Tabu Search to Plan Schedules in a Multiskill Customer Contact Center

Florence Mendes, Corinne Lucet, Aziz Moukrim

Tabu Search to Plan Schedules in a Multiskill Customer Contact Center*

F. Mendes[1], C. Lucet[1] and A. Moukrim[2]

[1]LaRIA EA 2083, 5 rue du Moulin Neuf 80000 Amiens, France ((Florence.Mendes, Corinne.Lucet)@laria.u-picardie.fr)
[2]HeuDiaSyC UMR CNRS 6599 UTC, BP 20529 60205 Compiègne, France (Aziz.Moukrim@hds.utc.fr)

ABSTRACT

We have studied a realistic case of scheduling problem in a customer contact center, dealing with multiskill agents. Our model combines the last two steps of the standard approach by determining shifts and by assigning them to agents at the same time (scheduling and rostering). Moreover, we have considered realistic vacations, according to legal constraints and preferences of agents. We have envisioned entire weeks of work, with variable meal times and meal durations, without overtime. In this paper, we define the problem and describe a Tabu search based solution.
Keywords: Scheduling, Call Center, Tabu Search

1. INTRODUCTION

A call center handles by phone the customer contacts of several customer companies. If the call center uses also other means of communication such as email or post, it is called a customer contact center (CCC) or outsourcer. The main part of CCC's operating costs is labor costs, so it is an important advantage to optimize these costs. We are interested in a particular scheduling problem in a customer contact center, dealing with several services and multiskill agents. First we define the kind of scheduling problem that we want to resolve and recall some general characteristics and solution methods proposed in literature. In the third section, we present our modeling and describe a greedy algorithm that provides a first solution. In section four, we describe the neighborhood of a solution and improve the first solution with an algorithm based on Tabu search. Finally computational results obtained on real-world instances are discussed.

2. PREVIOUS WORK AND PROBLEM DESCRIPTION

The scheduling problem that is considered here consists in determining work schedules of CCC's agents for a given time horizon. The main part of operating costs in CCC is due to personnel[3], so good scheduling algorithms can substantially reduce the costs. Scheduling problems are deeply studied in litterature, but there is often a gap between models and the complexity of typical call centers design.

2.1. Previous work

A scheduling problem is generally decomposed into steps that are solved separately. Tien and Kamiyama [15] proposed three main stages to solve a scheduling problem, once the predictions of load are known. First step, named *allocation*, computes the number of agents needed for each period of the planning. It determines also the minimum number of agents to employ over the entire planning period. Second step, named *off day scheduling*, consists in assigning off days to agents according to off day and work stretch constraints. Third step, named *shift assignment*, consists in assigning shifts to the schedule according to shift assignment constraints and load predictions.

Ger Koole and al. [6] defined four standard steps that occur in many software packages: call volume estimation, calculation of minimum number of agents, determining shifts, assigning agents to shifts.

The first set-covering formulation for shift scheduling problem has been developed by Dantzig [8]:

$$Min\ z = \sum_{j=1}^{m} C_j X_j$$

$$\sum_{j=1}^{m} A_{ij} X_j \geq B_i,\ with\ i = 1, \ldots, n\ and\ X_j \geq 0,\ X_j\ integer,$$

where n corresponds to the number of time intervals during the planning horizon, m is the number of valid shifts, C_j is the cost induced by the assignment of vacation j, A_{ij} equals 1 if interval i is a work period of shift j and 0 otherwise, B_i is the number of required employees during the period i and X_j is the number of employees assigned to vacation j. First line minimizes the number of agents to employ and second line ensures that the number of agents assigned is enough for each time interval. This model has been improved to take into account other constraints such as multiple breaks or multiskill capacities[13, 5, 4, 7]. Gartner and Miksch proposed a CSP [10], Fukunaga and al. combined a CSP with AI techniques [9]. Avramidis and l'Ecuyer [3] proposed Mathematical Programming formulation of the multiskill staffing and the multiskill scheduling problems.

Another interesting way to find good solutions to difficult problems is to apply metaheuristics such as methods based on local search. Local search has been used efficiently by Musliu and al. [11, 14] to design shifts in a single-skill call center. We have chosen to use a similar model. Details of problem modeling and a greedy algorithm to construct a feasible solution are given in the next paragraph of this paper.

We tackle the combined problem of designing and assigning shifts and off days at the same time, as it has been done for other scheduling problems.

