International Journal of Computational Intelligence Research.
ISSN 0973-1873 Vol.2, No. 4 (2006), pp. 309–326
© Research India Publications http://www.ijcir.info

# Heuristic Methods for Automatic Rotating Workforce Scheduling

**Nysret Musliu**[1]

[1]Vienna University of Technology
Karlsplatz 13, 1040 Wien, Austria
*musliu@dbai.tuwien.ac.at*

*Abstract*:
**Rotating workforce scheduling appears in different forms in a broad range of workplaces, such as industrial plants, call centers, public transportation, and airline companies. It is of a high practical relevance to find workforce schedules that fulfill the ergonomic criteria of the employees, and reduce costs for the organization.**
**In this paper we propose new heuristic methods for automatic generation of rotating workforce schedules. To improve the quality of each heuristic method alone, we further propose the hybridization of these methods. The following methods are proposed: (1)A Tabu Search (TS) based algorithm, (2) A heuristic method based on min-conflicts heuristic (MC), (3) A method that includes in the tabu search algorithm the min-conflicts heuristic (TS-MC) and random walk (TS-RW), (4) A method that includes in the min-conflicts heuristic the tabu mechanism (MC-T), random walk (MC-RW), and both the tabu mechanism and the random walk (MC-T-RW). The appropriate neighborhood structure, tabu mechanism, and fitness function, based on the specifics of the problem are proposed. The proposed methods are implemented and experimentally evaluated on the benchmark examples given in the literature and on the real life test problems, which we collected from a broad range of organizations. Empirical results show that the combination of the min-conflicts heuristic with tabu search can be used to solve this problem very effectively. The hybrid methods improve the performance of the commercial system for generation of rotating workforce schedules and are currently in the procees of being included in a commercial package for automatic generation of rotating workforce schedules.**

*Keywords*: rotating workforce scheduling, heuristic search, min-conflicts heuristic, tabu search, random walk, constraint satisfaction.

## I. Introduction

Workforce scheduling, in general, includes sub-problems which appear in many spheres of life as in industrial plants, hospitals, public transport, airlines companies etc. In Table 1 a typical representation of workforce schedules is presented. This schedule describes explicitly the working schedule of 9 employees during one week. The employees work in three shifts: day shift(D), afternoon shift(A) and night shift(N). The first employee works from Monday until Friday in a day shift (D) and during Saturday and Sunday has days-off. The second employee has a day-off on Monday and works in a day shift during the rest of the week. Further, the last employee works from Monday until Wednesday in a night shift (N), on Thursday and Friday has days-off, and on Saturday and Sunday works in the day shift. Each row of this table represents the weekly schedule of one employee.

*Table 1*: A typical week schedule for 9 employees

| Emp./day | Mon | Tue | Wed | Thu | Fri | Sat | Sun |
|---|---|---|---|---|---|---|---|
| 1 | D | D | D | D | D | - | - |
| 2 | - | D | D | D | D | D | D |
| 3 | D | - | - | N | N | N | N |
| 4 | - | - | - | - | A | A | A |
| 5 | A | A | A | A | - | - | - |
| 6 | N | N | N | N | N | - | - |
| 7 | - | - | A | A | A | A | A |
| 8 | A | A | - | - | - | N | N |
| 9 | N | N | N | - | - | D | D |

There are two main variants of workforce schedules: rotating (or cyclic) workforce schedules and non-cyclic workforce schedules. In a rotating workforce schedule all employees have the same basic schedule but start with different offsets. Therefore, while the individual preferences of the employees cannot be taken into account, the aim is to find a schedule that is optimal (or fulfills all constraints) for all employees. The schedule presented in Table 1 would be a cyclic schedule, if after the first week the first employee takes the schedule of the second employee, the second employee takes the schedule of the third, . . . , employee 9 takes the schedule of employee 1. In non-cyclic workforce schedules the individual preferences of the employees can be taken into consideration and the aim is to achieve schedules that fulfill the preferences of most employees. In both variants of workforce schedules many constraints such as the minimum number of employees required for each shift have to be met. For this reason, they are generally difficult to solve, which corresponds to the extremely large search space and conflicting constraints. In this paper we will consider the problem of rotating workforce scheduling. According to [29] this problem is an NP-complete problem.

Workforce schedules have a impact on the health and satisfaction of employees as well as on their performance at work. Therefore, computerized workforce scheduling has interested researchers for over 30 years. The critical features of workforce scheduling algorithms are their computational behavior and flexibility for solving a wide range of problems that appear in practice. For solving the problem of workforce scheduling, different approaches have been used in the literature. Examples of the use of exhaustive enumeration are [24] and [9]. Balakrishnan and Wong [7] solved a problem of rotating workforce scheduling by modeling it as a network flow problem. Laporte [26] considered developing the rotating workforce schedules by hand and showed how the constraints can be relaxed to get acceptable schedules. The use of constraint logic programming has been shown in [11]. Recently, Laporte and Pesant [28] have proposed constraint programming algorithm for solving rotating workforce scheduling problems. A simple genetic algorithm for rotating workforce scheduling was proposed in [32].

Several other approaches for workforce schedules have been proposed in the literature for solving similar problems related to rotating workforce scheduling. Glover and McMillan [23] rely on integrating techniques from management sciences and artificial intelligence to solve general shift scheduling problems. Constraint processing has been used for nurse scheduling in [6]. Examples of the use of tabu search for nurse scheduling and related workforce scheduling problems are [15], [34], [16], [12], [38], and [19]. The combination of tabu search with other approaches for nurse scheduling has been also considered in the literature. For example, in [18], tabu search is used in combination with evolutionary algorithms, and combining tabu search with integer programming models has been proposed in [14]. Other metaheuristic approaches, such as genetic algorithms, have been used successfully in solving different workforce scheduling problems. Aickelin and Dowsland [2] proposed an indirect genetic algorithm for nurse scheduling. The proposed algorithm uses individuals who do not represent direct encoding of solutions, and includes a decoder heuristic to obtain the solutions from these individuals. Authors report better results than those reported by applying tabu search [15]. Other variants of genetic algorithms for nurse scheduling are presented in [36] and [3]. The application of Bayesian optimization and classifier systems in nurse scheduling is presented in [30]. In [10] is investigated relaxation of coverage constraints for nurse scheduling to generate better schedules. Recently, in [13], the genetic algorithm is used to solve the workforce scheduling problem which includes the resource constrained scheduling problem as a sub-problem. The authors investigate the use of the multi-objective algorithm and the genetic algorithm which uses weighted sum objectives. They show that, with multi-objective algorithm and without information for the weights of the objectives, solutions can be obtained whose fitness is within 2% of the fitness of solution obtained by the genetic algorithm with weighted sum objectives. Based on these results and knowing that in practice it is very hard to determine the best weights for the objectives, the use of the multi-objective approach is a good alternative in situations when the weights of objectives is hard to determine. Surveys of other algorithms used for workforce scheduling problems are given in [8], [5], [4].

In [33] there is proposed and implemented a method for the generation of rotating workforce schedules, which is based on pruning of search space, by involving the decision maker during the generation of partial solutions. The algorithms have been included in a commercial product called First Class Scheduler (FCS) [21], which is part of a shift scheduling package called Shift-Plan-Assistant (SPA) of XIMES[1] Corp. The main feature of FCS is the possibility to generate high quality schedules through the interaction with the human decision maker. Apart from the fact that the generated schedules meet all hard constraints, it also allows the incorporation of preferences of the human decision maker regarding soft constraints that are more difficult to assess and to model otherwise. Although this package has been shown to work well in practice for solving a broad range of real life problems, for very large instances feasible solutions cannot always be guaranteed, because of the size of the search space.

In this paper we propose methods with the aim of more effective generation of solutions for large instances of rotating workforce schedules. The generated rotating workforce schedules should satisfy all given hard constraints, i.e. no soft constraints or approximate solutions should be accepted (see section II). Of particular interest is the development of techniques for this problem which can solve problem instances which can not be solved by FCS in a reasonable amount of time. The main contributions of this paper are:

- We propose a tabu search based method for solving the rotating workforce scheduling problem.
- A new method based on the min-conflicts heuristic for solving this problem is proposed.
- We propose a combination of the min-conflicts based heuristic and tabu search. A new method which includes the so called tabu mechanism in the min- conflicts based heuristic for solving rotating workforce scheduling problem is proposed, and using the min-conflicts heuristic in the tabu search is considered. Additionally, we consider the introduction of random noise and random walk in the adopted random restart min-conflicts heuristic and tabu search method.

[1]http://www.ximes.com/

- We introduce a set of 17 real life problems, which are collected from different areas in industry. The proposed methods have been implemented and empirically evaluated using problems which have appeared in the literature and the set of introduced real life problems. Additionally, we compare methods proposed in this paper with the commercial system for generation of rotating workforce schedules.

We proceed in this paper as follows: In the next section we give a precise definition of the rotating workforce scheduling problem. In Section III the tabu search algorithm is proposed. Section IV describes new proposed method based on min-conflicts heuristic. In Section V the combination of tabu search, min-conflicts heuristic and random walk is proposed. Test problems from literature and the 17 problems collected from real life situations are presented in Section VI. Computational results for real life workforce scheduling problems and the comparison of these results with other approaches in literature are given in Section VII. In the last section conclusions remarks are given.

