An Approach for Automated Surgery Scheduling

Karl-Heinz Krempels and Andriy Panchenko[⋆]

RWTH Aachen University,
Computer Science Department - Informatik IV,
Ahornstr. 55, D-52074 Aachen, Germany
{krempels,panchenko}@cs.rwth-aachen.de

Abstract. The planning of surgical operations forms a substantial element of hospital management. It is characterized by high complexity, which is caused by the uncertainty between the capacity offered and the true demand. Also, as emergency cases occur the planning requirements change. A semi-automated dialog-based system is therefore preferred rather than either fully manual or fully automated systems. This is because of the inability of the later to recognize the changes in a high dynamic environment and to take the responsibility for decisions made. As it has to be possible to add new tasks in the planning process "on the fly" and to adequately plan new situations, we involve a human planner in the scheduling activity. The planner acts as a "sensor" to identify changes as they occur and integrates his knowledge as well as his decision-making competence into the planning process. The proposals for the schedules are however made with the help of the heuristics. In order for a schedule to be accepted by those involved, it should take account of the interests and preferences of all the human actors. Existing systems do not do this and therefore suffer from levels of non-acceptance of their resulting schedules. In this paper we discuss suitable heuristics for operating theatre scheduling, the limits to which preferences can be considered in the scheduling process, and the validation of the approach in an experimental set of hospital scenarios.

1 Introduction

Operating theatre scheduling deals with assignment of limited hospital resources (rooms, doctors, nurses, etc.) to jobs (patient treatments, surgery, etc.) over the time, in order to perform a set of tasks according to their needs and priorities, and to optimize usage of hospital resources [13]. The whole process is restricted by a whole series of constraints, limitations and preferences [2]. It is further characterized by a high level of complexity due to:

1. the uncertainty of the relationship between the capacity offered and the true demand,
2. the inability to predefine treatments' workflow,

[⋆] This work was supported by the German Research Foundation in the research priority programme SPP 1083 – *Intelligent Agents in Real-World Business Applications.*

3. emergency cases, causing disturbance in existing schedules and the need to adapt as situation changes.

Typically, scheduling is currently done manually and involves specialized staff. [9] discuss an example of an optimization that can be achieved using process automization for nurses rostering[1]. In the hospital considered, one person takes 3-5 full working days to produce the nurses' schedule for the period of one month. That is only to the level of assigning shifts. Similar results have been reported in other studies [4]. Existing industrial schedulers usually assume a predefined workflow. Furthermore, they do not take staff and patient preferences into account and so are not generally applicable to the hospital domain [2]. The current methods in this domain are therefore: no planning, pen and paper, non-intelligent tool-based. Fully automated systems are not accepted because of their inability to give proper consideration to preferences and the need to show clear responsibility for decision making.

Existing solutions consider only sub-problems such as: nurse rostering, coordination, or surgery planning. Most tools provide only GUI for manual scheduling, although some check the validity of completed schedules, and some of the more advanced generate draft schedules (however, preferences are not taken into account). Some examples of such systems are Medico//S, ORBIS[2], MEDICUS [14], CARE2X[3].

This paper presents a problem, based on real-world requirements. In the following section we present our approach for semi-automatic, dialog- and preference-based scheduling in hospital scenarios. Then, after introducing the scheduling problem to be considered, the domain of discourse and the made assumptions in Section 2, we describe the problem decomposition, the discussion of the developed heuristics for each of the identified subproblems and provide a complexity analysis for the developed heuristics. In Section 3 the experimental context is described and an evaluation of the results presented. We finalize the discussion with conclusions based on the evaluation and look forward to further challenges.

2 Scheduling Heuristic

We divide the operating theatre scheduling problem into the four sub-problems: preference-based personnel assignment to shifts, preference-based building of teams in a shift, preference-based task (operation) assignment to teams and room assignment to tasks. For each of the sub-tasks we have developed a heuristic. Furthermore, we have implemented suitable problem-solving methods and evaluated the implementation by the means of simulation. The heuristics produce a proposal for the schedule. It is however up to the human planner whether to accept it in its entirety, accept parts of it, or to reject it completely. We propose a semi-automated dialog-based approach for the rostering.