*1-4244-0451-7/06/$20.00

2.2. Preliminary assumptions

In our problem, the number of available agents and terms of their employment contracts are fixed. As the center is a multiskill CCC, some agents have multiple skills, and can be assigned to several tasks on one single day of work. We suppose that each agent has at least one skill. The total length of the scheduling, named time horizon, is divided into weeks, days, and time intervals of the same length (typically 10 or 15 minutes).

We suppose that the first step of the scheduling, consisting in determining for each skill and each time interval the number of required agents to ensure a certain service level, has already been computed ([1][2]). This is done by using the standard Erlang formulae, raised by a percentage that is daily determined by taking into account average absenteism for this kind of day or average load for similar days (the planner agent plays a part in this determination).

2.3. Problem constraints

Constraints of the problem can be divided into hard constraints and soft constraints. Hard constraints contain legal regulations, due to work laws and collective bargaining agreements: work duration per day, per week, and per month for each agent, minimal and maximal working time before meal break, minimal and maximal length of lunch breaks, etc. Moreover, some of the agents' individual preferences that are contractually defined have to be enforced: it includes off day constraints, shift change constraints and work stretch constraints. Hard constraints include also some technical constraints: agents haven't got the same skills and cannot be assigned to all tasks. Soft constraints include constraints that are relative to agents' wellbeing by trying to take into account the equity between agents : for each agent, we watch the number of working schedules per week, the number of skills used per day, the average length of the meal period, etc. We assume that agents have 2 or 3 off days, according to their employment contract. No overtime is allowed, and the working time of an agent for one week is always the same.

2.4. Feasible scheduling solution

A *vacation* represents one day of work, for one agent. The solution of the scheduling problem is a schedule of the vacations of each agent on a given planning horizon: weekly or monthly. The schedule indicates also the different skills that will be used by the agent during his working days. After scheduling, for each skill and each time period, we obtain the number of agents ideally necessary and the number of agents effectively planned. A *covering curve* can be designed, showing *excess* and *shortage* intervals. Our main objective is to obtain at least minimal service levels for minimal costs, by using flexibility in task assignments. Multiskill agents are scheduled for one or several skills during their day of work. Moreover, *equity* between agents has to be maintained, notably in assignment of the off days and in the average length of meal periods. These parameters are evaluated by measuring standard deviation with regard to average values over all agents.

3. PROBLEM MODELING AND CONSTRUCTION OF A FEASIBLE SOLUTION

In this section, we present our modeling of the CCC scheduling problem.We start with some definitions and describe in next paragraph a simple algorithm to construct a feasible solution.

3.1. Problem modeling

We consider *weeks* $(W_1, ..., W_{NBW})$, *days* $(D_1, ..., D_7)$ and *time intervals* $(I_1, ..., I_{NBI})$ where NBW is the total number of weeks in the planning horizon and NBI is the number of time intervals during one day (24 hours). We are given $(A_1, ..., A_{NBA})$ *agents*, $(C_1, ..., C_{NBC})$ work *contracts* and $(SK_1, ..., SK_{NBSK})$ different *skills*. To each agent corresponds one contract and a set of skills. Let an *Activity* be a set of tasks that requires certain skill for the agents. Activities are numbered from Act_1 to Act_{NBAct}. We are given $NBAct$ *Charge curves* such that, if Act is an Activity with inbound calls, then $Ch(Act, W, D, I)$ is the number of agents ideally necessary at time interval I of the week W and the day D for Activity Act. We assume that for each other kind of Activity, the number of tasks that have to be done per day is known. Let a *vacation type* be defined by:

- earliest and latest start S_min and S_max,
- a length L,
- earliest and latest meal period start M_min and M_max,
- minimal and maximal length of meal period ML_min and ML_max

The vacation types are numbered from VT_1 to VT_{NBVT}. An example of vacation types is given in table 1. A *vacation* $V_{Wi,Dj,Ai} = (V\_beg, V\_length, M\_beg, M\_length)$ represents one day of work, for one agent. V_beg corresponds to the beginning interval of the vacation, V_length corresponds to the length of the vacation, M_beg is the first interval of meal break and M_length its length. In the simple skill case, to a vacation corresponds also the skill that will be used by the agent during the day. For a given week, all vacations of one agent must belong to the same vacation type. A *valid vacation* is a vacation that respects law constraints and technical constraints. Let a *shift* for a week, a day, and an agent be defined by an interval of working time intervals and an Activity: $s_{W,d,a,x} = ([I_{beg}, I_{end}], Act_a)$. For a given week, each agent is assigned to X_a shifts. Each vacation corresponds to a set of shifts (the sum of the length of shifts for a day and an agent equals the V_length value of the vacation) and the intersection of shifts for one day and one agent is empty (an agent is assigned to at most one Activity during an interval). Shifts allow us to compute $NBAct$ *Presence curves* such that $Pre(Act, W, D, I) =$ the number of agents assigned to the Activity Act during the time interval I of the week W and the day D.

undercover and *overcover* curves list for each activity every period of shortage or excess of agents, by comparing for each time interval the number of agents needed and the number of agents assigned.