## II. The rotating workforce scheduling problem

In this section, we describe the problem of assignment of shifts and days-off to employees in the case of rotating workforce schedules. This is a specific problem of a general workforce-scheduling problem. The definition is given below [33]:
**Instance:**

- Number of employees: $n$.
- Set $A$ of $m$ shifts (activities) : $a_1, a_2, \ldots, a_m$, where $a_m$ represents the special day-off "shift".
- $w$: length of schedule. The total length of a planning period is $n \times w$ because of the cyclic characteristics of the schedules.
- A cyclic schedule is represented by an $n \times w$ matrix $S \in A^{nw}$. Each element $s_{i,j}$ of matrix $S$ corresponds to one shift. Element $s_{i,j}$ shows on which shift employee $i$ works during day $j$ or whether the employee has time off. In a cyclic schedule, the schedule for one employee consists of a sequence of all rows of the matrix $S$. The last element of a row is adjacent to the first element of the next row, and the last element of the matrix is adjacent to its first element.
- Temporal requirements: The requirement matrix R $((m-1) \times w)$ (we use here $m-1$, because shift $a_m$ represents the day-off), where each element $r_{i,j}$ of the requirement matrix $R$ shows the required number of employees for shift $i$ during day $j$.
- Constraints:
  - Sequences of shifts not permitted to be assigned to employees. For example, one such sequence is $ND$ (Night Day): after working in the night shift, it is not allowed to work the next day in the day shift. The typical problem in rotating workforce scheduling does not allow several shift sequences. More information for the not allowed sequences for problems solved in this paper are given in section VI .
  - Maximum and minimum length of periods of consecutive shifts: Vectors $MAXS_m$, $MINS_m$, where each element shows the maximum respectively minimum permitted length of periods of consecutive shifts. Because the schedule is cyclic these constraints for the length of blocks of shift are identical for all employees and the shift sequences can lie in more than one week. For example, suppose that it is not allowed to work more than $4$ Night (N) shifts in a sequence. The employee 3 (in Table 1) works from Thursday to Friday in the night shift. If the employee $4$ would work on Monday in the Night shift this constraint would be violated, because after finishing his/her schedule, employee 3 would take the schedule of employee $4$ and thus work $5$ Night shifts in a sequence, which is not allowed.
  - Maximum and minimum length of blocks of workdays: $MAXW$, $MINW$. A Work block is a sequence consisting only from the working shifts (without day-off in beetwen). This constraint limits the number of days in which the employees can work without having a day off. Cyclicity should also be taken into consideration for the work blocks. For example, the work block of employee 2 in Table 1 is of length 7 (D D D D D D D), because he will work from the Tuesday of the current week to the Monday of the next week in the Day shift (employee 2 take the schedule of employee 3 for the next week).

**Problem:** Find a cyclic schedule (assignment of shifts to employees) that satisfies the requirement matrix, and all other constraints.
Note that in [33], finding as many non-isomorphic cyclic schedules as possible that satisfy all constraints, and are optimal in terms of weekends without scheduled work shifts (weekends off), are required. We consider in this paper the generation of only one schedule, which satisfies all the hard constraints given in the problem definition. Fulfilling of all given constraints for this problem in practice are usually sufficient. The same constraints that we use in this paper are used in the commercial software (FCS) for generation of rotating workforce schedules. Although this problem is a specific formulation of the workforce scheduling problem, this system has been used since 2000 in practice for many com-

panies in Europe. In literature and practice other different formulations of workforce scheduling problems also appear, which additionally include the soft constraints. Such problems include for example nurse scheduling, where soft constraints, like individual preferences of employees are of high importance.

## III. Tabu search for rotating workforce scheduling

Tabu search [22] is a powerful modern meta-heuristic technique, which has been used successfully for many practical problems. This technique belongs to a class of local search techniques. Local search strategies [1], [35] have been successfully applied to constraint satisfaction problems and other hard combinatorial optimization problems. The basic idea of local search techniques is to improve initial solutions iteratively during the search. One basic local search technique is hill-climbing, which starts with a randomly generated solution and moves during each iteration to the solution in the neighborhood with the best objective function value. This technique can easily get stuck in local optima. Different approaches have been used in the literature to escape from a local optimum during the search. Such examples includes simulated annealing [25] and tabu search [22].

The basic idea of tabu search is to avoid cycles (visiting the same solution) during the search by using the tabu list. In the tabu list specific information about the search history for a fixed specific number of past iterations are stored. The acceptance of the solutions for the next iterations in this technique depends not only on its quality, but also on the information about the history of the search. In the tabu list one stores for instance the moves (specific changes of solution, see Section III-B) or inverse moves that have been used during a specific number of past iterations. The stored moves are made tabu for several iterations. A solution is classified as a tabu solution if it is generated from a move that is in the tabu list. In this technique, a complete neighborhood (with defined moves) of the current solution is generated during each iteration. In the basic variant of TS the best solution (not tabu) from the neighborhood is accepted for the next generation. However, it is also possible to accept the tabu solution if it fulfills some conditions, which are determined by the so called aspiration criteria (i.e., the solution is tabu but has the best objective function value so far).

Although tabu search has been used to solve similar problems with rotating workforce scheduling [15], [34], [16], [12], [38], [19], to the best of our knowledge tabu search has not been used to directly solve the problem we consider in this paper. Rotating workforce scheduling involves typically different constraints compared to nurse scheduling, and what also makes the rotating workforce scheduling different from nurse scheduling is that the schedule is cyclic. During the generation of a solution for rotating workforce scheduling, one should also consider fulfilling constraints which appear because of the connection of the schedule of each employee with the schedule of the next employee.

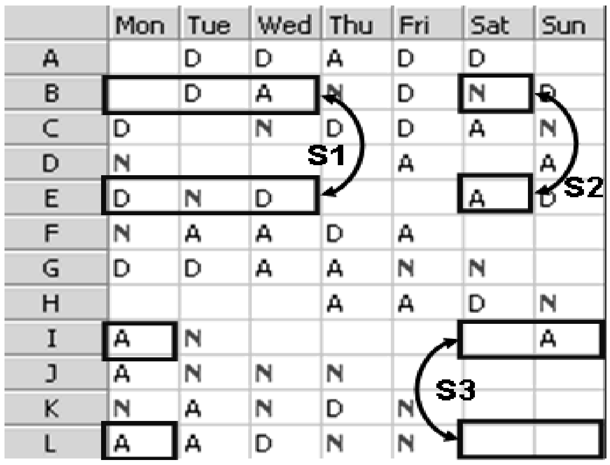
The appropriate neighborhood structure for tabu search is very important to reach ‘good’ solutions. In this section, we first describe the neighborhood structure for the problem we solve in this paper. We propose such a neighborhood structure based on the specifics of a problem, so that the search can be more effective. Afterwards, we propose the structure of memory (tabu list) during the search, acceptance of the descendant solutions and aspiration criteria. Furthermore, the appropriate fitness function which guides the search towards the ’good’ solutions for this problem is presented.

### *A. Generation of neighborhood*

As described in Section II, the cyclic schedule is represented by an $n \times w$ matrix $S$, where each element $s_{i,j}$ of matrix $S$ corresponds to one shift or day off. To generate the neighborhood of the current solution we apply a move, which swaps the contents of two elements in the matrix $S$. However, to reduce the neighborhood of the current solution, we only allow those solutions which fulfill workforce requirements, and thus the swapping of elements is done only inside of a particular column. The whole neighborhood of the current solution is obtained by swapping all possible elements (not identical) in all columns of a table. The applied move is denoted as $swap(j, i, k)$, where $j$ represents the column in which the swap of elements should be done, and $i$, $k$ represent elements in column $j$ that should swap their contents. The neighborhood of the current solution represents all solutions that can be obtained by applying this move($swap(j, i, k)$) to the current solution. Additionally, the above described move was extended to swap the blocks of more neighborhood cells (the last cell in the schedule of an employee is a neighbor of the first cell of the schedule for the next employee). This move is denoted $swapBlock(j, i, k, length)$, where $j, i, k$ are the same parameters as for the move $swap(j, i, k)$, whereas $length$ represents the length of a block of cells which have to be swapped. Similar neighborhoods have been used previously in the literature. For similar neighborhoods and other neighborhood relations used for nurse scheduling the reader is refereed to [34], [15], [17], [19].

*Example 2:* In Figure 1 three swaps of cells are illustrated. The first swap denoted **S1** represents the move $swapBlock(1, 2, 5, 3)$. With this move the contents of two blocks are swapped. The blocks begin in column 1 and in rows 2 and 5, correspondingly. The length of the block which in this case is 3 is determined by the last parameter in the $swapBlock$ move. The second swap **S2** swaps the contents of the single cells and represents the move swap(6,2,5). The last swap **S3** represents this move: $swapBlock(6, 9, 12, 3)$: The block of length three which begins in column 6 and row 9 should be swapped with the block of length 3 which begins in column 6 and row 12. Note, that because the blocks begin in column 6 and their length is 3, and knowing that the length of schedule is 7, the next cell to be sawpped after the last cell

$s(9,7)$ is the cell $s(9,1)$ (for the second block the next neighborhood cell of $s(12,7)$ is the cell $s(12,1)$. In general for the cell $s(i,j)$ of the schedule the neighborhood cell is the cell $s(i,j+1)$. In case the cell is in the last column ($j = 7$ for the week schedule) the next neighborhood cell is $s(i,1)$ because of rotation.



| | Mon | Tue | Wed | Thu | Fri | Sat | Sun |
|---|---|---|---|---|---|---|---|
| A | | D | D | A | D | D | |
| B | | D | A | N | D | N | D |
| C | D | | N | D | D | A | N |
| D | N | | | | A | | A |
| E | D | N | D | | | A | D |
| F | N | A | A | D | A | | |
| G | D | D | A | A | N | N | |
| H | | | | A | A | D | N |
| I | A | N | | | | | A |
| J | A | N | N | N | | | |
| K | N | A | N | D | N | | |
| L | A | A | D | N | N | | |

**Figure. 1**: Illustration of swap move

Based on the particular techniques proposed in this paper, three types of neighborhoods are generated. In Tabu Search (TS) the whole neighborhood is generated. This neighborhood is obtained by swapping all possible elements (not identical) in all columns of a table. In the case of Min-Conflicts (MC) heuristics the neighborhood of current solution is generated by swapping of the content of randomly selected cell which appears in some violated constraint (conflict) and other cells in the same column. Further, for the random walk strategy the neighborhood of the solution is generated by swapping of the content of a randomly chosen cell which appears in some conflict with the content of another cell which is selected randomly in the same column. For three techniques the same applies in the case of swapping blocks of cells.