[1] rostering and scheduling are used as synonyms here

[2] http://www.sieda.com/

[3] http://www.care2x.com/

In this section we discuss the heuristics for operating theatre scheduling according to the division of sub-tasks as described above. We assume that the preferences of the actors involved that need to be taken into account are provided in a frame-based representation, and that their context will be valid in the context of the domain: shift, team, task, and room [10].

Based on the nature of the preference, we distinguish between static (coworker, treatment) and dynamic (shift) ones. Each preference has a value from the closed domain {*willingly, undecided, unwillingly*}. We define the preference domain range as

$$\mathcal{D} = \{w, u, \bar{w}\},$$

where w stands for "willingly", u for "undecided" and $\bar{w}$ for "unwillingly". Each employee is able to specify a preference $x_i \in \mathcal{D}$ for another employee expressing his/hers willingness to work together with that employee. In the same way it is possible to specify preferences for shifts. If the preference is not specified it has the value u (undecided) by default.

There are basically two approaches for the staff rostering problem: *cyclic*[4] and *non-cyclic* [1]. In the first one, several sets of schedules are generated that satisfy the demand requirements. Staff are then rotated from one set of schedules to another in consecutive planning horizons. So for example, the schedule may be repeated every week or with a two week interval. Although it is easy to implement, cyclic schedules impose inflexibilities, and there is therefore less acceptance of the resulting outcome. We therefore use a non-cyclic algorithms.

It is also important to note that there are two possible approaches for team continuity: first one based on the static team assignment, that means a team remains unchanged until the end of the duty; the second one allows the reassignment of members to other teams (build new ones) when no job is assigned to the current one, in order to achieve continuity of occupation. Static assignment is less flexible, but has much less computational complexity since there is no need to search for the common time gaps across all team members. Furthermore, in the second, there may be individual time gaps that will not be filled in the future with any tasks.

2.1 Preference-Based Adaptive Assignment of Personnel to Shift

Preference-based assigning of personnel to shift considers:

- hierarchical position (senior physician, assistant doctor, anaesthetist, nurse);
- contract work hours (treated as soft-constraints);
- shift preferences (that have dynamic character since prefered working time can vary from day to day);
- hard-constraints:
    - minimum/maximum number of personnel in the shift (day, shift and qualification dependent);

[4] sometimes also called *rotational*

  - maximum duty duration;
  - minimum inter-duty pause;
- fulfilled wishes quota.

The heuristic selects personnel so as to avoid hard-constraint violations for each qualification in the following order of preference value:

$$\textit{willingly} \succ \textit{undecided} \succ \textit{unwillingly}.$$

The best and simplest case is when the number of actors that work gladly in the shift is within the bounds of the hard-constraints and maximum working time is not exceeded. This means that they can all be selected for the duty and it can be continued with the next lower position in the hierarchy of the department. Unfortunately the hard-constraints often prevent this. So, for example, the number of available personnel that would work "willingly" or "undecided" in the shift may be smaller than the minimum staffing level required. In this case all of the actors with these preferences have to be selected and those with the lower preferences have to fill the gaps. It may also happen that only some of the actors with the preference "unwillingly" for the current shift have to be selected. So those that are not selected will be in a better position compared to the selected ones. Furthermore, in the case of selecting only some with the "willingly" preference, those that are not selected will find themselves in the worse condition compared to the others. The selection procedure therefore contributes to placing some actors in a worse condition, compared to the others. Such a decision has to be memorized in order to achieve a degree of fairness in the scheduling. That means that an additional measure has to be introduced in order to keep track about the number of actor's wishes that have been, respectively have not been taken into account. The measure is mapped to the actor's weight. This weight is then used to influence further selection decisions of the form "some from many" in order to achieve fairness of the scheduling and give due consideration to the wishes of individual actors. It should also be considered that medical personnel are usually contracted to work for a certain amount of hours per week. The selection process therefore also needs to take into account that physicians that have not yet completed their weekly contract hours should have higher priority in getting a shift than those that are already on overtime. The weight (importance) of the choice for an actor can be determined with the help of the following weight function:

