Interfaces
Vol. 35, No. 5, September–October 2005, pp. 402–410
ISSN 0092-2102 | EISSN 1526-551X | 05 | 3505 | 0402

DOI 10.1287/inte.1050.0154
© 2005 INFORMS

Scheduling Employees in Quebec's Liquor Stores with Integer Programming

Bernard Gendron
Département d'informatique et de recherche opérationnelle, Université de Montréal, C.P. 6128, succursale Centre-ville, Montréal, Québec, Canada H3C 3J7, gendron@iro.umontreal.ca

The SAQ (in French, Société des alcools du Québec) is a public corporation of the Province of Quebec responsible for distributing and selling alcohol-based products in its territory through a large network of more than 400 stores and warehouses. Every week, the SAQ has to schedule more than 3,000 employees. Until 2002, it handled this process manually, incurring estimated expenses of $1,300,000 (CAN). I developed a solution engine that interacts with a Web-based database system developed in house to produce the desired schedules. This solution engine implements an integer-programming (IP) model using ILOG Concert Technology and solves the IP formulation with ILOG CPLEX. The project has contributed to increasing the efficiency of the organization by reducing the costs of producing the schedules and by improving the SAQ's management of human resources. Overall, the SAQ estimates that automated scheduling has saved over $1,000,000 (CAN) annually.

Key words: organizational studies: manpower planning; programming: integer.

The SAQ (in French, Société des alcools du Québec) is a public corporation of the Province of Quebec responsible for distributing and selling alcohol-based products in its territory through a large network of more than 400 stores and warehouses. The SAQ operates different types of stores: some stores offer a large selection of products (for instance, those located in densely populated areas), while others have a limited selection (for example, those located near restaurants where customers can bring their own wine). The stores have various hours of operation depending on the day, but also on their type and location: they open between 9:30 AM and 12:00 AM, and close between 5:00 PM and 10:00 PM. The warehouses operate overnight.

Every week, the SAQ has to schedule more than 3,000 employees. Until 2002, it handled this process manually, incurring estimated annual salary expenses of almost $1,000,000 (CAN). Using the manual process, schedulers made many errors, because they were unable to produce solutions that respected all the complex rules of the union agreement. To deal with the complaints employees filed, the company estimates it paid costs of approximately $300,000 (CAN) annually. After carefully examining the available computer-based workforce scheduling products, the company realized that none of them could handle its union agreement rules properly.

In March 2000, the SAQ asked me to develop a solution engine that would interact with a Web-based database system developed in house to produce the schedules the SAQ needed. I chose integer programming (IP) as the methodology of choice and implemented it using a state-of-the-art IP software package (ILOG CPLEX). This choice allowed me to develop a robust program that produces *optimal* schedules, that is, schedules that strictly adhere to all union agreement rules.

Although it often happens that complex personnel-scheduling problems cannot be dealt with using compact IP formulations (most notably in the airline industry), this problem is quite different from many scheduling problems in that it decomposes by employee. The union agreement imposes a sequential assignment: The SAQ must assign the most senior employee the best schedule; then, using the remaining shifts, it must assign the best schedule to the second most senior employee, and so on. This sequential process is guaranteed to produce a feasible schedule, as there are always enough employees on the availability list to fill the requirements of all the stores (thus, there is no need to back-track on prior schedules). In spite of this interesting feature, formulating the problem for each employee is challenging. In particular, one rule allows the SAQ to split shifts of six hours or more between two employees. Alone, this rule can be formulated quite easily using IP, but when coupled to another rule that forces employees to take one-hour unpaid lunch or dinner breaks, it produces complex situations. I had to take into account several other complicating rules: multiple types of shift across multiple stores can be assigned to each employee (thus, there are travel time constraints between stores). In addition to maximizing the number of work hours, I had to consider several secondary objectives, and I had to model limits on daily and weekly work hours, as well as many other constraints.

Problem Description

The problem is to generate a weekly schedule per employee, given the following constraints:

—the employee cannot work more than 10 hours per day, and 38 hours over the whole week;

—a week starts on a Sunday and ends on the next Saturday;

—the union agreement specifies that the schedule be generated a day at a time, starting from the end of the week (Saturday) and going backward until Sunday: this rule is called the backward-assignment rule;

—the objective is to maximize the number of hours the employee works on each day by taking into account the shifts to be assigned and the availabilities the employee expresses.