## B. Tabu mechanism and selection criteria

In order to avoid cycles during the search, the search history is used. Specific information for the search history is stored in the tabu list. In our case in the tabu list are stored the moves which should not be applied for several iterations during the search. For example, if the solution accepted for the next iteration is obtained by swapping the contents of cells 4 and 5 in the first column of the schedule ($swap(1,4,5)$), in the tabu list is stored the moves: $swap(1,4,5)$. This move is made tabu for several iterations depending on the length of the tabu list. This should avoid cycles during the search.
For finding the most appropriate tabu length for tabu search approach we experimented with different lengths of tabu list. In the first variant we experimented with the same length of tabu list for all instances, without taking into consideration the size of the problems. In the second variant which proved to be better the length of the tabu list is dependent on the size of the problem (number of employees in the schedule). We experimented with these lengths of tabu list: 2*NumberOfEmployees, 4*NumberOfEmployees, 6*NumberOfEmployees, ...,24*NumberOfEmployees. For each tabu length we experimented with two types of moves defined earlier. In the first variant only $swap(i,j,k)$ move was applied, whereas in the second variant also the $swapBlock(i,j,k,length)$ move was applied to become the neighborhood solution during each iteration.
The following criteria is applied to determine the acceptance of a solution for the next iteration. The best solution from the neighborhood, if it is not tabu, becomes the current solution in the next iteration. If the best solution from the neighborhood is tabu, then the aspiration criteria is applied. For the aspiration criterion, we use a standard version [22] according to which the tabu status of a move is ignored if the move has a cost better than the current best solution. Different other aspiration criteria and tabu lists have been used in the literature for nurse scheduling [34], [15].

## C. Fitness Function

During the generation of a neighborhood, the evaluation of solutions is very time consuming, because each solution has to be checked for many constraints. To calculate the fitness of the solution, for each violation of a constraint, a determined number of points (penalty) is given, based on the constraint and the degree of the constraint violation. The fitness represents the sum of all penalties caused by the violation of the constraints. Since the problem we want to solve only has hard constraints, the solution will be found when the fitness of the solution reaches the value 0. The fitness is calculated as follows:

$$\begin{aligned} \mathit{Fitness} = & \sum_{i=1}^{NW} P1 \times \mathit{Distance}(\mathit{WB}_i, \mathit{WorkBRange}) + \\ & \sum_{i=1}^{ND} P2 \times \mathit{Distance}(\mathit{DOB}_i, \mathit{DayOffBRange}) + \\ & \sum_{j=1}^{NumOfShifts} (\sum_{i=1}^{NS_j} P3 \times \mathit{Distance}(\mathit{SB}_{ij}, \mathit{ShiftSeqRange}_j)) + \\ & P4 \times \mathit{NumOfNotAllowedShiftSeq} \end{aligned}$$

$NW$, $ND$, represent, the number of work blocks, and days off blocks, respectively. $NS_j$, represents the number of shift sequences (blocks) of shift $j$. $WB_i$, $DOB_i$, represent, a work block $i$, and a days-off block $i$, respectively. $SB_{ij}$, represents the $i$-th shifts block of shift $j$. The penalties of the violation of constraints are set as follows: $P1 = P2 = P3 = P4 = 1$. The function $Distance(XBlock, range)$ returns

$0$ if the length of the block $XBlock$ is inside the range of two numbers ($range$), otherwise it returns the distance of the length of $XBlock$ from the range. For example, if the legal range of work blocks is $4-7$ and the length of work block $XBlock$ is 3 or 8 then this function will return value 1.

*Example 1*: We have given the problem with $5$ employees and one shift: Day shift (D). For the given shift from Monday to Saturday 4 employees are needed, whereas on Sunday no employees are needed. A rotating week schedule has to be constructed which fulfills the following constraints:

- Length of work blocks should be between 4 and 6 days ($WorkBRange : 4-6$)
- Length of days-off blocks should be between 2 and 4 ($DayOffBRange : 2-4$)
- Length of periods of successive shifts for shift D should be between 4 and 6 ($ShiftSeqRange_1 : 4-6$)

Suppose that during the search the current solution for this problem is the solution given in Table 2. The given solution contains $5$ work blocks $WB1, ...WB5$ with legth $6, 4, 6, 6, 2$, respectively. The length of days off blocks $DOB1, ..., DOB5$ are $3, 1, 1, 1, 5$, respectively. Finally, the length of shift sequences for shift D are $6, 4, 6, 6, 2$, respectively. The components of the fitness function are:

$$\sum_{i=1}^{5} P1 \times Distance(WB_i, WorkBRange) = P1 \times 0 + P1 \times 0 + P1 \times 0 + P1 \times 0 + P1 \times 2$$

$$\sum_{i=1}^{5} P2 \times Distance(DOB_i, DayOffBRange) = P2 \times 0 + P2 \times 1 + P2 \times 1 + P2 \times 1 + P2 \times 1$$

$$\sum_{j=1}^{1}(\sum_{i=1}^{NS_j} P3 \times Distance(SB_{ij}, ShiftSeqRange_j)) = P3 \times 0 + P3 \times 0 + P3 \times 0 + P3 \times 0 + P3 \times 2$$

$$P4 \times NumOfNotAllowedShiftSeq = 0$$

For $P1 = P2 = P3 = P4 = 1$ the fitness of the given solution is: $Fitness = 8$.

*Table 2*: Current solution for Example 1

| Emp./day | Mon | Tue | Wed | Thu | Fri | Sat | Sun |
|---|---|---|---|---|---|---|---|
| 1 | D | D | D | D | D | D | |
| 2 | | | D | D | D | D | |
| 3 | D | D | D | D | D | D | |
| 4 | D | D | D | D | D | D | |
| 5 | D | D | | | | | |

### D. Generation of initial solution

The initial solution is generated randomly considering the positions of shifts inside of each column. However, the initial solution fulfills all the workforce requirements. In each column of the schedule a determined number of shifts (determined by workforce requirements) are distributed randomly. The neighborhood operator applied to the current solution generates a solution which does not violate the workforce requirements.

### E. The overall tabu search algorithm

The pseudo code of the tabu search algorithm is presented in Algorithm 1.

**Algorithm 1** Tabu search algorithm (TS)

```
Generate random initial solution

Initialize tabu list

while termination-condition not true do
    Generate a whole neighborhood of current solution as defined in Section III-A

    Evaluate neighborhood solutions based on fitness function defined in Section III-C

    Select the solution for the next iteration based on selection criteria defined in Section III-B

    Update tabu list

end while
```

The termination condition is fulfilled if the solution which fulfills all constraints is found, or if a determined number of evaluations of solutions is reached. The limit for the number of evaluations for each method proposed in this paper is set to be 10 million (see Section VII).

## IV. Min-conflicts based heuristic for rotating workforce scheduling

The min conflicts heuristic [31] has been used successfully for solving constraint satisfaction problems. In this technique during each iteration a conflicted variable is selected randomly. For the selected variable the new value is picked such that the number of conflicts is minimized. The pure min conflicts technique can also get stuck in a local optimum. To escape the local optimum the algorithm can be restarted or noise strategies such as random walk [37] can be introduced. Based on the ideas of the min-conflicts heuristic, in this paper we propose a new method for the generation of rotating workforce schedules. Due to the specifics of the problem we consider here, the proposed method has several differences compared to the pure min-conflicts heuristic. Further, we present the adopted random restart min conflicts heuristic (MC). By using random restart the min-conflicts heuristic is started for a determined number of times from different initial solutions. The pseudo code of random restart min conflicts (MC) heuristics for rotating workforce scheduling is given in Algorithm 2.

**Algorithm 2** Random restart min-conflicts heuristic (MC)

```
while ($NumOfRestart < MaxRestart$) do
  Generate random initial solution

  while termination-condition do
    Select randomly a cell of schedule among the cells
    which appear in violated constraints

    Generate neighborhood of current solution by swap-
    ping the content of selected cell and the other cells in
    the same column (or by swapping the block of cells)

    From the generated neighborhood select for the next
    iteration a solution which minimizes the number
    of conflicts (this corresponds to the solution which
    mimimizes fitness defined in Section III-C)

  end while
end while
```

Considering the generation of a neighborhood for the current solution in the simplest case the neighborhood is generated by swapping the content of the cell which is randomly selected from cells which appear in some violated constraint with the content of other cells appearing in the same column. The swap is limited to being between cells which appear in the same column, because with this limitation it is assured that solutions always fulfill the workforce requirements. We experimented also with swapping the block of cells. The first cell in a block to be swapped is the cell which is selected randomly from the cells which appear in violated constraints. The next elements of the block are the neighborhood cells of the randomly selected cell. We experimented with these lengths of blocks: 1, 2, 3, 4. Note that the workforce requirements are also always fulfilled when blocks of cells are swapped.

The main difference of this technique compared to the previous proposed Tabu Search (TS) algorithm is in the generation of the neighborhood of the current solution. In the min-conflicts (MC) heuristic only part of the neighborhood is generated during each iteration. The part of the neighborhood to be explored is determined based on the information for the cells of the schedule which appears in conflicts (not fulfilled constraints). Only the neighborhood which can be generated by swapping in a column in which one of these cells is located is considered in MC during each iteration. This makes the search much more effective, especially for larger instances, in which the whole neighborhood of the current solution is very large. To have an idea of the number of solutions to be explored by TS and MC during each iteration, suppose that we have a problem with $9$ employees and three shifts. Suppose that the requirements for each day are to have $2$ employees in each shift. If only the move for swapping single cells is used the number of solutions explored by TS during each iteration will be 210. This number is obtained in the following way. In each column, the first cell (cell in the first row) can be swapped with $8$ other cells in that column, second cell with 7 other cells and so on, the cell in row 8 can be swapped with only the last cell in the column. The number of total swaps in one column is $8 + 7 + \ldots + 1 = 36$. As $6$ of these swaps are between the cells with the same content, the total number of swaps per column remains $30$. In the schedule with the given requirements and 7 columns there exist 210 swaps. The number of solutions explored by MC will be maximally 7 (swaps of one cell in one of the columns with all other cells in that column minus 1, because at least one swap will be between cells with the same shift). Another difference of the MC heuristic compared to TS concerns the acceptance of a solution for the next iteration. In the proposed MC heuristic the tabu search principles are not used. The combination of the MC technique and tabu mechanism is subject of a subsequent part.

