Generating Personnel Schedules in an Industrial Setting Using a Tabu Search Algorithm

Pascal Tellier[1] and George White[2]

[1] PrairieFyre Software Inc.,
555 Legget Dr., Kanata K2K 2X3, Canada
pascal@prairiefyre.com
[2] School of Information Technology and Engineering,
University of Ottawa, Ottawa K1N 6N5, Canada
white@site.uottawa.ca

Abstract. We describe a system designed to be used in an industrial setting for the scheduling of employees in a Contact Centre. The demand for the employees' services is driven by an estimated forecast for each period of operation, currently set with a duration of 15 minutes. The employees are drawn preferentially from a homogeneous pool. The constraints are set by the conditions of employment and must satisfy the requirements both of the company and the desires of individual employees who may have vary diverse interests. The system uses an initial scheduling module that constructs a feasible schedule followed by an optimizing tabu search based heuristic optimizer to cast the final schedule. This system is evaluated both with artificially constructed data and real industrial data.

1 Introduction

One of the most important problems that arises in organizations composed of more than a few workers is the management of personnel. Issues that arise in this arena can be very complex and failure to observe the rules of play can result in a wide range of outcomes ranging from isolated grumbling, reduction in general staff morale, work stoppages, strikes, mass resignations and expensive lawsuits. The financial impact of these factors can be enormous.

The issues involved in this area are such things as:

- who does what jobs
- when do they do it
- where do they do it
- who do they do it with
- how much are they paid

These issues are formalized by the constraints whose origins lay in the customs of the local workplace culture, collective agreements, government legislation, and informal arrangements between the persons responsible for the scheduling of personnel and resources and those directly involved by the schedule. Errors, misunderstanding and misinterpretation of the schedules can raise a host of problems.

E. K. Burke, H. Rudová (Eds.): PATAT 2006, pp. 293–302. ISBN 80-210-3726-1.

A complicating factor is the emergence of a class of so-called "temporary workers" who work only as required. The constraints involved in the scheduling of these people can be more complicated to formulate and more difficult to respect because of the various start and stop times that can be different for each person and may depend on the day of the week, the month of the year, forecasted demand and/or a spectrum of other considerations specific to the industry. All this must be done within the framework of corporate objectives. The overall goal is to balance meeting the service level while optimizing the budget and respecting employees' rights and preferences to keep them happy.

2 Tour Scheduling in Contact Centres

The scheduling of personnel can often be accomplished in two phases, the phase that deals with time-of-day or shift scheduling, and the phase that deals with day-of-week scheduling. Baker [1] has named this type of labour scheduling *tour scheduling*. If the work force can be classified into different types (usually corresponding to different skill levels) it is described as *heterogeneous*. If we are dealing with one skill level, as is the case with this paper, it is called *homogeneous*.

Recent reviews of the tour scheduling literature have been published by Alfares [2] and by Ernst *et al*[3]. It is one thing to find *feasible* schedules *i.e.* schedules that satisfy all the staffing rules but quite another to find an *optimal* feasible schedule *i.e.* one that not only satisfies the rules but also minimizes (or maximizes) some objective function. Alfares [2] has found 14 different criteria used by various authors to formulate these objective functions. Of these criteria, the ones germane to the present problem are the total daily and weekly hours worked, under and over staffing and employee satisfaction.

Alfares has also partitioned the methods used for optimizing functions that incorporate these criteria into 10 categories. The work reported here uses the tabu search (TS) created by Glover [4].
As stated by Ernst *et al.*: [3]

> .... heuristics are generally the method of choice for rostering software designed to deal with messy real world objectives and constraints that do not solve easily with a mathematical programming formulation.

Contact centres are an integral part of many businesses. They are composed of their staff whose job it is to answer or initiate telephone calls, to answer questions or to solicit business. The literature on contact centres has been reviewed by Gans *et al.* [5]. Scheduling the staff in these centres is almost never solved to optimality due to the complexity of the requirements. Suboptimal solutions that can be used in practice are formulated using heuristic methods.

This paper describes a typical problem that arises in the scheduling of personnel in contact centres and describes a solution based on tabu search. The system, part of which is described here, was undertaken by PrairieFyre Software Inc.