$$\begin{aligned} Weight_i =& RemainingWorkShifts \cdot PositionWeight \cdot \alpha \\ &+ Weight_{i-1} \cdot stimulus \cdot (1-\alpha) \end{aligned} \tag{1}$$

where

- $0.5 \leq \alpha < 1$ coefficient that determines influence of the previous weight value. It is assumed, that the remaining working time has more influence on the weight function than whether wishes have been complied with, or rejected;

- $PositionWeight$ weight, determined by the position of the actor;
- $RemainingWorkShifts$ quota regular working shifts stated in the contract as represented by the soft-constraints;
- $Weight_{i-1}$ previous (old) weight value;
- $stimulus$ determines the weight adjustment manner:
  - > 1 in case of dissatisfying selection;
  - $= 1$ if no preference was specified;
  - < 1 in case of taking into account a wish.

$PositionWeight$ is used in the weight function since it is assumed that duties may require personnel with some qualification, independent of the position. However, the wishes of the physician with a higher position in the ward hierarchy are given a higher significance.

Defined as above, the stimulus effect decreases the weight function in the case of selecting an actor with the preference "willingly" or by not selecting one with the preference "unwillingly". The value increases by selecting with "unwillingly" or by not selecting with "willingly". The adjustment strategies of the weight with respect to stimulus can be seen in Table 1.

preference value	**stimulus for selection**	**stimulus for rejection**
"willingly"	< 1	> 1
"undecided"	$= 1$	$= 1$
"unwillingly"	> 1	< 1

Table 1. Weight Adjustment Strategies

The preference "undecided" is treated as neutral and does not influence the stimulus, but the weight function changes in case of selection because of the adjustment of remaining work shifts. At the beginning of the week the weight is initialized as:

$$Weight_0 = contractualWorkShifts \cdot PositionWeight \tag{2}$$

The weight is always adjusted each time the person is selected for the shift (since the *RemainingWorkShifts* number changes) and also, for all those actors with a preference $\in$ {*willingly, unwillingly*} that were able to take the duty (i.e. caused no hard-constraints violation). For not selected actors only the stimulus will have an impact on the weight. If the *RemainingWorkShifts* reaches the value zero, the weight function update decreases significantly if the actor is selected (since it is treated as an undesired overtime):

$$Weight_i = Weight_{i-1} \cdot stimulus \cdot (1 - \alpha) \tag{3}$$

In case of rejection, for those actors with a preference $\in$ {*willingly, unwillingly*} only the stimulus has an influence on the weight:

$$Weight_i = Weight_{i-1} \cdot stimulus \qquad (4)$$

All relevant constraints and their characteristics are given in Table 2.

The minimum and maximum number of personnel required in the shift has a dynamic character since these values depend on the day (e.g. regular week day, weekend, etc.). Where more than minimum required personnel with a preference *willingly* are available, they are all selected up the to maximum allowed number by the hard-constraint. In the case of preferences ∈ {*undecided, unwillingly*} only the minimum number stated in the constraints are selected. The heuristic therefore satisfies the wishes of as many actors as possible. It is important to mention, that the personnel requirements have following restrictions:

max required senior physician ≤ min required assistant doctor
max required assistant doctor ≤ min required nurse
max required assistant doctor ≤ min required anaesthetist

Where there is a need to make a selection decision "select some from many", those with the higher value of the weight function are chosen. This contributes towards fairness, since higher values mean that there are more contractual working hours still unused in the current week, a higher position in the ward's hierarchy or smaller number of preferences already taken into account.

In view of the linear simplicity of the heuristic we do not discuss the complexity analysis for shift assignment.