Thus, the SAQ produces a weekly schedule for each employee by assigning a schedule for each day, starting from the end of the week and going backward to its beginning. The rationale behind the backward-assignment rule is to push the days off towards the beginning of the week (Sunday and Monday) and ideally to grant (the most senior) employees an additional day off (Tuesday). If it did not follow this rule, which maximizes the number of work hours on each day, but not over the whole week, the SAQ could produce schedules that contain more work hours over the whole week. I was not allowed to modify this rule, because my solution method had to strictly adhere to all union agreement rules.

Each day is divided into 15-minute periods. I call a set of consecutive time periods an *interval* and an interval worked entirely by the employee a *work interval*. The union agreement specifies that each work interval must consist of at least three hours. Although most workers are available only during the daytime to work in stores, some workers also work overnight in warehouses. Typically, an employee working at night should not work during the preceding or following day: the rest rule requires at least eight hours of rest after and before an overnight work interval. The SAQ classifies the work intervals in three categories: *daytime*, for those between 8:00 AM and 11:00 PM; *overnight*, for those between 9:00 PM on one day and 8:00 AM on the next; and *mixed*, for those that cross over the time intervals defining the two other categories (for example, the work interval 6:00 AM to 11:00 AM is mixed). Thus, when planning the schedule for a given day, in addition to the 24 hours of the day, one must also consider the last three hours of the day before. Because of the backward-assignment rule, it is easy to enforce the rest rule for every day of the week, including Sunday, provided that one knows when the employee stopped working on the preceding Saturday. For example, if the employee finished working at 9:00 PM on Saturday the week before, one cannot assign an overnight shift to that employee on Sunday.

The substitution rule specifies that on any given day, one can assign the employee to a guaranteed shift, unless one can identify another schedule that produces more work hours of the same work-interval category and in the same store. For example, suppose that the employee has a guaranteed daytime shift with six hours in store A, and that the maximum number of work hours is nine when the scheduler assigns the guaranteed shift. If a schedule with 10 work hours exists, including seven daytime work hours in store A, then substitution will take place. In this example, if one could replace "seven daytime work hours in store A" with "six daytime work hours in store A," or with "seven mixed work hours in store A," or with "seven daytime work hours in store B," substitution will not take place in any of these three cases, because the resulting schedules do not produce more than six daytime work hours in store A.

The employee might work in several stores on the same day; hence, one must enforce the travel-time rule, which guarantees the employee time to travel between the stores. When an employee is scheduled for two disjoint work intervals (for example, when the employee works in two stores), it creates a discontinuity. The rules permit no more than two such discontinuities (corresponding to three disjoint work intervals). In addition, the SAQ should not create schedules with discontinuities, unless they contain at least 1.25 work hours more than any schedules without discontinuities. For example, a schedule with seven hours and no discontinuity is preferred to a schedule with eight hours and one discontinuity, but not to a schedule that contains 8.25 work hours (irrespective of its number of discontinuities).

The employee must take a one-hour lunch break at noon if the work interval entirely contains the interval 10:30 AM to 3:30 PM: this is the lunch-break rule (similar rules apply for dinner, defined by the interval 3:30 PM to 8:30 PM, and for overnight shifts). Because the employee is not paid for this one-hour break, it is not counted as a work hour. This rule is easy to implement on its own but not when it is coupled to another rule that allows shifts to be split between two employees.

The split-shift rule allows the SAQ to break splitable shifts into two parts: the *piece*, which is assigned to the employee under consideration, and the *residual*, which subsequently will be assigned to another employee. Given that each work interval must contain at least three hours, splitable shifts must consist of at least six hours, so that both the piece and the residual have at least three work hours. The split-shift rule was designed to help the SAQ to create schedules that come close to reaching the daily limit on work hours. For example, if an employee has a guaranteed shift from 9:00 AM to 4:00 PM (six work hours) and there is a splitable shift from 12:00 AM to 7:00 PM (creating a shift of seven work hours), the piece from 4:00 PM to 7:00 PM should be assigned to that employee, leaving nine work hours and a residual from 12:00 AM to 4:00 PM.