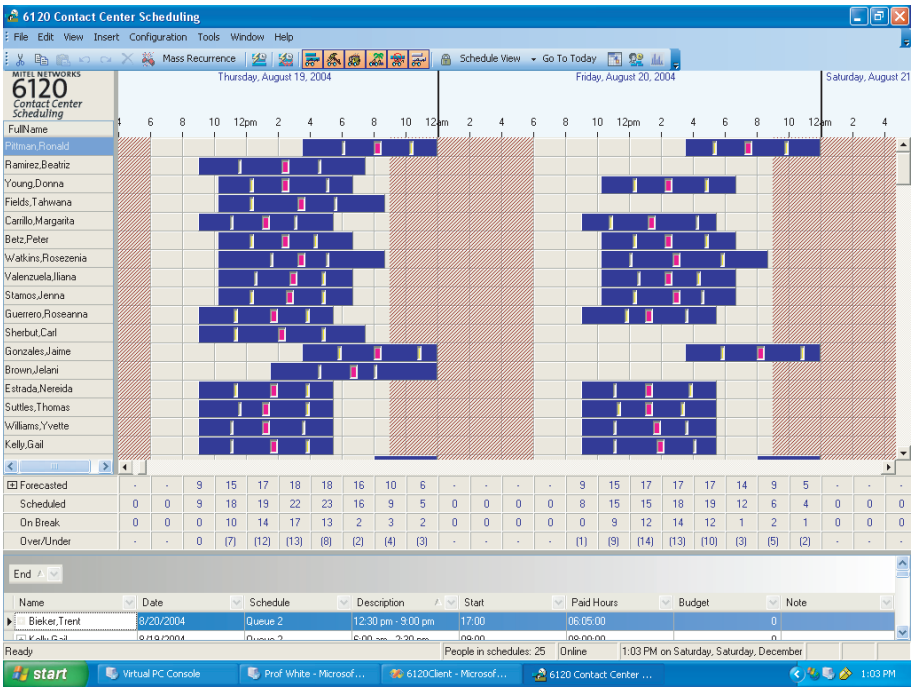

Fig. 1. Typical schedule

3 Problem Specifics

The problem at hand is to generate a schedule $s_{i,j}$ where: i is an integer index specifying a *person* and j is an integer index specifying a *period*, a fixed interval of time. $s_{i,j}$ represents the *state* of $person_i$ during $period_j$. This *state* can have values corresponding to: on-the-job, not working, on paid break, on unpaid break, on vacation, and a few other things. For our purposes we can simplify the value of $s_{i,j}$ such that it equals 1 when $person_i$ is on-the-job during $period_j$ and 0 otherwise. Then the number of persons working during $period_i$ is given by:

$$w_j = \sum_i s_{i,j}$$

The total number of person-periods working over the entire schedule is given by:

$$W = \sum_j w_j$$

For each $period_j$ a forecast f_j is obtained by some method such as reference to historical data or by a modified Erlang calculation. This forecast is an estimate of the number of persons "on duty" during the period concerned. This is just equal to the number of persons at work at the time minus the number having a

break. If $w_j - f_j$ is negative, there are too few persons working and the schedule is said to be underscheduled during that period. If the difference is positive, there are too many persons working during that period and the schedule is said to be overscheduled during that period. The *penalty* assessed for the entire schedule S is calculated as

$$P(S) = \sum_j (w_j - f_j)^2$$

In the field it may be desirable to change the form of the penalty to reflect the overall number of people being scheduled. A shortfall of 2 persons is not too important when 50 people are supposed to be scheduled, however a shortfall of 2 when 3 are to be scheduled can be very bad. A "typical" schedule looks like the one shown in figure 1.

Each person in the workforce and each shift have several important attributes. These are:

- Persons
  - name
  - scheduling priority
  - minimum hours between shifts
  - minimum daily hours
  - maximum daily hours
  - minimum weekly hours
  - maximum weekly hours
  - earliest start time (for each day of the week)
  - latest end time (for each day of the week)
- Shifts
  - name
  - type
  - typical hours
  - maximum hours
  - minimum hours
  - earliest start time
  - latest start time
- Breaks
  - name
  - duration
  - paid or not
  - shift length required to qualify
  - earliest start time after beginning of shift
  - latest start time after beginning of shift
  - minimum minutes before end of shift

The *persons* are the staff members to be scheduled. Each line on figure 1 is a graphic representation of the working day of a different person. The *shifts* are the periods during which a person is on the job. Each shift has zero or more *breaks* during which that person is off duty. A break may be paid or unpaid.