$uc[Act, W, d, i] = Ch[Act, W, d, i] - Pre[Act, W, d, i]$ if $Ch[Act, W, d, i] > Pre[Act, W, d, i]$, 0 otherwise. $oc[Act, W, d, i] = Pre[Act, W, d, i] - Ch[Act, W, d, i]$ if $Ch[Act, W, d, i] < Pre[Act, W, d, i]$, 0 otherwise.

A *scheduling solution* for a week W is made of the union of all shifts of all agents.

$$S_W = \bigcup_{a=A_1 \text{ to } A_N BA, d=1to7, x=1 \text{ to } X_a} s_{W,d,a,x}$$

In the scheduling solution, the sum of shortages, or *Under-Cover*(UC) has to be minimized.

$$UC(S_W) = \sum_{Act=1toNBAct} \sum_{d=1toNBD} \sum_{i=1toNBI} uc[Act, W, d, i]$$

We use boolean variables $Soff$ to evaluate equity between agents. $Soff(A_i, W_j) = 1$ if agent A_i works on Saturday for week W_j, 0 otherwise. Let $NS(A_i, W_j)$ be the number of Saturdays-off for agent A_i between week W_1 and week W_j:

$$NS(A_i, W_j) = \sum_{k=1toj} Soff(A_i, W_j)$$

Let $\overline{NS(W_j)}$ be the average number of Saturdays-off for all agents between weeks $W1$ and W_j. $SE(S_{W_j})$ is the *Saturday Equity* parameter value for solution S_{W_j} :

$$SE(S_{W_j}) = \frac{\sum_{i=1toNBA} |\overline{NS(W_j)} - NS(A_i, W_j)|}{NBA * j}$$

Let $NM(A_i, W_j)$ be the number of meal breaks taken by agent A_i between weeks W_1 and W_j. Let $DM(A_i, W_j)$ be the average length of meal break taken by agent A_i between weeks W_1 and W_j. Let $\overline{(DM(W_j)}$ be the average length of meal break for all agents between weeks W_1 and W_j. $ME(S_{W_j})$ is the *Meal Equity* parameter value for solution S_{W_j}:

$$ME(S_{W_j}) = \frac{\sum_{i=1toNBA} |\overline{DM(W_j)} - DM(A_i, W_j)|}{NBA}$$

Table 1: Example of vacation types

VT	S_min	S_max	L	M_min	M_max	ML_min	ML_max
1	08:00	10:00	07:20	11:00	14:10	00:40	01:10
2	09:00	11:00	07:20	11:00	14:10	00:40	01:10
3	10:00	12:00	07:20	11:00	14:10	00:40	01:10
4	12:00	14:00	07:20	16:00	18:00	00:30	00:30
5	13:00	15:00	07:20	16:00	18:10	00:30	00:30
6	14:00	15:40	07:20	17:00	19:30	00:30	00:30

3.2. Priority algorithm to generate a scheduling solution

A first solution to the scheduling problem is constructed with a greedy algorithm (see algorithm 1). This algorithm considers each week separately. We consider for each agent a number of working vacations according to his employment contract. First, agents are randomly numbered, then vacations of agents are determined according to this order, agent by agent, for the entire week. For each agent, the vacation type $VT(Ai, Wj)$ for the week and the off days are determined according to the maximal shortage of agents among all the skills of the agent. For each vacation and each Activity that is in the skills of the agent, the sum of undercovers $\sum_i uc[Act, W, d, i]$ for time intervals i corresponding to the vacation type VT is determined. Activity Act that involves the maximal sum of undercovers is chosen. Agent vacations (V_beg,M_beg,M_length) are determined successively according to priority rules, inspired by the heuristic [14] : a good shift starts when requirements increase and ends when requirements decrease. So, when evaluating different starting intervals for a vacation, the maximal value is given when working hours correspond to a load increasing of Activity Act. Finally, shifts of agents that have been constructed correspond to a scheduling solution S_W and the UnderCover of the solution $UC(S_W)$ is known.