The employees and the union appreciate the advantages the split-shift rule provides; it is, however, a nightmare for management. It also makes the scheduling problem very complex to formulate and solve. It is a challenge to accurately model the split-shift rule on its own; combining it with the lunch-break rule creates unpleasant interactions. First, in coupling the two rules, one must manage the residuals to keep track of the work hours. The principle is simple: if the employee is assigned a work interval of p work hours, created by splitting some shifts whose total number of work hours is n, then the residuals should not total more than $n-p$ work hours. This way, the employer pays for no more work hours than required. Although simple in principle, this rule is not so easy to manage in practice; in three cases, the scheduler must adjust the residual to satisfy it:

1. In the first case, it must remove one hour from the residual. For example, assume that it can assign only one shift, from 10:30 AM to 8:30 PM, for a total of eight work hours (lunch and dinner breaks are imposed). The employee being scheduled is available only from 10:30 AM to 4:00 PM; by splitting the shift, the scheduler assigns 4.5 work hours to that employee (plus the lunch break). The residual cannot start at 4:00 PM, because the resulting shift would contain 4.5 work hours (with no dinner break allowed because the work interval would not contain the interval 3:30 PM to 8:30 PM). The piece and the residual would then sum up to nine work hours, which would exceed the total of eight work hours for the original splitable shift. We thus have to remove one hour from the residual, creating a new shift from 5:00 PM to 8:30 PM.

2. In the second case, it must remove two hours from the residual. If we consider the same example, but we change the hours in which the employee is available to 10:30 AM to 3:00 PM, we would assign the employee the piece from 10:30 AM to 3:00 PM, for a total of 4.5 work hours (with no lunch break, because the work interval would not contain the interval 10:30 AM to 3:30 PM). The residual cannot start at either 3:00 PM or 4:00 PM, because in both cases, it would then contain 4.5 work hours (in the first case, the dinner break would be given, but not in the second case). Thus, the scheduler would have to remove two hours from the residual, leaving a shift from 5:00 PM to 8:30 PM.

3. In the third case, it would have to add one hour to the residual. For example, assume that the employee is available from 8:00 AM to 11:00 PM

and there are two splitable shifts, one from 8:00 AM to 2:00 PM (six work hours) and the other from 11:00 AM to 6:00 PM (seven work hours). We assign the employee the work interval from 8:00 AM to 6:00 PM by splitting the first shift at 11:00 AM and combining the piece 8:00 AM to 11:00 AM with the second shift of 11:00 AM to 6:00 PM. This work interval corresponds to nine work hours; by combining the two shifts, we created a lunch break, while neither of the two splitable shifts contained one. Because the residual runs from 11:00 AM to 2:00 PM, the total number of work hours would be 12, while the two original shifts totalled 13 work hours. Thus, the scheduler must add one hour to the residual and make it start at 10:00 AM.

These three cases affect the mathematical formulation of the problem, because the model must ensure that residuals always contain at least three hours.

Apart from managing the residuals, the SAQ must manage another complex situation created by the interaction of the split-shift rule and the lunch-break rule. As a simple example, suppose that an employee is available from 10:30 AM to 4:00 PM and that the scheduler can assign only one shift, which starts at 10:30 AM and ends at 7:30 PM, to that employee. Normally it would split the shift at 4:00 PM and assign 4.5 work hours (from 10:30 AM to 4:00 PM minus a one-hour lunch break) to the employee. But there is a "better schedule," which consists of splitting the shift at 3:15 PM and assigning 4.75 work hours to the employee (10:30 AM to 3:15 PM with no lunch break because the work hours do not entirely cover the interval 10:30 AM to 3:30 PM). The model has to forbid this type of split, called an *opportunistic split*.

Although the objective is to maximize the number of work hours, the model must include a penalty to capture the existence of at least one discontinuity in the schedule (a discontinuity is created when the employee is scheduled for two disjoint work intervals). To discriminate between two equivalent schedules, the SAQ defined seven secondary objectives, by order of importance:

1. Favor the preferred type of shift: Each employee can be assigned two different types of shift but prefers one over the other. If two schedules provide the same number of work hours, the model should select the one that favors the employee's preferred type of shift.