4 Constraints

A *feasible* schedule is one that satisfies a set of constraints. The constraint set used here can be divided into four parts.

4.1 Weekly Constraints

For all workers

- sum of working paid hours each week $\leq$ maximum weekly hours for that worker
- sum of working paid hours each week $\geq$ minimum weekly hours for that worker
- for all days, each person-shift must be separated by at least the minimum hours between shifts

4.2 Daily constraints

Each person-shift

- must start $\geq$ earliest start time for that day for that person
- must end $\leq$ latest end time for that day for that person
- must start $\geq$ earliest start time for that shift
- must end $\leq$ latest end time for that shift
- shift length $\leq$ maximum daily hours for that person
- shift length $\geq$ minimum daily hours for that person (if working that day)
- shift length $\leq$ maximum shift length
- shift length $\geq$ minimum shift length

4.3 Breaks

Each shift-break

- length of shift into which break is embedded $\geq$ shift length required to qualify
- start time $\geq$ earliest start time after beginning of shift
- start time $\leq$ latest start time after beginning of shift
- start time $\leq$ shift end time - minimum hours before end of shift

4.4 Global constraints

Since this work is designed for use in a real contact centre a number of practical features have been built in. These features generally remove some of the constraints, thereby reducing the penalty of an eventual solution. Some of these are

- override the available times specified by the employees
- override the days specified as unavailable by the employees
- calculate penalty only during "office hours"

There are six classes of overrides that can be used by the scheduling officers.

4.5 Other

Another feature completely inhibits optimization of one or more days of the schedule. Use of this feature will worsen the quality of the optimization but allows the unoptimized days to be scheduled "as is" *i.e.* exactly as cast by an existing initial schedule generator or cast manually or partially manually by the scheduling officer. Not all of the constraints have been implemented in the current version.

5 Goal

Each day of interest is divided into a number of periods of the same duration. The results described here refer to the case where the day consists of 96 periods, each one lasting 15 minutes. Each period is assigned a forecast of the number of workers required during that period. The number who are actually on duty should ideally be equal to the number forecast for that period. Each period is assigned a penalty equal to the square of the difference between the forecasted value and the number actually on duty. The penalty corresponding to a complete day is defined to be the sum of the penalties of all its periods. The goal of the scheduler is to cast a schedule such that all constraints are satisfied and the penalty of each day is as low as possible.

The forecasted number of employees required for each period is calculated by using a modified Erlang calculation. These numbers may be modified by a supervisor to account for local or unusual circumstances.

In practice, it may be preferable to attempt to minimize a penalty function that takes into account the total number of employees being scheduled and minimize the percentage deviation from the desired number of employees rather than the square of the absolute deviation. At present, however, we are concerned exclusively with absolute deviations.

With this explanation we can write formally that the problem consists in casting S such that:

1. $P(S)$ is minimized
2. the constraint set $c_i \odot a_i$ is satisfied
   - c_i is the left hand side of the constraint
   - $\odot$ is a relational operator
   - a_i is the constant on the right hand side

The implementation has been constructed in such a way that other possible goals can be easily incorporated. There are two alternate goals under consideration:

1. The number of *understaffed* periods is zero and the overall penalty as previously described is minimized.
2. The total dollar cost (salary) of the workers (including various rates of pay and overtime) is minimized while supplying a minimum level of service.

6 Tabu Search

The well known tabu search methodology [4] has proven to be a robust method of finding heuristic minimums of complex scheduling problems in reasonable times [6] [7] [8] [9] . Tabu search (TS) is a heuristic procedure designed to guide the search for an optimal schedule through the trap of local minima. Starting from an initial solution s_0 the TS algorithm iteratively explores a subset V^* of the neighbourhood $N(s)$ of a current solution s. The member of V^* that gives the minimum value of the objective function becomes the new current solution independently of whether its value is better or worse than the value corresponding to s. If $N(s)$ is not too large, it is possible and convenient to take $V^* = N(s)$. This strategy may induce cycling.