Algorithm 1 Greedy algorithm

Input: agents contraints and charge courbes for week W
Output: S_W and UC_{S_W}
  Order agents from 1 to NBA
  FOR each Activity, each Day,and each Interval **DO**
    $uc[Act, W, D, I] = Ch[Act, W, D, I]$;
  END FOR
  FOR each agent A **DO**
    Determine a vacation type VT
    Order days from D_1 to D_7
    FOR $D = D_1$ to $D_{NbVacations(A)}$ **DO**
      Determine activity Act;
      Determine V_beg,M_beg and M_length;
      $V_{W,D,A} = (V\_beg, V\_length, M\_beg, M\_length)$;
      $s_{W,D,A,1} = ([V\_beg, M\_beg], Act)$;
      $s_{W,D,A,2} = ([M\_beg + M\_length, V\_beg + V\_length + M\_length], Act)$;
      FOR each time interval I of the vacation **DO**
        Compute $uc(Act, W, D, I)$;
      END FOR
    END FOR
  END FOR
  Compute UC_{S_W}

4. TABU SEARCH METHOD

The greedy algorithm produces non satisfying solutions. It assigns agents to only one Activity per day and does not bother about Equity constraints. Solutions involve Under-Cover on some Activities whereas some agents are in excess on other Activities. We want to improve iteratively the quality of our scheduling solution, by choosing at each step a solution that is near from the current solution and that is of better quality : considering first the UnderCover value, and then the equity parameters (see algorithm 2).

The moves that are allowed when exploring the neighborhood of the current solution are based on agents or activities. Some of these moves aim at improving the Under-Cover value (UC), the others aim at improving equity values ME and SE. Four kinds of moves are used to generate the neighborhood : changing the starting interval of a vacation for an agent, changing the Activity assigned to an agent during a set of time intervals, swap all the vacations between two agents, swap one vacation between two agents.

Algorithm 2 Evaluation of the neighbor solution

Input: S and S' two neighbor solutions
Output: The best solution $Best$

```
IF UC(S) < UC(S') THEN
  Best = S
ElSE
  IF (UC(S) = UC(S') and (SE(S) < SE(S')) THEN
    Best = S
  ElSE
    IF (UC(S) = UC(S')) and (SE(S) = SE(S')) and (ME(S) < ME(S')) THEN
      Best = S
    ElSE
      Best = S'
    END IF
  END IF
END IF
```

4.1. Exploring the neighborhood

We don't generate all the neighbors of a solution, because the entire neighborhood would be too large. Some conditions are common before applying any move : the new vacations obtained after the move must be valid vacations and correspond to the same vacation type as the initial vacation. So, only the moves that allow to construct a feasible scheduling solution are generated. At each step and for each move a set of agents is selected and moves are applied only to agents of this set. When no better solution is found after a number of iterations, the number of selected agents is increased.

Move Vacation: Moves $MoveVacationLeft$ and $MoveVacationRight$ consist in changing the starting interval of a vacation, without changing its length. This moves are applied when the starting or ending period of vacation shifts corresponds to a shortage of workers. The shifts are moved of one time interval in order to minimize undercover.

Selection of agents for an earlier beginning : Consider agent A_i with vacation $V_{W,D_j,A_i} = (V\_beg, V\_length, M\_beg, M\_length)$. $V\_end = V\_beg + V\_length$ and $M\_end = M\_beg + M\_length$. Let S'_W be the solution obtained by applying $MoveVacationLeft$ to solution S_W and agent A_i. We obtain a new vacation V'_{W,D_j,A_i} with $V'\_beg = V\_beg - 1$, $V'\_end = V\_end - 1$, $M'\_beg = M\_beg - 1$ and $M'\_end = M\_end - 1$.

- If $oc[Act, W, D, V\_beg - 1] > 0$, $oc[Act, W, D, M\_end - 1] > 0$, $uc[Act, W, D, M\_beg] > 0$ and $uc[Act, W, D, V\_end] > 0$ then $UC(S'_W) < UC(S_W)$
- If $oc[Act, W, D, V\_beg - 1] \geq 0$ and $oc[Act, W, D, M\_end - 1] \geq 0$ then $UC(S'_W) \geq UC(S_W)$

The move is applied only to a restricted number of agents. Agents corresponding to the first item are chosen. If such agents don't exist, agents who don't correspond to the second item are randomly chosen. The same reasoning is used for the selection of agents before applying $MoveVacationRight$.