2. Minimize discontinuities: If the previous objective does not allow the model to discriminate between two schedules, it should favor the schedule with the fewest discontinuities. When one schedule has one discontinuity and the other (otherwise equivalent) has two discontinuities, both schedules would receive a one-hour penalty for having at least one discontinuity, but the model would choose the first.

3. Minimize the number of stores: If the previous objective does not allow the model to discriminate between two equivalent schedules, it will favor the one with the fewest stores assigned to the employee.

4. Minimize the number of split shifts: If the previous objective does not allow the model to discriminate between two equivalent schedules, it will favor the one with the fewest split shifts.

5. Favor preferred stores: Each employee provides the SAQ with a list of preferred stores in order of preference. The model attempts to satisfy this preference; if it cannot, the model changes the order of preference for the next day to favor the stores associated with the assigned shifts. This rule allows the model to improve the continuity of the schedule from one day to the next.

6. Favor the earliest periods: If all the above objectives do not allow the model to discriminate between two schedules, it will select the schedule consisting of work periods early in the day (starting at 8:00 AM; the employees like the overnight periods the least).

7. Maximize the number of shifts: If all the above objectives do not allow the model to choose between two possible schedules for an employee, the model will choose the one that maximizes the number of assigned shifts; the rationale is that doing so reduces the effort needed to produce the schedule for the next employee.

Implementation

I programmed a C++ code that interacts with the Web-based database system the SAQ developed to acquire and store data on its employees. The SAQ system creates three data files representing (1) the shifts each employee can work, (2) each employee's availabilities and preferences, and (3) a list of parameters (daily and weekly limits on the number of work hours, a limit on discontinuities, and so forth), which are fixed and identical for all employees. The third data file helps the SAQ to analyze the impact of changes in the values of the parameters and to adapt the program when some values change. The C++ code implements the mathematical model using ILOG Concert Technology and then solves the IP formulation with ILOG CPLEX. I calibrated the parameters of CPLEX to optimize performance. I observed one striking example of the effect of fine tuning CPLEX parameters when the time CPLEX (version 7.1) took to solve a particular instance dropped from 20 minutes to 20 seconds.

The SAQ obtains most schedules very quickly (a few minutes at most). However, for some senior employees working in large subdivisions with many stores, the IP models for the end of the week (Friday and Saturday) can take hours to solve. Usually, the number of splitable shifts is a good indicator of the difficulty of the problems; typically, if one day contains more than 10 splitable shifts, the resulting model will be very hard to solve. To produce the schedule on time every week, the SAQ has acquired two CPLEX licenses and has implemented a simple queueing system that ensures that it solves only one of these difficult instances at a time.

Typically, the store managers enter data for the coming week on Wednesday nights, and the SAQ sends schedules to the employees on Thursdays (sometimes, on Friday mornings). Three employees dedicate part of their time to the project: a computer analyst maintains the database and interface system and updates the CPLEX versions; an employee from the human resources department and a representative of the union ensure that schedules respect all rules of the union agreement and answer the store managers' and the employees' questions about the schedules the system produces.

The project started in March 2000, when the consultant in charge of implementing the Web-based database system approached me to see if I could produce a complete solution to the scheduling problem. My early developments focused on modeling the splitable shifts. I produced a first release of the C++ code in May 2000. I then discovered several difficulties related to the interaction of split shifts and unpaid breaks; I fixed these problems in the following months. After 13 releases that required multiple bug fixes, I released version 1.0 in December 2000: the format of the data files was very close to the actual existing format, and it implemented most rules. Between December 2000 and July 2001, I produced 11 other releases and then developed version 3.0: it included several rules that were not in the previous versions, including the substitution rule and the rules governing preferred types of shift and overnight shifts. After 12 other minor releases, I produced version 4.0 in August 2002; it allows users to stop the execution for an employee after some time and to restart it later using the schedule generated so far for that employee. This feature is used by the queueing system developed by the SAQ to make sure that no single employee becomes a bottleneck to the whole scheduling process. That same month, I released Version 5.0, which is compatible with CPLEX version 8.0. The SAQ also implemented the system in all the stores in the Province of Quebec during the summer of 2002. Prior to that, the SAQ gradually tested the system on a limited set of stores. The current version is 6.1, released in March 2005.