In order to prevent cycling, the algorithm calculates a so-called tabu list, a list of moves which the search portion of the procedure is forbidden to make. This is a list of the last k accepted moves (in reverse order) and it is often implemented as a queue of fixed size. At equilibrium, when a new move is added, the oldest one is discarded.

There is also a mechanism that may override the tabu status of a move. If a move could give a large improvement (reduction) of the objective function, then its tabu status is dropped and the resulting solution is accepted as the new current one. This is implemented by defining an aspiration function A that, for each value v of the objective function, returns another value v' that represents the value that the algorithm aspires to reach from v. Given a current solution s, the objective function $f(s)$, and a neighbour solution s', then if $f(s') \leq A(f(s))$ then s' can be accepted as the new current solution, even if s' is in the tabu list.

The procedure stops either after a given number of iterations without improvements or when the value of the objective function in the current solution reaches a given lower bound.

The main control parameters of the procedure are the length of the tabu list, the aspiration function A, the cardinality of the set V^* of neighbour solutions tested at each iteration, and $Tsmax$, the maximum number of iterations allowed that do not improve the objective function.

The tabu search strategy is implemented in the following way:

1. An initial schedule $s^0 \in X$ is chosen by an approximate but rapid method. X is a set of *feasible* solutions.
2. Certain variables controlling the stopping conditions are initialized here. These variables consist of the maximum number of iterations permitted, the maximum time the program is permitted to run and other similar quantities. These variables are used to determine whether the program should terminate or continue at step 4. The tabu list T is implemented as a linked list. The tenure was given a fixed value of 10. The value of the objective function corresponding to the initial solution is calculated.
3. The aspiration function is set to return the minimum value found so far.
4. At this point the program enters its main loop which exits when one of the following conditions become true:

   - the best solution found yet has a penalty = 0.
   - the number of iterations without improvement $\geq$ some maximum value, currently set to 1,000.
5. Each schedule $s \in X$ can be modified by applying a simple perturbation called a *move* to s. The neighbourhood, $N(s)$, consists of all feasible solutions that can be obtained by applying the perturbation to s. There are two separate move classes considered:
   - swapping two periods of one person
   - extending or shortening the schedule of one person by one period at the beginning or the end of their shift. This has the effect of either lengthening or shortening the shift or sliding it forwards or backwards by a small amount.

   For each person, these are applied sequentially. First all swapping possibilities are considered. Next, depending on whether the schedule is over or under scheduled, shortening or lengthening the shift is tried. Finally sliding the shift forward and backward is considered.
6. A subset of feasible solutions $V^* \subseteq N(s)$ is constructed from the elements of $N(s)$ in the following way. An element $s' \in N(s)$ is placed in V^* unless s' is in the tabu list T. However, even if $s' \in T$, it will be still be placed in V^* if $f(s') < A(f(s))$. $f(s')$ is the penalty or objective function associated with schedule s' and $A(f(s))$ is the value of the aspiration function associated with each value of the objective function f. Thus $A(f(s))$ is the value of the aspiration function associated with schedule s. In our work, $A(f(s)) = f(s^0)$ where s^0 is the best schedule found so far *i.e.* $f(s^0)$ is the lowest value of the objective function calculated as of yet. Note that s is not a member of its own neighbourhood. Therefore $s \notin V^*$.
7. The penalties $f(s^*)$ associated with each $s^* \in V^*$ are calculated and the value s^* having the lowest value $f(s^*)$ is chosen. This is done even if s^* is worse than the current schedule. There are cases where the shift and break requirements are so strict that no legal moves are possible. In this case, the step is exited immediately.
8. The tabu list is updated with the new solution found s^*. Rather than storing the entire solution in the list, the *move itself* is stored, making list management easier. Any list entries whose tenure ≥ 10 are discarded.
9. The new solution is compared to the best solution found so far s^0.
10. If the new solution is better, it replaces the now former best solution.
11. The current solution becomes the new solution.