MC heuristic for rotating workforce scheduling has these differences compared to the pure random restart min conflicts heuristic [31] due to the specifics of this problem: In our case not only one variable (cell) in conflict changes the value, here it is possible that more than one variable change their contents. In the case of a simple neighborhood two variables will change their contents, whereas in the case of a swap of block of length 4, their contents will change 8 variables. In our procedure also a worse solution can be accepted for the next iteration, whereas in pure min conflicts this does not happen. The basic idea of minimizing the number of conflicts remains, but we evaluate solutions not only based on the number of conflicts, but also based on the degree of constraint violation (see section III-C ). A similar approach for calculating fitness based on penalty functions is proposed by Galinier and Hao [20].

As for tabu search the termination condition will be met if the solution which fulfills all constraints is found or if the determined number of evaluations of solution are reached. Note that in computational experiments for the min-conflicts heuristic the number of restarts is set to be 10. The maximal number of evaluations per random restart is 1 mil. evaluations.

## V. Combining min-conflicts heuristic, tabu search and random walk

In this section we propose a combination of the techniques proposed in the previous section and consider also including the random walk strategy in the proposed methods. We propose a combination of tabu search with min conflicts heuristics and random walk, using random walk in the min conflicts heuristics and introducing the tabu mechanism in the min-conflicts heuristic. Furthermore, we consider combining min-conflicts, tabu search and the random walk strategy. By combining the tabu search (TS) with the min-conflicts (MC) heuristic and random walk, the basic idea is not to explore the whole neighborhood during each iteration as in classical tabu search, but with some probability to apply either the random walk or min conflicts heuristic in each iteration. This makes possible, an exploration of different regions of the search space or diversification of the search and also reduces neighborhoods to be explored during each iteration. Note that the random walk strategy has been modified according to the specifics of the problem we consider here. The random walk based strategy in our case picks randomly one cell which appears in some violated constraint (conflict). The content of this cell is swapped with the content of another cell which is selected randomly in the same column. The reason why we allow the swapping of the contents of cells only inside of one column as described earlier is to assure that the solutions during each iteration always fulfill the workforce requirements. We have also experimented with another variant in which the two cells are picked randomly, without the restriction that the first cell appears in some conflict (random noise strategy). The combination of tabu search (TS) with a random walk is described in Algorithm 3.

**Algorithm 3** Tabu search and random walk (TS-RW)

```
For Each Iteration

With probability $p$:
pick randomly the cell which is in conflict. Pick in the same column another cell randomly and swap the contents of the two selected cells

With probability $1 - p$:
follow the tabu search (TS) algorithm
```

Combination of tabu search (TS) with the min conflicts (MC) strategy described in Section IV is given with the pseudo code in Algorithm 4:

**Algorithm 4** Tabu search and min-conflicts heuristic (TS-MC)

```
For Each Iteration

With probability $p$:
apply min conflicts (MC) strategy

With probability $1 - p$:
follow the tabu search (TS) algorithm
```

As we can see from the procedures, during each iteration the standard tabu search (TS) approach proposed in section III will be applied with some probability $1 - p$, whereas with probability $p$ the random walk or min-conflicts (MC) strategies are applied. The best values for the parameter $p$ are determined experimentally. We experimented with different probabilities for the random walk: $0.05, 0.1, \ldots, 0.4$. Considering the probability for min-conflicts (MC), we experimented with these values for $p$: $0.05, 0.1, \ldots, 0.7$.
Further, we considered the combination of the MC with the random walk and random noise. The combination of the MC with the random walk is described by the pseudo code in Algorithm 5.

**Algorithm 5** Min-conflicts heuristic and random walk (MC-RW)

```
while ($NumOfRestart < MaxRestart$) do
  Generate random initial solution

  while (termination-condition) do
    With probability $p$:
    pick randomly the cell which is in conflict. Pick in the same column another cell randomly and swap the contents of two selected cells

    With probability $1 - p$:
    follow the min conflicts (MC) based procedure
  end while
end while
```

The procedure in which random noise is introduced in the MC heuristic is the same as when the random walk is introduced, except that in this case with probability $p$ both cells to be swapped are selected randomly, i.e. the column and the rows of cells to be swapped are selected randomly. We experimented with different values for the probability $p$ : $0.02, 0.05, 0.1, 0.15$.

### *A. Introducing tabu mechanism in min conflicts based heuristic*

We propose an extension of the min conflicts (MC) based heuristic described in Section IV by introducing the tabu mechanism [22] in this heuristic. The basic idea is to store information about the swap of cells during each iteration as for the tabu search technique. The information about the history of swaps is then used in the selection process of a solution for the next iteration. We proceed as follows. Each swap used for obtaining the solution for the next iteration is stored in the tabu list. For example, if the solution accepted for the next iteration is obtained by swapping the contents of cells 4 and 5 in the first column (move $swap(1, 4, 5)$) of schedule, in the tabu list is stored the moves $swap(1, 4, 5)$. The information for swaps is kept in the list only for a determined number of further iterations. Note that during the swapping of blocks of cells also the information for the length of blocks is stored in the history of swaps. During the selection of a solution from the neighborhood for the next iteration the solution obtained from the swaps stored in the tabu list are not taken into consideration for selection. An exception is made if the solution has the best objective function value so far (aspiration criteria [22]). The pseudo code of the procedure which includes the tabu mechanism in a min conflicts based heuristics is given in Algorithm 6.

**Algorithm 6** Min-conflicts heuristic with tabu list (MC-T)

```
while ($NumOfRestart < MaxRestart$) do
  Generate random initial solution

  while (termination-condition) do
    Select randomly a cell from the cells which appear in violated constraints

    Generate a neighborhood of the current solution by swapping the content of the selected cell and other cells in the same column (or by swapping of blocks of cells)

    Eliminate from the neighborhood the solutions which are obtained by swaps contained in the tabu list and which do not have the best fitness so far

    From the remaining solutions in the neighborhood select the best solution which minimizes the number of conflicts

    Add to the tabu list the swap with which the selected solution was obtained, and remove from the tabu list the oldest swap

  end while
end while
```

The main difference of this algorithm compared to MC heuristics is that in this algorithm not always the solution which is the best according minimizing the number of conflicts is selected for the next iteration. The solution will be accepted for the next iteration only if it is obtained by moves which are not stored in the tabu list during the past determined number of iterations. This should help to avoid cycles or repetitions of the same solutions during the search. The only case in which a solution which is obtained by the forbidden swaps is accepted is the case when the solution has the best fitness so far.
When including a tabu mechanism it is important to have the appropriate value for the length of the tabu list. Very short tabu lists may not have enough effects to avoid repetition during the search, and too long tabu lists may forbid sometimes the solutions which could lead to the better solutions in the search space. In our case we determined this parameter based on empirical results. We use tabu lists, such that the length of the tabu list is dependent on the size of the problem. By size of the problem we mean the number of employees (groups) for which the schedule should be generated. We experimented with these lengths of tabu list: $0.5 * n, 1 * n, 4 * n, 78 * n, 10 * n$, where $n$ represents number of employees.
The described procedure includes the tabu mechanism in the random restart min-conflicts based heuristic. We also experimented with introducing both the tabu mechanism and the random walk in the random restart min conflicts (MC) based heuristic. The pseudo code of this method is presented in Algorithm 7:

**Algorithm 7** Min-conflicts heuristic with tabu list and random walk (MC-T-RW)

```
For Each Iteration
With probability $p$:
pick randomly the cell which is in conflict. Pick in the same column another cell randomly and swap the contents of the two selected cells

With probability $1 - p$
follow the min-conflicts heuristic with the tabu mechanism (MC-T)
```

The only difference of this procedure with the previous procedure is that with a determined probability $p$ during each iteration a random walk strategy will be applied, instead of using the min-conflicts heuristic with the tabu list (MC-T).

## VI. Test problems

In this paper we report the results for 20 problems (Example 1-20). Example 1-3 appeared earlier in the literature and they are described later in this section. In this paper we present for the first time the collection of 17 (corresponds to Examples 4-20) real life problems which has been created from shifts and temporal requirements which appeared in a broad range of organizations, like airports, factories, health care organizations, etc. The collection of these problems can be downloaded from http://www.dbai.tuwien.ac.at/staff/musliu/benchmarks/.
Note that from the given temporal requirements and shifts different size of problems can be created based on the average number of hours per employee per week. For most examples the number of employees and constraints are taken as they are usually proposed by consultants for the given workforce requirements and shifts. To test algorithms, for a few instances we have chosen the average number of hours such that the number of groups for these instances gets very large (instances with more than 48 groups). However, also

for these instances the temporal requirements and shifts are from real life situations. These 17 problems have these features:

- The number of employees (or groups) for these problems are from 7 to 163. The collection contains small, middle-size and very large instances considering number of employees.
- The number of shifts for 15 examples is 3, whereas for 2 examples, the number is 2. Problems include standard shifts: D (day), A (Afternoon) and N (Night) shift.
- Each example includes constraints about not allowed shift sequences, length of shift sequences, and length of work and days-off blocks.

In Table 3 the features of instances 4-20 are presented in detail. The second column represents the number of groups (or employees) for each problem. The number of shifts is presented in the third column. In the fourth column are presented not allowed sequences of shifts. Considering not allowed sequences there are two types of sequences. Critical sequences (named seq1): "‘N D"’ (the employee is not allowed to work on a day shift after a night shift), "‘N A"’ and "‘A D"’. Second type of not allowed sequences named seq2 are: "‘N - D"’ (the employee is not allowed to work on a night shift then have a free and then work on a day shift), "‘N - A"’, "‘A - D"’, "‘N - N"’ and they are used only if the single day off is allowed. Column named "‘Seq. length"’ represents the length of sequences for each shift (in order of shifts D, A and N). The column named WBL represents the possible length of work blocks and the last column the allowed length of days-off blocks. Workforce requirements for each example are not presented here and the reader is refereed to http://www.dbai.tuwien.ac.at/staff/musliu/benchmarks/ for the specific temporal requirements for each problem.

