with model ($P1$), both procedures resulted in the same objective value. The branch-and-bound procedure of CPLEX was, however, computationally superior since it consumes an average time of 49.58 s to find these solutions compared to 112.78 s on average for the local branching method. Moreover, the branch-and-bound procedure was able to prove the optimality of four of the seven equal solutions. When considering model ($P2$), the proposed local branching procedure yielded the same solution as the branch-and-bound of CPLEX 9.0 for six instances (over 20) and worse solutions only for two instances; 2.04% worse for demand pattern 3 of ENV1, and 0.5% worse for demand pattern 5 of ENV1. The branch-and-bound procedure of CPLEX 9.0 is also computationally superior to the local branching strategy. In fact, it needed 261.98 s on average to find the six equal solutions compared to 689.96 s for the local branching strategy.

The results obtained with the windowing approach show, once again, that decomposing a continuous shift scheduling problem may considerably deteriorate the quality of the solution. This heuristic may appear inefficient for some easy instances that can easily be solved by local branching or traditional branch-and-bound procedures. However, as reported in Tables 2 and 3, except some instances (four in total for ($P1$) and five for ($P2$)), the windowing method results in feasible solutions that are comparable to those obtained with the first two solution methods. For more complex instances, for which a first feasible solution is hard to find, the windowing approach is able to yield better solutions. For example, for demand pattern 2 of $ENV5$ formulated with ($P1$), the windowing approach yielded a solution with 148 employees compared to 153 employees for the local branching method and to 157 employees for the branch-and-bound procedure of CPLEX. Moreover, the local branching and the classical branch-and-bound procedures were not able to find any feasible solution for four instances modeled with ($P1$) and for six instances formulated with ($P2$). On those instances, the windowing approach gave relatively good feasible solutions.

It is worth mentioning that a recent release of CPLEX (version 9.1) implements a local branching scheme. A new parameter controls whether this scheme is invoked by CPLEX or not. This parameter is off by default. We considered the branch-and-bound procedure of CPLEX 9.1 with this parameter turned on and found that the solutions obtained were either as good as or worse than those obtained with our own local branching procedure.

Moreover, one should notice that for the local branching strategy, the "best" value of parameter k (the neighborhood size) closely depends on the problem structure. We considered different values of parameter k (all other parameters were kept at their initial values, *i.e.*, a total time limit of 3600 s, a node time limit of 300 s and $dv_{max} = 5$) and found that a neighborhood size of 10 is the most effective on average. Tables 4 and 5 display the IP solutions and the corresponding best solution times obtained with models $(P1)$ and $(P2)$ for the instances of environment four with different values of k. Recall that the best solution time corresponds to the time needed to find the best feasible solution for the first time within the total time limit.

We also studied the impact of varying computing time parameters, *i.e.*, the total time limit and the node time limit, on solutions quality (while keeping other parameters at their initial values, *i.e.*, $k = 10$ and $dv_{max} = 5$). Tables 6 and 7 report the IP solutions and the corresponding best solution times obtained with models $(P1)$ and $(P2)$ for the instances of environment four with different values of the total time limit (denoted by TTL) and the node time limit (denoted by NTL). In fact, time parameters are varied in a way that keeps the ratio of the total time limit and the node time limit constant ($\frac{TTL}{NTL} = 12$). As one can see from these tables, on average, the best solutions (in terms of either the solution value or the computing time) are obtained for $TTL = 3600$ s and $NTL = 300$ s. This can be explained by the fact that 300 s are sufficient to identify improved feasible solutions while exploring a local branching node. In other words, considering a node time limit of 450 s or 600 s does not improve, in most cases, the solution found in 300 s.

6. Conclusions

This paper considers a continuous shift scheduling problem that includes a high degree of flexibility in defining shifts and breaks. It extends the shift scheduling problem studied by Rekik *et al.* [18] from one operating day to a w-day planning horizon. It thus incorporates the concepts of fractionable (or dividable) breaks and the pre- and post-break work stretch duration restrictions.

Two implicit formulations are presented. These formulations extend those proposed by Rekik *et al.* [18]. They incorporate generalized forward and backward constraints reformulated with slack variables to reduce the matrix density.