Impact on the Organization

The project has contributed in many ways to increasing the SAQ's efficiency by reducing its scheduling costs and by improving its management of human resources. By replacing manual scheduling, the automated process saves an estimated $750,000 (CAN) or more annually (about 80 percent of the total prior salary expenses). In addition, because the program produces accurate schedules that respect all the rules of the union agreement, employees make very few complaints; this reduction in complaints translates into annual savings estimated at about $250,000 (CAN) (90 percent of the total prior expenses related to employees' complaints). Overall, the SAQ estimates that the automated scheduling program saves over $1,000,000 (CAN) annually. Because developing the new scheduling system (over 2.5 years) cost around $1,300,000 (CAN), the payback period is less than two years.

In the stores, the system has greatly simplified the work of the managers and union representatives by eliminating paperwork, by simplifying the management of data, and overall by reducing the time dedicated to scheduling. In addition, the system interprets the union agreement rules in a uniform way in all stores across the province, which has eliminated many of the complaints union representatives made prior to its implementation.

From its beginning, the project involved all the people concerned: store managers, union representatives, and human resources personnel. These three groups have worked together to reach consensus and help the consultants in their quest for successful implementation and results. The project enhanced working relationships all across the organization: between the employees and the managers in the stores, and between the union and the human resources department. The solution method I developed contributed to this success.

Appendix

The IP Model

I formulated all the rules within the IP model. To avoid overloading the presentation, I describe only a subset of these rules to illustrate some of the main modeling difficulties and to emphasize the flexibility of the modeling approach.

I first introduce the notation for sets: I denotes the set of time periods; J, the set of shifts, partitioned into splitable shifts, J^D, and unsplitable shifts, J^U; S is the set of stores; K is the set of breaks (lunch, dinner, and overnight); and $f(I')$ and $l(I')$ denote the earliest and latest periods in interval I'.

Constraints

Assignment constraints: I introduce three types of binary variables:

$y_i = 1$: if period i is assigned;

$z_j = 1$: if shift j is assigned;

$x_{ij} = 1$: if splitable shift j is assigned at period i.

The assignment constraints then take the following form:

$$\sum_{j\in J_i^U} z_j + \sum_{j\in J_i^D} x_{ij} = y_i, \quad i \in I,$$

where J_i^U and J_i^D are the sets of unsplitable and splitable shifts, respectively, that include period i. These constraints ensure that no more than one shift can be assigned at each period.

Break constraints: I introduce the following binary variables:

$r_k = 1$: if the employee takes break k.

The constraints can then be written as follows:

$$r_k \geq \sum_{i\in I_k} y_i - |I_k| + 1, \quad k \in K,$$

where I_k is the set of periods corresponding to break k. These constraints simply state that the employee deserves a break if all periods in I_k are assigned. This is a simplified form of the break constraints, because other considerations must be taken into account. For example, overnight breaks are allowed if the employee works at least 8.5 hours at night, including all the periods in the interval 11:30 PM to 5:30 AM; the constraints must then be adapted by including additional variables to handle this case.

Work-hour constraints: I introduce the binary variables:

$y_i^W = 1$: if period i is worked.

The constraints that define work hours can then be written as follows:

$$\sum_{i\in I_k} y_i^W = \sum_{i\in I_k} y_i - \rho r_k, \quad k \in K,$$

$$y_i^W = y_i, \quad k \in K,\ i \notin I_k,$$

where $\rho = 4$ is the number of periods in a one-hour break. It would have been possible to avoid introducing the variables y_i^W; however, they help in defining the objective, specifically the criteria related to preferences. Using these variables, the constraint limiting the number of daily work hours can be written as follows:

$$\sum_{i\in I} y_i^W \leq \kappa,$$

where κ is the upper bound on the number of work time periods, based on the limits of 10 work hours per day and 38 work hours per week.

Discontinuity constraints: I first introduce variables for each period that corresponds to the beginning of a work interval:

$u_i = 1$: if the employee is assigned to period i but not period $i-1$.

These variables are defined by the following series of inequalities:

$$u_i \geq y_i - y_{i-1}, \quad i \in I,\ i \neq f(I),$$

$$u_i \leq y_i, \quad i \in I,$$

$$u_i \leq 1 - y_{i-1}, \quad i \in I,\ i \neq f(I).$$

These variables serve multiple purposes. First, they can be used to enforce the rule stating that each work interval must contain at least three hours:

$$u_i \leq y_{i'}, \quad i \in I,\ i < i' \leq \min(i + \tau - 1, l(I)),$$

where $\tau = 12$ is the number of time periods in each work interval. These constraints are valid under the assumption that no work interval can start after 9:00 PM, which is always satisfied because shifts starting after 9:00 PM are overnight shifts, which are scheduled as part of the next day.