Examples 1-3 appeared previously in the literature and are included in a collection of problems for which we give the computational results here. These three problems are described in detail below.

**Example 1**: The first problem from the literature is a small problem solved by Butler [9] for the Edmonton police department in Alberta, Canada. Characteristics of this problem are:
Number of employees: 9
Shifts: 1 (Day), 2 (Evening), 3 (Night)

Temporal requirements:

$$R_{3,7} = \begin{pmatrix} 2 & 2 & 2 & 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 3 & 3 & 3 & 2 \\ 2 & 2 & 2 & 2 & 2 & 2 & 2 \end{pmatrix}$$

Constraints: (1) The length of work periods should be between 4 and 7 days; (2) Only shift 1 can precede the rest period preceding a shift 3 work period; (3) Before and after weekends off, only shift 3 or shift 2 work periods are allowed; (4) At least two consecutive days must be assigned to the same shift; (5) No more than two 7-day work periods are allowed and these work periods should not be consecutive.

Let us note here that in all three examples given, we cannot model the problem exactly (the same is true for approaches [7] and [33] to the original problems), which is to a high degree due to the different legal requirements found in diffrent states, but we tried to mimic the constraints as closely as possible or to replace them by similar constraints that appeared more meaningfully in the European context. In this paper we consider the same constraints considered in [33].

Considering this problem constraints two and three cannot be represented in our framework. The other constraints can be applied in our model and are left as in the original problem. As mentioned, we include additional constraints about maximum length of successive shifts and maximum and minimum length of days-off blocks. In summary, additional constraints used for our solver and in [33] are: Not allowed shift changes: (N D), (N A), (A D); Length of days-off periods should be between 2 and 4; Vector $MAXS_3 = (7, 6, 4)$.

**Example 2**, Laporte et al. [27]: There exist three non overlapping shifts D, A, and N, 9 employees, and requirements are 2 employees in each shift and every day. A week schedule has to be constructed that fulfills these constraints: (1) Rest periods should be at least two days-off, (2) Work periods must be between 2 and 7 days long if work is done in shift D or A and between 4 and 7 if work is done in shift N, (3)Shift changes can occur only after a day-off, (4) Schedules should contain as many weekends as possible, (5) Weekends off should be distributed throughout the schedule as evenly as possible, (6) Long (short) work periods should be followed by long (short) rest periods, (7)Work periods of 7 days are

*Table 3*: Features of instances 4-16

| Ex | Gr | Shif. | Illegal seq. | Seq length | WBL | DOBL |
|---|---|---|---|---|---|---|
| 4 | 13 | 3 | seq1, seq2 | 2-6, 2-6, 2-4 | 3-7 | 1-4 |
| 5 | 11 | 3 | seq1, seq2 | 2-6, 2-5, 2-4 | 4-7 | 1-4 |
| 6 | 7 | 3 | seq1, seq2 | 2-6, 2-6, 2-6 | 4-7 | 1-4 |
| 7 | 29 | 3 | seq1 | 2-7, 2-6, 2-5 | 4-7 | 2-4 |
| 8 | 16 | 3 | seq1 | 2-7, 2-6, 2-5 | 3-7 | 2-4 |
| 9 | 47 | 3 | seq1 | 2-7, 2-7, 2-6 | 2-7 | 2-4 |
| 10 | 27 | 3 | seq1 | 2-7, 2-6, 2-5 | 4-7 | 2-4 |
| 11 | 30 | 3 | seq1 | 2-6, 2-5, 2-4 | 3-7 | 2-4 |
| 12 | 20 | 2 | seq1 | 2-6, 2-5 | 4-7 | 2-4 |
| 13 | 24 | 3 | seq1 | 2-6, 2-5, 1-4 | 3-7 | 2-4 |
| 14 | 13 | 3 | seq1, seq2 | 2-6, 2-5, 2-4 | 4-7 | 1-4 |
| 15 | 64 | 3 | seq1, seq2 | 2-6, 2-6, 2-5 | 3-6 | 1-4 |
| 16 | 29 | 3 | seq1 | 2-6, 2-5, 2-4 | 4-7 | 2-4 |
| 17 | 33 | 2 | seq1 | 2-6, 2-5 | 3-7 | 2-4 |
| 18 | 53 | 3 | seq1 | 2-7, 2-6, 2-5 | 4-7 | 2-4 |
| 19 | 120 | 3 | seq1 | 2-6, 2-5, 2-4 | 3-7 | 2-4 |
| 20 | 163 | 3 | seq1, seq2 | 2-6, 2-6, 2-5 | 3-6 | 1-4 |

preferred in shift N.
Constraint 1 is straightforward. Constraint 2 can be approximated if we take the minimum of work blocks to be 4. Constraint 3 can also be modeled if we take the minimum length of successive shifts to be 4. For the maximum length of successive shifts we take 7 for each shift. Other constraints can not be modeled in our solver.
**Example 3**: This problem is a larger problem first reported in [24]. Characteristics of this problem are:
Number of employees is 17 (length of planning period is 17 weeks).
Three nonoverlapping shifts.
Temporal requirements are:

$$R_{3,7} = \begin{pmatrix} 5 & 4 & 4 & 4 & 4 & 4 & 3 \\ 5 & 4 & 4 & 4 & 4 & 4 & 4 \\ 4 & 3 & 3 & 3 & 4 & 4 & 4 \end{pmatrix}$$

Constraints: (1) Rest-period lengths must be between 2 and 7 days; (2) Work-periods lengths must be between 3 and 8 days; (3) A shift cannot be assigned to more than 4 consecutive weeks in a row; (4) Shift changes are allowed only after a rest period that includes a Sunday or a Monday or both; (5) The only allowable shift changes are 1 to 3, 2 to 1, and 3 to 2.
We cannot model constraints 3, 4, and 5 in their original form. We allow changes in the same work block and for this reason we have other shift change constraints. In our case the following shift changes are not allowed: 2 to 1, 3 to 1, and 3 to 2. Additionally, we set the rest period length from 2 to 4 and the work periods length from 4 to 7. Maximum and minimum length of blocks of successive shifts are given with vectors $MAXS_3 = (7, 6, 5)$ and $MINS_3 = (2, 2, 2)$, repectively. The constraints for this problem are the same as constraints used in [33].

## VII. Computational results

In this section we report on computational results obtained with the current implementation of methods described in this paper. The results for 20 problems described in previous section are given. For comparison the number of evaluations needed to find the solution and the time for which the solution is found, are taken into consideration. The experiments were performed with a Pentium 4 machine, 1,8GHZ, 512 MB RAM. For each problem 10 independent runs with each algorithm are executed. The maximal number of evaluations for each run is 10 million evaluations.

### *A. Results*

In Table 4 (TS) a summary of results for the tabu search strategy for 20 problems is given. This table represents the best results obtained with the tabu search algorithm, and they are obtained with tabu length $10 * NumberOFEmployees$. Table 5 (TS-RW) presents results for tabu search and random walk. These are the best results, which are obtained with probability $p = 0.2$ for random walk. Further, in Table 6 (TS-MC) the results for tabu search and min conflicts are given. The best probability for min-conflicts was shown to be $0.5$. For each problem, the average number of evaluations and the average time needed (in seconds) for generation of feasible solution (solution which fulfills all constraints), are presented. Note that the time for which the solutions are found is usually very important, and especially in cases when in a short time consultants have to propose to the decision makers different solutions which need to be discussed. For calculation of averages only the successful runs are considered. The last column named 'solutions' represents the number of feasible solutions found in 10 runs.

*Table 4*: (TS) Tabu search algorithm (10 runs)

| Example | Number of Evaluat. | Time(sec) | Solutions |
|---|---|---|---|
| 1 | 66554.3 | 1.0 | 10 |
| 2 | 62631.3 | 1.1 | 10 |
| 3 | 454894 | 11.1 | 10 |
| 4 | 55568.3 | 1.3 | 10 |
| 5 | 150836 | 3.2 | 10 |
| 6 | 19953.9 | 0.3 | 10 |
| 7 | 6.12E+06 | 239.2 | 3 |
| 8 | 231230 | 5.4 | 10 |
| 9 | 4.24E+06 | 255.7 | 9 |
| 10 | 1.43E+06 | 52.8 | 10 |
| 11 | 4.82E+06 | 185.5 | 6 |
| 12 | 2.44E+06 | 61.4 | 8 |
| 13 | 435139 | 14.9 | 10 |
| 14 | 354999 | 8.1 | 10 |
| 15 | - | - | - |
| 16 | 1.08E+06 | 40.9 | 10 |
| 17 | 1.22E+06 | 48.5 | 10 |
| 18 | 5.88E+06 | 411.2 | 6 |
| 19 | - | - | - |
| 20 | - | - | - |

From tables 4, 5, 6 we can conclude that tabu search approach finds solutions in at least three runs (out of 10 runs) for each instance except for problems 15, 19, and 20 which are the largest problems in the collection of problems. As described in Section III the tabu search strategy explores whole neighborhood during each iteration, and thus it is hard for this strategy to find solutions (in 10 mil. evaluations) for very large instances as neighborhood during each iteration is very large. Tabu search approach is improved in combination with random walk considering both the average number of evaluations and also the number of runs in which solutions are found. However, also including the random walk heuristic does not help to solve problems 15,19 and 20. The combination of tabu search with the min-conflicts heuristic outperforms both tabu search and tabu search with random walk considering the time and number of evaluations for which the solutions are generated and the number of successful runs.