7 The initial solution

The initial solution s_0 is constructed by an existing algorithm. The quality of this solution varies widely from instance to instance, sometimes producing good solutions and sometimes producing bad ones. The present program starts by calling a module we call the initial solution evaluator that analyzes the quality of this solution and may alter it severely, removing some shift instances completely if the solution is overscheduled and adding new shift instances if it is underscheduled.

8 Evaluation of the algorithm

The algorithm has been tested both with contrived data (test data) *i.e.* data that has been constructed to test specific features of the specification and the algorithm, and customer data *i.e.* real data furnished by existing customers. In both cases measurements have been made of the algorithm's ability to cast heuristically optimal schedules. These tests have been made using the module that starts by generating initial schedules and without using this module. In the latter case, the tabu search starts from an empty schedule and uses its own initial schedule evaluator to build a crude initial schedule and then uses the tabu portion to optimize it. The results are shown in the table below.

Table 1. Some test results

Name	length	conditional	before	after	reduction	% reduction	time (sec)
Test1	1 day	with initial	128	13	115	0.900	35.9
Test2	1 day	with	983	61	922	0.938	36.8
		without	5095	71	5024	0.986	33.0
Test3	7 days	with	1742	145	1597	0.917	218.0
Cust1	1 day	with	6241	32	6209	0.995	90.2
		without	1381	24	1357	0.983	13.6

The relative reduction in these examples is never less than 0.90, showing that the tabu algorithm is capable of reducing the penalty of a schedule to less than 10% of its initial value. The best case shown was obtained for customer data (from a real customer!) and reduced the penalty from 6241 to 32, or about one half of one percent of its original value.

Obviously the initial schedule has a heavy influence on the penalty of the final schedule. The relative reduction values that look so impressive are partially due to the bad results sometimes produced when the penalty of the initial schedule is calculated.

In all cases the TS algorithm was able to reduce the penalties of schedules significantly (better than 90%).

9 Implementation Details

The algorithm was coded in C# using the Microsoft Visual Studio .NET 2003 IDE. The results quoted in the table above were measured using a single processor Pentium 4 System with 1 GByte of RAM clocked at 2.40 GHz.

10 Conclusions

After testing with both artificial and real data the heuristic optimizer based on the classic tabu search paradigm was proven to generate employee schedules that

satisfy the constraints and have penalties much lower than schedules generated by the initial construction algorithm. Typically the penalty associated with an initial schedule is reduced by 90% or more.

A visual inspection of the final schedule often gives the impression that it is optimal, although one can never be sure. There does not appear to be a test suite for personnel problems such as the one available for examination schedules, so there is no way of comparing our results with those found using other systems.

References

1. K. R. Baker. Workforce allocation in cyclic scheduling problems: A survey. *Operation Research Quarterly*, 27:155–167, 1976.
2. H. K. Alfares. Survey, categorization, and comparison of recent tour scheduling literature. *Annals of Operations Research*, 127:145–175, 2004.
3. A. T. Ernst, H. Jiang, M. Krishnamuoorthy, and D. Sier. Staff scheduling and rostering: A review of applications, methods and models. *European Journal of Operational Research*, 153:3–27, 2004.
4. F. Glover and M. Laguna. *Tabu search*. Kluwer Academic Publishers, 1997.
5. N. Gans, G. Koole, and A. Mandelbaum. Telephone call centers: Tutorial, review and research prospects. *Manufacturing and Service Operations Management*, 5:79–141, 2003.
6. F. Glover and C. McMillan. The general employee scheduling problem: An integration of management science and artificial intelligence. *Computers and Operations Research*, 13:563–593, 1986.
7. G.M. White, B.S. Xie, and S. Zonjic. Using tabu search with longer-term memory and relaxation to create examination schedules. *European Journal of Operational Research*, 153:80–91, 2004.
8. E.K. Burke, G. Kendall, and E. Soubeiga. A tabu-search hyperheuristic for timetabling and rostering. *Journal of Heuristics*, 9(6):451–470, 2003.
9. Luca Di Gaspero and Andrea Schaerf. Tabu search techniques for examination timetables. *Lecture Notes in Computer Science*, 2079:104–117, 2001.