Change Activity: This move consists in assigning to an agent another Activity during a given time period. It allows us to use the multiple skills of the agents. The application of this move reduces the undercover for an Activity, without inducing undercover on any other activities. A shift of the agent is split into several shifts implying several skills.

Selection of agents : If we change the Activity assigned to an agent which may be needed for his initial Activity, the gain obtained on the second Activity may not compense the undercover induced on the first Activity. So, the best gain is expected when the move satisfies these conditions :

- During the shift period, there exists a shortage of workers for another Activity.
- The agent is in excess in his Activity during this period of his shift.
- The Activity which is in shortage belongs to the skills of the agent.

For each agent we restrict the exploration to moves that allow a maximal swap length.

Change Week: This move consists in exchanging shifts of all the days of week W between agents A_i and A_j, in order to minimize the SE parameter. $ChangeWeek$ move does not affect UC value.

Selection of agents : To produce a valid scheduling, skills used by agent A_i(resp. A_j) during week W have to be in the skills of agent A_j(resp. A_j). Let S_W be the initial scheduling solution and S'_W the scheduling solution obtained after applying $ChangeWeek$ between A_i and A_j. If both agents A_i and A_j work on Saturday (or do not work on Saturday), then $SE(S'_W) = SE(S_W)$. Otherwise, if $NS(A_i, W) < \overline{NS(W)} < NS(A_j, W)$ then $SE(S'_W) < SE(S_W)$. The move is applied priorly to agents that ensure to improve SE.

Change Day: This move consists in swapping vacations between agents A_i and A_j for a given day D in order to minimize the ME parameter. $ChangeDay$ does not affect UC and SE values.

Selection of agents : Let S_W be the initial scheduling solution and S'_W the scheduling solution obtained after applying $ChangeDay$ between A_i and A_j. Agents are selected according to the following property : if $DM(A_i, W) > \overline{DM(W)}$, $DM(A_j, W) < \overline{DM(W)}$, $M\_length(A_i, D) > \overline{DM(W)}$ and $M\_length(A_j, D) < \overline{DM(W)}$ then $ME(S'_W) < ME(S_W)$.

4.2. Tabu algorithm

Tabu search method has been introduced by Glover in 1977 and is based on local search(see [12] for a complete description). The main interest of this method is to avoid cycles during the local search. We maintain a general Tabu

list composed by the latest moves that led to the current solution and which runs as a FIFO list.

The moves are applied successively to the current solution, as far as it can be improved and provided the maximal number of iterations authorized is not reached. At each step, the neighborhood of the current solution is explored, but is never totally generated. For each move, only few neighbors are evaluated. The best solution which is not tabu or which improves the solution S_{best} becomes the new current solution. The current solution replaces the best solution, otherwise the algorithm stops after a few number of attempts.

The algorithm ends when the maximal number of iterations has been met or when no improvement is found during MAX_ITER iterations.

Algorithm 3 Tabu Search algorithm

Input: initial solution S_{init}
Output: improved solution S_{best}
$i = 0; M = 0; S_{best} = S_{init}; S_{curr} = S_{init}$;
WHILE $i \leq MAX\_ITER$ **DO**
$i = i + 1; M = M + 1 (mod 5)$;
Apply $MoveM$ to $NBSelect$ agents
Choose the new S_{curr}
Add move to tabu list
IF S_{curr} is better than S_{best} **THEN**
replace it
END IF
END WHILE

4.3. Numeric Results

We have implemented and tested the Tabu Search Algorithm on real-world instances. Results of these experiments are reported in Table 2 and Table 3. For each move, at most 100 neighbors are visited. At most 25000 solutions are tested. We consider two Activities. The first Activity is open from 8am to 8pm from Mondays to Saturdays, the second one is open from 8am to 11pm from Mondays to Saturdays. We use time intervals of 10 minutes. We had to generate schedules for 120 agents, during 6 weeks of work. Some of the agents have a working contract that specifies working periods from 8am to 8pm, whereas others can work up to 11pm.