*Table 5*: (TS-RW) Tabu search and random walk (10 runs)

| Example | Number of Evaluat. | Time(sec) | Solutions |
|---|---|---|---|
| 1 | 64320.7 | 1.0 | 10 |
| 2 | 55209 | 0.9 | 10 |
| 3 | 296947 | 7.5 | 10 |
| 4 | 50817.7 | 1.2 | 10 |
| 5 | 216117 | 4.6 | 10 |
| 6 | 20553.8 | 0.3 | 10 |
| 7 | 1.74E+06 | 69.0 | 3 |
| 8 | 192726 | 4.5 | 10 |
| 9 | 2.22E+06 | 139.1 | 10 |
| 10 | 566757 | 22.1 | 10 |
| 11 | 2.66E+06 | 103.7 | 10 |
| 12 | 960118 | 24.7 | 10 |
| 13 | 223626 | 7.7 | 10 |
| 14 | 144895 | 3.3 | 10 |
| 15 | - | - | - |
| 16 | 793351 | 30.8 | 10 |
| 17 | 689217 | 28.6 | 10 |
| 18 | 4.49E+06 | 322.7 | 10 |
| 19 | - | - | - |
| 20 | - | - | - |

*Table 6*: (TS-MC) Tabu search and min-conflicts heuristic (10 runs)

| Example | Number of Evaluat. | Time(sec) | Solutions |
|---|---|---|---|
| 1 | 17846.1 | 0.3 | 10 |
| 2 | 16536.9 | 0.3 | 10 |
| 3 | 134973 | 3.4 | 10 |
| 4 | 21949 | 0.5 | 10 |
| 5 | 77054.9 | 1.6 | 10 |
| 6 | 18945.6 | 0.3 | 10 |
| 7 | 5.08E+06 | 196.1 | 9 |
| 8 | 98563.5 | 2.3 | 10 |
| 9 | 1.21E+06 | 79.5 | 10 |
| 10 | 300060 | 12.2 | 10 |
| 11 | 3.48E+06 | 134.2 | 10 |
| 12 | 721982 | 18.3 | 10 |
| 13 | 188700 | 6.4 | 10 |
| 14 | 98887.1 | 2.3 | 10 |
| 15 | 7.27E+06 | 738.0 | 5 |
| 16 | 262337 | 10.8 | 10 |
| 17 | 256918 | 10.9 | 10 |
| 18 | 2.29E+06 | 169.2 | 10 |
| 19 | 9.4E+06 | 1550.99 | 3 |
| 20 | - | - | - |

The only problem which can not be solved by this strategy is problem 20. The results in general show that the combination of tabu search with random walk and min-conflicts heuristic improves this technique in the case of rotating workforce scheduling problems.

In Table 7 (MC) a summary of results for the random restart min-conflicts based strategy is presented. For each run 10 random restarts are performed. Maximal number of evaluations per random restart is 1 mil. evaluations (for each run total 10 mil. evaluations). Further, in Table 8 (MC-RW) the best results for min-conflicts based strategy and random walk are presented (MC-RW). These results are obtained with probability 0.05 for the random walk. Results for min-conflicts and random noise (MC-RN) are similar to these results and are not presented here. MC-RN considering the number of solutions found in 10 runs, gives the same results as MC-RW and considering time for finding of solutions this technique needs on average slightly more time than MC-RW. The best results for the min-conflicts based strategy in which the tabu mechanism is included are given in Table 9 (MC-T). These results are obtained with a length of tabu list which is equal to the number of employees for each example. From Tables 7 (MC), 8 (MC-RW), 9 (MC-T) we can conclude that the ingredients given to random restart min-conflicts improves this technique for the rotating workforce scheduling problem. Introduction of random walk (and random noise) makes it possible to solve all the problems in each run. Finally, results show that introducing the tabu mechanism into the min-conflicts heuristics makes this technique more powerful for the problems we consider here. The results obtained by MC-T are slightly better considering the average time for solving of problems compared to introducing the random walk strategy.

We also experimented with introducing random noise and random walk into MC-T. However, the combination of MC-T and random walk (or random noise) does not give better results than MC-T alone and for this reason these results are not presented here.

In summary, based on the experimental results we can conclude that from the 6 methods presented in this paper the best results are obtained by the min-conflicts heuristic which includes tabu mechanism or random walk during the search.

Further we did experiments for three largest problems $15, 19, 20$, to investigate where the most of the computational time is used during the search. In figures 2, 3, and 4 are presented the changes of fitness during the time for TS, MC, and MC-TS, respectively. From the figures we can conclude that at the beginning of the search the fitness is decreased very fast, and the most of the time during the search is used to fulfill only few constraints which remain unfulfilled. In case of TS the fitness decreases slower, as during each iteration the whole neighborhood is explored. According to our experience with other examples, usually the fitness very near to $0$ is found very fast, and most of the time during the search is used to fulfill the few remained constraints.

In Tables 10 and 11 are presented solutions for the min-conflicts heuristic with the tabu mechanism for problem 2 and problem 4 (only one solutions from 10 runs). We do not present all the solutions here and the reader is refereed to http://www.dbai.tuwien.ac.at/staff/musliu/benchmarks/ for the solutions obtained with the MC-T strategy for all the

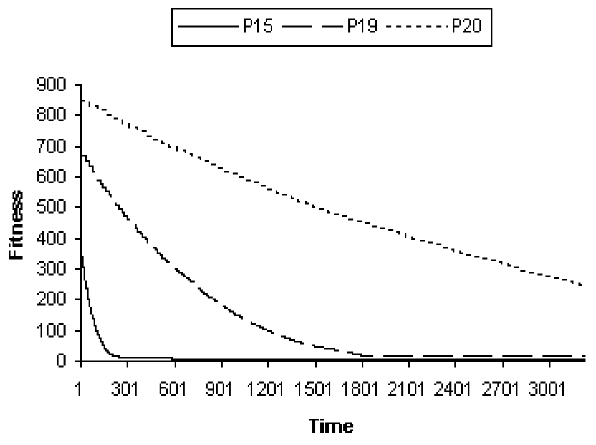


**Figure. 2**: Change of fitness over the time for problems 15,19,20 in case of TS

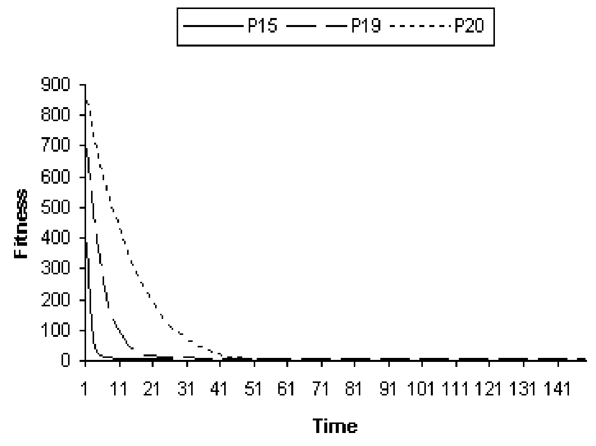


**Figure. 3**: Change of fitness over the time for problems 15,19,20 in case of MC

*Table 7*: (MC) Random restart min-conflicts based strategy (10 runs)

| Example | Number of Evaluat. | Time(sec) | Solutions |
|---|---|---|---|
| 1 | 306517 | 4.77 | 10 |
| 2 | 104486 | 1.48 | 10 |
| 3 | 2.82E+06 | 69.36 | 10 |
| 4 | 5303.3 | 0.12 | 10 |
| 5 | 820761 | 15.78 | 10 |
| 6 | 207406 | 2.89 | 10 |
| 7 | 1.72E+06 | 62.51 | 10 |
| 8 | 1.52E+06 | 32.52 | 10 |
| 9 | 1.52E+06 | 84.17 | 5 |
| 10 | 320288 | 11.40 | 10 |
| 11 | 7.05E+06 | 254.82 | 1 |
| 12 | 3.12E+06 | 74.26 | 10 |
| 13 | 2.23E+06 | 68.32 | 10 |
| 14 | 417234 | 8.77 | 10 |
| 15 | 3.65E+06 | 331.11 | 7 |
| 16 | 415611 | 14.48 | 10 |
| 17 | 1.51E+06 | 54.79 | 10 |
| 18 | 967663 | 60.58 | 10 |
| 19 | 4.28E+06 | 577.96 | 7 |
| 20 | 779768 | 183.82 | 10 |

problems presented in this paper.
The hybridization of techniques gives very good results as shown in tables 8 (MC-RW), and 9 (MC-T). The MC-T technique takes advantage of two techniques, TS and MC. Using the principles of the MC technique, the size of neighborhood to be explored in each iteration is very small as the search is concentrated only in regions in which conflicts appear. By including the tabu mechanism in this technique, cycles during the search are avoided as much as possible, and thus different regions of the search space are explored. This mechanism improves the MC technique, as the cycles can appear more easily in this technique, especially when the search is concentrated only in regions within the violated constraints. Regarding the MC-RW strategy, the random walk introduces some diversification process during the search. Although the MC strategy includes some randomness by selecting randomly the cell to be swapped, in the last phase of the search, usually only few constraints are violated and, consequently, the cell which is selected randomly is selected only between small numbers of cells which appear in the conflicts. RW provides further randomization by selecting randomly the position with which the selected cell should be swapped. This mechanism looks to make it possible for the MC strategy to escape from the local optimum.