I also use variable u_i to define discontinuities, because there is a discontinuity at period i if a work interval starts at period i and another work interval starts at some period $i' < i$:

$$u_i^D \geq u_i + u_{i'} - 1, \quad i \in I,\ f(I) < i' < i,$$

where $u_i^D = 1$ if there is a discontinuity at period i. No further constraints are necessary to define these variables, because the objective contains a criterion to minimize the number of discontinuities.

Using these variables, I specify the constraint that limits the number of discontinuities:

$$\sum_{i \in I} u_i^D \leq \phi,$$

where $\phi = 2$ is the maximum number of discontinuities. I also define a variable u_0^D, which assumes value 1 if there is at least one discontinuity:

$$u_0^D \geq u_i^D, \quad i \in I.$$

This variable will be used to penalize a schedule that contains at least one discontinuity.

Travel time constraints: I define the following binary variables:

$p_{is} = 1$: if period i is assigned to a shift in store s.

These variables are defined by the following constraints:

$$\sum_{j \in J_i^U \cap J_s} z_j + \sum_{j \in J_i^D \cap J_s} x_{ij} = p_{is}, \quad i \in I,\ s \in S_i,$$

where J_s is the set of shifts in store s and S_i is the set of stores that can be assigned at period i.

Given that $\delta_{ss'}$ is the minimum travel time between stores s and s', the travel time constraints can be simply written as follows:

$$p_{is} + p_{i's'} \leq 1, \quad i \in I,\ s \in S_i,\ i < i' \leq \min(i + \delta_{ss'}, l(I)),$$
$$s' \in S_{i'},\ s' \neq s.$$

These constraints ensure that two stores cannot be assigned within the time window defined by the minimum travel time between the two stores.

Split shift constraints: I define two types of variables representing split shifts:

$v_{ij} = 1$: if splitable shift j is split "forward" at period i, i.e., $x_{ij} = 1$ and $x_{(i-1)j} = 0$;

$w_{ij} = 1$: if splitable shift j is split "backward" at period $i - 1$, i.e., $x_{ij} = 0$ and $x_{(i-1)j} = 1$.

The following set of equations completely characterizes these variables:

$$x_{ij} - v_{ij} - x_{(i-1)j} + w_{ij} = 0, \quad i \in I,\ j \in J_i^D \cap J_{i-1}^D,\ i \neq f(I).$$

It is easy to state the constraint guaranteeing that each splitable shift can be split only once:

$$\sum_{i \in I_j} (v_{ij} + w_{ij}) \leq 1, \quad j \in J^D,$$

where I_j is the set of time periods containing shift j.

Finally, I give a simplified version of the constraints that allow a shift to be split only if the residual contains at least three hours:

$$\sum_{i \in I_j} x_{ij} + \tau \sum_{i \in I_j} (v_{ij} + w_{ij}) \leq |I_j|, \quad j \in J^D,\ |K_j| = 0,$$

where K_j is the set of breaks associated with shift j (for example, a shift from 10:30 AM to 8:30 PM has two associated breaks, lunch and dinner). It is easy to verify that these constraints are valid when the shift contains no breaks. The situation is, however, much more complex when one or two breaks are associated with the shift, because I then need to introduce auxiliary variables and to adapt the constraints to accurately represent the cases I described earlier. For the sake of clarity, I omit the details related to the formulation of these constraints, as well as those of the constraints necessary to eliminate the opportunistic splits, which are also complex.

Objective

I present only the first four components of the objective function. Assuming that we are maximizing, these four components can be described as follows.

Maximize the number of work hours: This is simply written as

$$\sum_{i\in I} y_i^W.$$

Penalize discontinuity: When there is at least one discontinuity, the following term is penalized, so as to favor a schedule with at most one hour less, but no discontinuity (the penalty term is specified at the end of this section):

$$-u_0^D.$$

Favor the preferred type of shift: The more the employee is assigned to a preferred type of shift, the better the schedule. Hence, first I need to define variables representing the assignment by type of shift for each period:

$o_{it} = 1$: if period i is assigned to type of shift t.