The complexity of the continuous scheduling problem studied was essentially due to the presence of shifts that overlap from one day to the next. Two solution approaches are proposed: a local branching and a windowing approach. The local branching method considers the problem as a whole over all the planning horizon. It is based on the local branching strategy initially introduced by Fischetti and Lodi [12] to solve complex MIP problems. We proposed particular local branching cuts that are specific to shift scheduling problems and used the same solution scheme as proposed by Fischetti and Lodi [12]. The windowing approach decomposes the problem into subproblems, one for each time window of the planning

TABLE 4. Computational results for the local branching approach for model ($P1$) with different values of k.

Demand	$k=2$		$k=5$		$k=10$		$k=15$		$k=20$	
Pattern	Value	Time (s)	Value	Time (s)	Value	Time (s)	Value	Time (s)	Value	Time (s)
1	161	3604.47	163	909.88	161	614.21	161	885.24	160	1327.24
2	165	3604.90	165	1242.27	163	2321.72	163	1952.28	163	1078.20
3	125	68.56	124	75.60	124	42.00	124	859.03	124	596.68
4	110	124.07	108	2126.62	105	1775.72	109	2143.28	109	896.41
5	170	617.92	170	2008.87	170	551.86	170	2772.74	171	185.31
6	184	294.41	184	151.54	183	121.49	183	2888.90	183	1026.06

TABLE 5. Computational results for the local branching approach for model (P2) with different values of k.

Demand	$k=2$		$k=5$		$k=10$		$k=15$		$k=20$	
Pattern	Value	Time (s)	Value	Time (s)	Value	Time (s)	Value	Time (s)	Value	Time (s)
1	162	3605.51	160	3565.70	159	1527.26	159	1033.83	160	2419.46
2	166	3606.21	164	1991.32	162	1995.79	162	2586.36	165	2083.07
3	127	3607.01	123	3616.16	124	1321.70	124	3740.63	126	1997.24
4	112	936.06	111	3600.17	107	3265.78	109	699.64	110	3457.69
5	171	544.43	172	2072.82	168	1480.97	169	1208.37	170	1029.91
6	184	1229.66	185	3602.50	184	1510.93	185	1165.56	185	623.65

TABLE 6. Computational Results for the local branching strategy for model ($P1$) with different values of the Time (s) limit parameters.

	TTL=1800		TTL=3600		TTL=5400		TTL=7200	
Demand	NTL=150		NTL=300		NTL=450		NTL=600	
Pattern	Value	Time (s)	Value	Time (s)	Value	Time (s)	Value	Time (s)
1	161	686.39	161	614.21	160	2222.05	160	2375.80
2	164	1001.16	163	2321.72	162	3256.21	162	3911.00
3	123	257.50	124	42.00	123	557.49	123	709.80
4	108	1138.47	105	1775.72	108	2644.91	108	3258.51
5	171	178.17	170	551.86	170	1029.12	170	1178.84
6	183	118.06	183	121.49	183	116.29	183	118.92

TABLE 7. Computational Results for the local branching strategy for model ($P2$) with different values of the time limit parameters.

	TTL=1800		TTL=3600		TTL=5400		TTL=7200	
Demand	NTL=150		NTL=300		NTL=450		NTL=600	
Pattern	Value	Time (s)	Value	Time (s)	Value	Time (s)	Value	Time (s)
1	160	974.67	159	1527.26	160	1432.88	160	2197.81
2	164	1709.52	162	1995.79	164	2614.92	162	6230.06
3	126	771.81	124	1321.70	125	2723.93	124	6976.66
4	111	1021.17	107	3265.78	109	3745.51	108	7136.36
5	173	1801.00	168	1480.97	169	3275.79	168	3364.10
6	186	548.66	184	1510.93	184	1532.97	184	1856.74

horizon. These time windows are enlarged during the process to handle the overlapping shifts from one time window to the next. In fact, when comparing the two proposed solution approaches, the local branching approach has the advantage of considering the overall problem and may thus yield better objective values than the windowing approach. Moreover, it may act as an exact solution method and

yield optimal solutions for some easy instances. However, a local branching strategy starts from an initial feasible solution that may be very difficult to obtain for some instances. The windowing approach has the advantage of always yielding feasible solutions in reasonable time, even for complex shift scheduling problems. The solution quality and computing time needed will depend on the way time windows are defined.