### B. *Comparison with other methods in literature and with a commercial workforce scheduling system*

In this section the results of min-conflicts heuristic with tabu mechanism proposed in this paper are compared with

*Table 8*: (MC-RW) Min-conflicts based heuristic with random walk (10 runs)

| Example | Number of Evaluat. | Time(sec) | Solutions |
|---|---|---|---|
| 1 | 4885.6 | 0.07 | 10 |
| 2 | 7374.3 | 0.11 | 10 |
| 3 | 28205.4 | 0.68 | 10 |
| 4 | 5920.5 | 0.13 | 10 |
| 5 | 23291.4 | 0.48 | 10 |
| 6 | 4308.1 | 0.06 | 10 |
| 7 | 1.40E+06 | 51.68 | 10 |
| 8 | 28332.2 | 0.63 | 10 |
| 9 | 205379 | 11.48 | 10 |
| 10 | 25878.4 | 0.94 | 10 |
| 11 | 274413 | 10.02 | 10 |
| 12 | 89181.1 | 2.17 | 10 |
| 13 | 56653.4 | 1.74 | 10 |
| 14 | 35890.5 | 0.79 | 10 |
| 15 | 3.60E+06 | 328.52 | 10 |
| 16 | 24319.9 | 0.90 | 10 |
| 17 | 34727.7 | 1.31 | 10 |
| 18 | 108731 | 7.22 | 10 |
| 19 | 312237 | 44.18 | 10 |
| 20 | 317008 | 78.36 | 10 |

*Table 9*: (MC-T) Min-conflicts based heuristic with tabu mechanism (10 runs)

| Example | Number of Evaluat. | Time(sec) | Solutions |
|---|---|---|---|
| 1 | 5071.1 | 0.07 | 10 |
| 2 | 4890.5 | 0.07 | 10 |
| 3 | 16798.5 | 0.42 | 10 |
| 4 | 5183.5 | 0.11 | 10 |
| 5 | 20638.9 | 0.43 | 10 |
| 6 | 5941.1 | 0.08 | 10 |
| 7 | 1.42E+06 | 52.79 | 10 |
| 8 | 33455 | 0.74 | 10 |
| 9 | 283803 | 15.96 | 10 |
| 10 | 15815.7 | 0.60 | 10 |
| 11 | 354996 | 13.15 | 10 |
| 12 | 47553.7 | 1.17 | 10 |
| 13 | 27535 | 0.87 | 10 |
| 14 | 34284.6 | 0.76 | 10 |
| 15 | 1.72E+06 | 159.04 | 10 |
| 16 | 13728.4 | 0.54 | 10 |
| 17 | 57943.7 | 2.16 | 10 |
| 18 | 101432 | 6.83 | 10 |
| 19 | 543171 | 75.83 | 10 |
| 20 | 284938 | 71.38 | 10 |

other approaches proposed in the literature for the three existing problems in literature. The results are compared with [7],[33], [28], and [32]. In [7] the rotating workforce scheduling problem is solved by modeling it as a network flow problem. Their algorithm was implemented in Fortran and was tested on an IBM 3081 computer. In [28] this problem is solved by using a constraint programming algorithm which was coded in ILOG Solver 4.4. They tested their algorithms with a Sun Sparc Ultra 5 workstation (400MHz, 128 Mb RAM). Algorithms proposed in [33] are part of the commercial software First Class Scheduler (FCS). The algorithms proposed in [33], [32], and the algorithms proposed in this paper are tested on a Pentum 4, 1,8GHZ, 512 MB RAM. In Table 12 the results for three problems are shown. For each of approaches proposed in literature the time needed to reach the first solution is shown (for MC-T this is the average time over 10 runs). Although the experiments were performed on different computers and the results are not totally comparable because of some differences in constraints, we can conclude from the table that the MC-T approach proposed in this paper gives very good results for those problems and comparable results with those given in literature.

In Table 13 the comparison of MC-T and the simple genetic algorithm proposed in [32] for three problems are shown. For each of the approaches the average number of evaluations over 10 runs to reach the first solution is shown. We can conclude from the table that the MC-T gives much better resuts than the given simple genetic algorithm. However, this comparison does not give a general argument that our methods give better results in comparison to the evolutionary approaches, as the proposed genetic algorithm could be probably further improved to give better results than the simple genetic algorithm. Recently, different variants of genetic algorithms [36], [2], [3], [13] have been used successfully for related workforce scheduling problems. A direct comparison between these approaches for the problem we consider in this paper is not possible, as the problem solved by these approaches differs in several aspects from our problems, and these approaches can not be used directly to solve rotating workforce scheduling. An interesting issue for future work is the adaptation of these methods to rotating workforce scheduling, but this is outside the scope of this paper.

*Table 10*: One of the solutions for problem 2 with the MC-T method

| Emp./day | Mon | Tue | Wed | Thu | Fri | Sat | Sun |
|---|---|---|---|---|---|---|---|
| 1 | D | D | D | D | - | - | - |
| 2 | A | A | A | A | A | A | A |
| 3 | - | - | - | - | N | N | N |
| 4 | N | N | N | N | - | - | - |
| 5 | - | D | D | D | D | D | D |
| 6 | D | - | - | - | N | N | N |
| 7 | N | N | N | N | - | - | - |
| 8 | - | A | A | A | A | A | A |
| 9 | A | - | - | - | D | D | D |

For the next 17 Problems proposed in this paper, we can not make a comparison with other methods (except the method proposed in [33] and [32]), as we do not have access to these methods. MC-T gives much better results compared to sim-

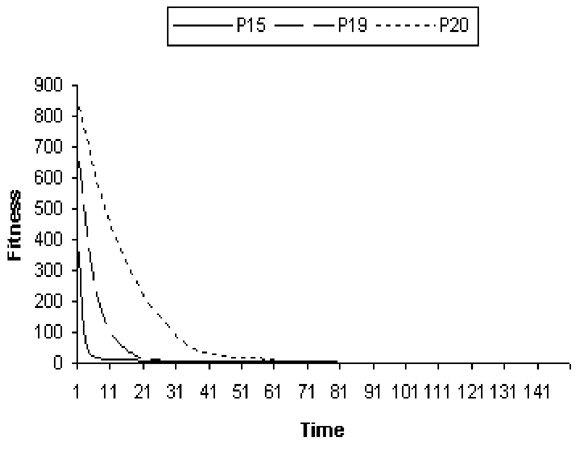


**Figure. 4**: Change of fitness over the time for problems 15,19,20 in case of MC-TS

*Table 11*: One of the solutions for problem 4 with the MC-T method

| Emp./day | Mon | Tue | Wed | Thu | Fri | Sat | Sun |
|---|---|---|---|---|---|---|---|
| 1 | A | A | A | A | A | A | - |
| 2 | A | A | A | - | - | - | - |
| 3 | D | D | D | A | A | A | - |
| 4 | N | N | N | - | - | - | - |
| 5 | A | A | A | N | N | - | - |
| 6 | A | A | A | A | A | A | - |
| 7 | A | A | A | A | A | A | - |
| 8 | - | - | - | D | D | D | - |
| 9 | D | D | D | D | D | D | - |
| 10 | D | D | D | A | A | A | - |
| 11 | - | - | - | D | D | D | - |
| 12 | D | D | D | D | D | D | - |
| 13 | D | D | D | D | D | D | - |

ple genetic algorithm proposed in [32] and we do not give results for other problems here (results for three problems are given Table 13). However, for other examples we compare the best method proposed in this paper with the method included in the last version of the commercial software for generation of rotating workforce schedules (First Class Scheduler (FCS) [21]). This system has been used very successfully since year 2000 in practice for many companies in Europe. To our best knowledge FCS is the state-of-art commercial system for generation of rotating workforce schedules. FCS is based on interaction with a decision maker. To make possible direct comparison of FCS with the methods proposed in this paper the process of generation of solution in FCS is automated.

The comparison of the methods proposed in this paper and the last version of FCS is given in Table 14. The second column in the table named groups represents the number of groups (or employees if the group has only one employee) for each example. The third column represents the time in seconds needed to generate a first solution in FCS, and the last column represents the average time from 10 runs to find solutions using tabu search in combination with the min-conflicts based heuristic (MC-T). The experiments for MC-T and FCS were performed on the same machine (Pentum 4, 1,8GHZ, 512 MB RAM).

*Table 12*: Comparison between MC-T and other aproaches in literature

| Example | Groups | [7] | [33] | [28] | MC-T |
|---|---|---|---|---|---|
| 1 from [9] | 9 | 73.54 | 0.9 | 3.78 | 0.07 |
| 2 from [27] | 9 | 310.84 | 0.4 | 0.03 | 0.07 |
| 3 from [24] | 17 | 457.98 | 1.9 | 10.26 | 0.42 |

*Table 13*: Comparison (regarding number of evaluations) between MC-T and a simple genetic algorithm

| Example | Groups | [32] | MC-T |
|---|---|---|---|
| 1 from [9] | 9 | 485070 | 5071,1 |
| 2 from [27] | 9 | 531840 | 4890,5 |
| 3 from [24] | 17 | 3302941 | 16798,5 |

*Table 14*: Comparison between First Class Scheduler and MC-T

| Ex. | Groups | FCS (time in sec) | MC with tabu list (sec) |
|---|---|---|---|
| 1 | 9 | 0.9 | 0.07 |
| 2 | 9 | 0,4 | 0.07 |
| 3 | 17 | 1.9 | 0.42 |
| 4 | 13 | 1.7 | 0.11 |
| 5 | 11 | 3.5 | 0.43 |
| 6 | 7 | 2 | 0.08 |
| 7 | 29 | 16.1 | 52.79 |
| 8 | 16 | 124 | 0.74 |
| 9 | 47 | >1000 (?) | 15.96 |
| 10 | 27 | 9.5 | 0.60 |
| 11 | 30 | 367 | 13.15 |
| 12 | 20 | >1000 (?) | 1.17 |
| 13 | 24 | >1000 (?) | 0.87 |
| 14 | 13 | 0.54 | 0.76 |
| 15 | 64 | >1000 (?) | 159.04 |
| 16 | 29 | 2.44 | 0.54 |
| 17 | 33 | >1000 (?) | 2.16 |
| 18 | 53 | 2.57 | 6.83 |
| 19 | 120 | >1000 (?) | 75.83 |
| 20 | 163 | >1000 (?) | 71.38 |

From Table 14 we can conclude that the MC-T method proposed in this paper outperforms FCS almost in all instances. Considering larger instances FCS solves faster instances 7 and 18. These two problem instances have the same workforce requirements for all days and shifts. According to previous experiences with FCS, this system can usually solve instances of such nature, even if they are large instances. Based on empirical results for the methods proposed here such instances seem not to be easier to solve compared to the other problems of similar size. For instances in which there is written FCS '>1000 (?)' in column three, FCS could not find a solution in 1000 seconds and it is not clear if FCS can find solutions for these problems. For small problems FCS has its advantages, because of the generation of more solutions and the possibility to include soft constraints in interaction with a decision maker. However, methods proposed in this paper improve the performance of a system for solving of middle and larger instances of problems which can not be solved by FCS in a reasonable amount of time.