These variables are characterized by the following constraints:

$$\sum_{j\in J_i^U\cap J_t} z_j + \sum_{j\in J_i^D\cap J_t} x_{ij} = o_{it}, \quad i\in I,\ t\in T_i,$$

where J_t is the set of shifts associated with type of shift t and T_i is the set of types of shift that can be assigned at period i.

These variables are not sufficient, because we want to maximize the number of work hours for which the preferred type of shift is assigned. Thus, I need a variable that counts the number of work periods assigned to type t:

$o_{it}^W = 1$: if period i is worked and assigned to type of shift t.

I simply define these variables in terms of the o_{it} and y_i^W variables:

$$o_{it}^W \le y_i^W, \quad i\in I,\ t\in T_i,$$
$$o_{it}^W \le o_{it}, \quad i\in I,\ t\in T_i.$$

By associating with each type of shift a preference number θ_t ($\theta_t = |T|$ for the most preferred type and $\theta_t = 1$ for the least preferred one), I can now model the term of the objective related to the preferred types of shift as follows:

$$\sum_{i\in I}\sum_{t\in T_i} \theta_t o_{it}^W. \tag{1}$$

The formulation of the criterion related to store preference is very similar.

Minimize discontinuities: This term is simply defined as follows:

$$-\sum_{i\in I} u_i^D.$$

Global objective: The four terms are assembled together in a single objective that respects the hierarchy among the criteria:

$$M_1\sum_{i\in I} y_i^W - M_2 u_0^D + \sum_{i\in I}\sum_{t\in T_i}(M_3+\theta_t)o_{it}^W - \sum_{i\in I} u_i^D.$$

In this expression, M_3 represents a strict upper bound on the value of the term $\sum_{i\in I} u_i^D$; I use $M_3 = \phi + 1$, because $\sum_{i\in I} u_i^D \le \phi$. Similarly, I define M_1 so that it gives a strict upper bound on the term $\sum_{i\in I}\sum_{t\in T_i}(M_3+\theta_t)o_{it}^W$. Because κ is an upper bound on the number of work periods (thus an upper bound on $\sum_{i\in I}\sum_{t\in T_i} o_{it}^W$) and $\theta_t \le |T|$, I use $M_1 = \kappa(M_3 + |T|) + 1$. I use the same value to define M_2 except that I must add a penalty corresponding to one hour in case of a discontinuity: $M_2 = (\xi + \epsilon)M_1$, where $\xi = 4$ and ϵ is any small positive number, to ensure that any schedule with at least one discontinuity is dominated, unless it has more than one work hour more than any other schedule.

Acknowledgments

I am grateful to the following individuals, who contributed in so many ways to the success of the project: Robert Beaulieu, union representative; Jean-François Boucher, computer analyst; Luc Bourgeault, consultant; Jean-Philippe Brianchon, computer analyst; Patrick Dion, computer analyst; Brian Gibb, director of information technology; Sylvain Laroque, human resources representative; Claude Nadeau, store managers' representative; and Véronique Lucas, computer analyst. I express my apologies to any other persons that I may have forgotten and who were involved in one way or another in achieving the success of the project.

Lynda Nadeau, Director of Information Technology Applications Expertise Center, Société des alcools du Québec, 7501, rue Tellier, 2e étage, Montréal, Québec, Canada H1N 3W2, writes: "The purpose of this letter is to acknowledge the contribution of Professor Bernard Gendron to the success of a computer system called GASPER that produces the working schedules of all our employees across Quebec (some 3,000 employees working in more than 400 stores). Professor Gendron has designed a computer code that interacts with the Web-based database engine in GASPER to generate working schedules that adhere to all union agreement rules. Since its implementation in 2002, we estimate that GASPER has generated annual savings of the order of 1,000,000$. In addition, this project was the result of a close collaboration with the union and has greatly contributed to improve the working relations all across the organization.

"We are indebted to Professor Gendron for his contribution to the success of GASPER. We fully support his candidature for the Daniel H. Wagner Prize for Excellence in Operations Research Practice and we give him the authorization to present his work for the Société des alcools du Québec during the course of the competition."