We compared the proposed approaches to the classical branch-and-bound procedure of CPLEX 9.0 for real-life instances obtained from an air-traffic control agency requirements and for some generated instances. Both local branching and windowing methods used CPLEX 9.0 to solve corresponding MIP sub-problems. The local branching method proposed yielded, on average, better objective values than the two other approaches. The results obtained with the windowing approach are a little worse especially for relatively easy instances. This was predictable since decomposing a continuous problem is likely to deteriorate the quality of the solution. However, on average, the solutions obtained with the windowing approach remain comparable to those obtained with the branch-and-bound procedure of CPLEX 9.0. For more difficult instances, whereas the branch-and-bound procedure of CPLEX 9.0 were not able to identify any feasible solution, the windowing approach yielded good feasible solutions.

Acknowledgements. This work was supported by the Natural Sciences and Engineering Research Council of Canada under grant CRDPJ228083-99 and by AD OPT Technologies Inc. This support is gratefully acknowledged. We are grateful to two anonymous referees for their valuable comments.

References

[1] I. Addou and F. Soumis. *Bechtold-Jacobs generalized model for shift scheduling with extraordinary overlap.* Technical report, GERAD, HEC Montréal, (2004).
[2] T. Aykin. Optimal shift scheduling with multiple break windows. *Manage. Sci.* **42** (1996) 591–602.
[3] T. Aykin. A composite branch and cut algorithm for optimal shift scheduling with multiple breaks and break windows. *J. Oper. Res. Soc.* **49** (1998) 603–615.
[4] T. Aykin. A comparative evaluation of modeling approaches to the labor shift scheduling problem. *Eur. J. Oper. Res.* **125** (2000) 381–397.
[5] S.E. Bechtold and L.W. Jacobs. Implicit modeling of flexible break assignments in optimal shift scheduling. *Manage. Sci.* **36** (1990) 1339–1351.
[6] S.E. Bechtold and L.W. Jacobs. Labor utilization effects of labor scheduling flexibility alternatives in a tour scheduling environment. *Decision Sciences* **24** (1993) 148–166.
[7] M.J. Brusco and L.W. Jacobs. Cost analysis of alternative formulations for personnel scheduling in continuously operating organisations. *Eur. J. Oper. Res.* **86** (1995) 249–261.
[8] M.J. Brusco and L.W. Jacobs. Starting-time decisions in labor tour scheduling: an experimental analysis and case study. *Eur. J. Oper. Res.* **131** (2001) 459–475.
[9] T. Çezik and O. Günlük. Reformulating linear programs with transportation constraints-with applications to workforce scheduling. *Naval Research Logistics* **51** (2004) 275–296.
[10] E. Danna, E. Rothberg, and C. Le Pape. Exploring relaxation induced neighborhoods to improve MIP solutions. *Math. Program.* **102** (2005) 71–90.

[11] G.B. Dantzig. A comment on Edie's traffic delays at toll booths. *Oper. Res.* **2** (1954) 339–341.
[12] M. Fischetti and A. Lodi. Local branching. *Math. Program.* B **98** (2003) 23–47.
[13] W.B. Henderson and W.L. Berry. Heuristic methods for telephone operator shift scheduling: an experimental analysis. *Manage. Sci.* **22** (1976) 1372–1380.
[14] D.S. Hochbaum and A. Levin. Cyclical scheduling and multi-shift scheduling: Complexity and approximation algorithms. *Discrete Optimization* **3**(4) (2006) 327–340.
[15] N. Mladenovic and P. Hansen. Variable neighborhood search. *Comput. Oper. Res.* **24** (1997) 1097–1100.
[16] S.L. Moondra. An L.P. model for workforce scheduling in banks. *J. Bank Res.* **6** (1976) 299–301.
[17] M. Rekik, J.-F. Cordeau, and F. Soumis. Using Benders decomposition to implicitly model tour scheduling. *Ann. Oper. Res.* **128** (2004) 111–133.
[18] M. Rekik, J-F. Cordeau, and F. Soumis. Implicit shift scheduling with multiple breaks and work stretch duration restrictions. Technical report, GERAD-2005-15, (2005).
[19] G.M. Thompson. Improved implicit optimal modeling of the labor shift scheduling problem. *Manage. Sci.* **41** (1995) 595–607.
[20] G.M. Thompson. A simulated annealing heuristic for shift scheduling using non-continuously available employees. *Comput. Oper. Res.* **32** (1996) 275–288.
[21] S. Topaloglua and I. Ozkarahan. Implicit optimal tour scheduling with flexible break assignments. *Computers & Industrial Engineering* **44** (2002) 75–89.