### C. Conclusions

In this paper we proposed novel methods for solving rotating workforce scheduling problems. We proposed the tabu search based algorithm and a method which is based on ideas of the min-conflicts based heuristic. Introducing the min-conflicts strategy and random walk into tabu search was further considered. The method based on min-conflicts strategy was extended by introducing the tabu mechanism into this strategy. Additionally, combination of all three methods: min-conflicts heuristic, random walk and tabu search was considered.

The proposed methods have been implemented and experimentally evaluated for examples from literature and other real life examples, which appeared in a broad range of organizations. Experimental results show that combination of tabu search and min-conflicts strategies proposed in this paper improves significantly the performances of tabu search and min-conflicts based heuristic alone. In particular, the introduction of the random walk and min-conflicts into tabu search improves the performance of tabu search alone. The min-conflicts strategy is improved by introducing the tabu mechanism and random walk. Overall, experimental results show that the min-conflicts based strategies are a good choice for this problem to reach very good results in a short time. The best results are reached by introducing the tabu mechanism or random walk into a min-conflicts based heuristic. Both techniques in each run can find solutions for each problem in a short amount of time.

The min-conflicts heuristic with tabu mechanism has been compared with four other approaches from literature for 3 available benchmark problems proposed earlier in the literature. The results show that our approach gives very comparable results for these problems. Furthermore, comparison for 20 problems with a commercial workforce scheduling system First Class Scheduler show that the proposed approach improves significantly the performance of this system, especially for large instances. The proposed methods in this paper are in the process of being included in the latest version of a system for automatic generation of rotating workforce schedules.

For the future, we are considering applying the hybrid techniques proposed in this paper to solve the nurse scheduling problem. This would only be the adaptation of existing methods for another problem, but it would be of interest to investigate how good these methods perform for nurse scheduling and, in particular, how they compare to the methods recently proposed for nurse scheduling [15], [34], [17], [36], [38], [19], [2], [3]. In this problem other constraints may appear and individual preferences of employees can be taken into consideration. Furthermore, it would be of interest to adapt the methods used successfully for nurse scheduling to solve rotating workforce scheduling, and investigate how good these methods perform for this type of problem.

## Acknowledgment

This work was supported by FWF (Austrian Science Fund) Project No. **Z29-N04**. We would like also to thank the reviewers for their detailed comments on this paper.

## References

[1] Emile Aarts and Jan Karl Lenstra, editors. *Local Search in Combinatorial Optimization*. Wiley, 1997.

[2] U. Aickelin and K. Dowsland. An indirect genetic algorithm for a nurse scheduling problem. *Computers and Operations Research*, 31:761–778, 2003.

[3] U. Aickelin and P. White. Building better nurse scheduling algorithms. *Annals of Operations Research*, 128:159177, 2004.

[4] H. K. Alfares. Survey, categorization, and comparison of recent tour scheduling literature. *Annals of Operations Research*, 127(1-4):145–175(31), 2004.

[5] M. Krishnamoorthy B. Owens A.T. Ernst, H. Jiang and D. Sier. An annotated bibliography of personnel scheduling and rostering. *Annals of Operations Research*, 127:21144, 2004.

[6] Harald Meyer auf'm Hofe. Solving rostering tasks as constraint optimization. In *Proceedings of the 3rd international conference on the practice and theory of automated timetabling (PATAT 2000), pages 280-297, Konstanze, Germany,* 2000.

[7] Nagraj Balakrishnan and Richard T. Wong. A network model for the rotating workforce scheduling problem. *Networks*, 20:25–42, 1990.

[8] Edmund K. Burke, Patrick De Causmaecker, Greet Vanden Berghe, and Hendrik Van Landeghem. The state of the art of nurse rostering. *J. of Scheduling*, 7(6):441–499, 2004.

[9] B. Butler. Computerized manpower scheduling. Master's thesis, University of Alberta, Canada, 1978.

[10] Patrick De Causmaecker and Greet Vanden Berghe. Relaxation of coverage constraints in hospital personnel rostering. *E. Burke and P. De Causmaecker (Eds.): PATAT 2002, LNCS*, 2740:129–147, 2003.

[11] Weil G. Chan, P. Cyclical staff scheduling using constraint logic programming. *In: Burke, E, Erben W. (Eds.): PATAT 2000, Lecture Notes in Computer Science*, 2079:159–175, 2001.

[12] Marko Chiarandini, Andrea Schaerf, and Fabio Tiozzo. Solving employee timetabling problems with flexible workload using tabu search. In *Proceedings of the 3rd international conference on the practice and theory of automated timetabling (PATAT 2000), pages 298–302, Konstanze, Germany*, 2000.

[13] Peter I. Cowling, Nic Colledge, Keshav P. Dahal, and Stephen Remde. The trade off between diversity and quality for multi-objective workforce scheduling. In *EvoCOP*, pages 13–24, 2006.

[14] K. A. Dowsland and J. M. Thompson. Solving a nurse scheduling problem with knapsacks, networks and tabu search. *Journal of the Operational Research Society*, 51(7):825–833, 2000.

[15] Kathryn Dowsland. Nurse scheduling with tabu search and strategic oscillation. *European Journal of Operational Research*, 106(2-3):393–407, 1998.

[16] P. De Causmaecker E. K. Burke and G. Vanden Berghe. A hybrid tabu search algorithm for the nurse rostering problem. *B. McKay et al. (Eds.): Simulated Evolution and Learning,Lecture Notes in Artificial Intelligence*, 1585:187–194, 1999.

[17] P. De Causmaecker E. K. Burke and G. Vanden Berghe. Novel metaheuristic approaches to nurse rostering problems in belgian hospitals. *n J. Leung (ed.), Handbook of Scheduling: Algorithms, Models and Performance Analysis, Chapter 44, CRC Press*, 106:44.144.18, 2004.

[18] P. De Causmaecker G. Vanden Berghe E.K. Burke, P. Cowling. A memetic approach to the nurse rostering problem. *Applied Intelligence special issue on Simulated Evolution and Learning*, 15(3):199214, 2001.

[19] F. Della Croce R. Tadei F. Bellanti, G. Carello. A greedy-based neighborhood search approach to a nurse rostering problem. *European Journal of Operational Research*, 153:2840, 2004.

[20] P. Galinier and J.K. Hao. A general approach for constraint solving by local search. *Journal of Mathematical Modelling and Algorithms*, 3(1):73–88, 2004.

[21] Johannes Gärtner, Nysret Musliu, and Wolfgang Slany. Rota: A research project on algorithms for workforce scheduling and shift design optimisation. *Artificial Intelligence Communications*, 14(2):83–92, 2001.

[22] Fred Glover and Manuel Laguna. *Tabu search*. Kluwer Academic Publishers, 1997.

[23] Fred Glover and Claude McMillan. The general employee scheduling problem: An integration of MS and AI. *Comput. Ops. Res.*, 13(5):563–573, 1986.

[24] N. Heller, J. McEwen, and W. Stenzel. Computerized scheduling of police manpower. *St. Louis Police Department, St. Louis, MO*, 1973.

[25] S. Kirkpatrick, C. D. Gelatt Jr., and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, 1983.

[26] G. Laporte. The art and science of designing rotating schedules. *Journal of the Operational Research Society*, 50:1011–1017, 1999.

[27] G. Laporte, Y. Nobert, and J. Biron. Rotating schedules. *Eur. J. Ops. Res.*, 4:24–30, 1980.

[28] G. Laporte and G. Pesant. A general multi-shift scheduling system. *Journal of the Operational Research Society*, 55/11:1208–1217, 2004.

[29] Hoong Chuin Lau. On the complexity of manpower shift scheduling. *Computers. Ops. Res.*, 23(1):93–102, 1996.

[30] J. Li and U. Aickelin. The application of bayesian optimization and classifier systems in nurse scheduling. *Proceedings of the 8th International Conference on Parallel Problem Solving from Nature (PPSN VIII), LNCS*, 3242:581–590, 2004.

[31] Steven Minton, Mark D. Johnston, Andrew B. Philips, and Philip Laird. Minimizing conflicts: a heuristic repair method for constraint satisfaction and scheduling problems. *Artificial Intelligence*, 58:161–205, 1992.

[32] Michael Mörz and Nysret Musliu. Genetic algorithm for rotating workforce scheduling. In *Proceedings of second IEEE International Conference on Computational Cybernetics ( pages 121-126), Vienna, Austria*, 2004.

[33] Nysret Musliu, Johannes Gärtner, and Wolfgang Slany. Efficient generation of rotating workforce schedules. *Discrete Applied Mathematics*, 118(1-2):85–98, 2002.

[34] K. Nonobe and T. Ibaraki. A tabu search approach to the constraint satisfaction problem as a general problem solver. *European Journal of Operational Research*, 106:599–623, 1998.

[35] Colin R. Reeves, editor. *Modern Heuristic Techniques for Combinatorial Problems*. Halsted Pr., 1993.

[36] K. A. Dowsland U. Aickelin. Exploiting problem structure in a genetic algorithm approach to a nurse rostering problem. *Journal of Scheduling*, 3:139–153, 2000.

[37] Richard J Wallace and Eugene C Freuder. Heuristic methods for over-constrained constraint satisfaction problems. In *Overconstrained Systems, Springer 1106*, 1996.

[38] Christine A. White and George M. White. Scheduling doctors for clinical training unit rounds using tabu optimization. *E. Burke and P. De Causmaecker (Eds.): PATAT 2002, LNCS 2740*, pages 120–128, 2003.