Table 2: Cover Results of Tabu Search algorithm

W	Act1		Act2		Global Cover	
	G	TS	G	TS	G	TS
1	92.21%	98.78%	86.73%	97.01%	90.09%	98.09%
2	80.66%	86.94%	84.21%	88.32%	82.81%	87.78%
3	81.54%	88.47%	85.02%	89.73%	83.69%	89.25%
4	85.59%	93.36%	87.39%	91.12%	86.63%	92.07%
5	85.08%	92.89%	86.81%	91.03%	86.09%	91.80%
6	83.20%	91.26%	88.39%	92.32%	86.48%	91.93%

Table 2 shows the cover results obtained on Activities $Act1$ and $Act2$, by algorithms Greedy (columns G) and Tabu Search (columns TS). The last columns indicate the global percentage of charge load that is covered by each algorithm. To each line of the table corresponds one week of work.

Table 3: Equity Results of Tabu Search algorithm

W	ME		S E	
	G	TS	G	TS
1	2.61	2.44	0.42	0.42
2	1.32	0.56	0.28	0.23
3	0.57	0.27	0.18	0.07
4	0.32	0.17	0.14	0.12
5	0.21	0.17	0.10	0.04
6	0.18	0.13	0.09	0.08

Table 3 shows the equity results obtained for each week by algorithms Greedy and Tabu Search. Columns ME indicate the Meal Equity parameter values and columns SE indicate the Saturday Equity parameter values.

Tabu Search improves largely the quality of solutions, as well for the load covering as for equity between agents. Equity between agents has been improved by the moves Change Week and Change Day. After 6 weeks the average length of meal periods are very near (we obtain almost equality if we consider an historical record of 10 weeks). Results about Saturday off days are also satisfying : after 6 weeks, each agent has had 2 or 3 Saturdays off.

We have compared our schedules with the schedules used currently in the Customer Contact Center. The schedules provided by our Tabu algorithm are better than the manual ones, improving the global cover of at least 5 to 13 percents.

5. CONCLUSIONS

In this paper, we have presented our work about shift scheduling in a multiskill customer contact center. The resolution method, based on local search, allowed us to provide to the CCC an automated solution to its scheduling problem. The quality of our algorithm solutions is better than the ones computed manually until today in this CCC. However, we hope to improve the quality of our solutions by introducing new moves in the neighbourhood of a solution.

REFERENCES

[1] J. Atlason, M.A. Epelman, and S.G. Henderson. Call center staffing with simulation and cutting plane methods. *Annals of Operations Research*, 127:333–358, 2004.

[2] A.N. Avramidis, A. Deslauriers, and P. L'Ecuyer. Modeling daily arrivals to a telephone call center. *Management Science*, 50(7):896–908, 2004.

[3] A.N. Avramidis and P. L'Ecuyer. Modeling and simulation of call centers. In *Proceedings of the 2005 Winter Simulation Conference*, Orlando, FL USA, 2005.

[4] T. Aykin. Optimal shift scheduling with multiple break windows. *Management Science*, 42(4):591–602, 1996.

[5] S.E. Bechtold and L.W. Jacobs. Implicit modeling of flexible break assignments in optimal shift scheduling. *Management Science*, 36(11):1339–1351, 1990.

[6] S. Bhulai, G. Koole, and A. Pot. Simple methods for shift scheduling in multi-skill call centers. *submitted*, 2005.

[7] X. Cai and K.N. Li. Genetic algorithm for scheduling staff of mixed skills under multi-criteria. *European Journal of Operational Research*, 125:359–369, 2000.

[8] G.B. Dantzig. A comment on edie's traffic delays at toll booths. *Operational Research*, 2(3):339–341, 1954.

[9] F. Fukunaga, E. Hamilton, J. Fana, D. Andre, O. Matan, and I. Nourbakhsch. Staff scheduling for inbound call centers and customers contact centers. *AI Magazine*, 2002(4):30–40, 2002.

[10] J. Gartner and S. Miksch. Shift scheduling with the projections first strategy. *Osterreiches Forshunginstitut fur AI*, 1995.

[11] J. Gartner, N. Musliu, and W. Slany. Rota : a research project on algorithms for workforce scheduling and shift design optimization. *Artificial Intelligence Communications*, 14(2):83–92, 2001.

[12] F. Glover and M. Laguna. *Tabu Search*. Kluwer Academic Publishers, 1997.

[13] S.L. Moondra. An lp model for workforce scheduling in banks. *Journal of Banks Ressources*, 6(4):299–301, 1976.

[14] N. Musliu, A. Schaerf, and W. Slany. Local search for shift design. *European Journal of Operational Research*, 153(1):51–64, 2004.

[15] J. Tien and A. Kamiyama. On manpower scheduling algorithms. *SIAM Review*, 24:275–287, 1982.