2.2 Preference-Based Building of Teams in a Shift

After the staff selection for the duties, the available personnel in the hospital department is determined for each shift. As tasks arrive, there is a need to group the shift personnel into teams in order to perform those tasks. Requirements for personnels' qualifications, resources, specializations of the involved actors in each treatment have to be considered. In order to increase the acceptance of the planning system in the hospital, it is also important to take into account personal preferences of the involved actors. So, for example, the doctor "Z" may prefer to work better with nurse "Y" than with the nurse "X".

Constraint	**Type**	**Character**
shift preference	preference	dynamic
max shifts per duty	hard-constraint	static
min inter duty pause	hard-constraint	static
min required personnel	hard-constraint	dynamic
max required personnel	hard-constraint	dynamic
contract work hours	soft-constraint	static

Table 2. Constraints for Shift Assignment

The problem can be formalized as follows: given S, D, E, N - sets consisting of senior physicians, assistant doctors, anaesthetists and nurses having duty in a shift, so that:

$$|S| = s,\ |D| = d,\ |E| = e,\ |N| = n \qquad \text{with } s < d < e < n.$$

This means that there are always more nurses in the shift than anaesthetists and more anaesthetists than assistant doctors, and more assistant doctors than senior physicians. Each employee can specify a preference with respect to another, expressing his willingness to work with them:

$$\forall a_i, a_j \in S \cup D \cup E \cup N, \quad i \neq j \qquad \exists\ \mathcal{P}_{a_j}(a_i) = x_{ij}, \\ x_{ij} \in \mathcal{D}, \quad i, j \in (1, s+d+e+n).$$

Every employee is also characterized by the weight, that depends first of all on the hierarchical position of the person:

$$\forall a \in S \cup D \cup E \cup N \qquad \exists\ g(a) \geq 0.$$

The utility of team i, consisting of $\tilde{n}$ actors is defined as

$$\mathcal{U}_i = \sum_{k=1}^{\tilde{n}} q_{a_1,\ldots,a_{k-1},a_{k+1},\ldots,a_{\tilde{n}}}(a_k) \cdot g(a_k) \tag{5}$$

where

$$q_{a_j}(a_i)_{i \neq j} = \begin{cases} 1 & \text{if } \mathcal{P}_{a_j}(a_i) = w, \\ 0 & \text{if } \mathcal{P}_{a_j}(a_i) = u, \\ -1 & \text{if } \mathcal{P}_{a_j}(a_i) = \bar{w} \end{cases} \tag{6}$$

is the value of the coworker preference of actor a_i regarding the colleague a_j. Correspondingly, for a set of n coworkers $a_1, \ldots, a_n$ it is defined as

$$q_{a_1,\ldots,a_{k-1},a_{k+1},\ldots,a_{\tilde{n}}}(a_k) = q_{a_1}(a_k) + \cdots + q_{a_{k-1}}(a_k) + q_{a_{k+1}}(a_k) + \ldots + q_{a_{\tilde{n}}}(a_k). \tag{7}$$

The total utility of h teams in a shift, each with its own size of $\tilde{n}_i$ is defined as

$$\mathcal{U}_{total} = \sum_i \mathcal{U}_i = \sum_{i=1}^{h} \sum_{k=1}^{\tilde{n}_i} q_{\{a_1,\ldots,a_{k-1},a_{k+1},\ldots,a_{\tilde{n}}\}_i}(a_{ki}) \cdot g(a_{ki}). \tag{8}$$

The goal is now to build teams in such a way, that the total utility function (satisfaction of coworkers to work together) is maximized.

Optimization Criteria. It is assumed that there exist two configuration patterns for operation teams: those with four actors (senior physician, assistant doctor, anaesthetist and nurse) and, for less complicated operations, consisting of three actors (assistant doctor, anaesthetist and nurse). The goal is to find teams, consisting of staff members from a shift, and maximizing the total utility function.

So, for example, in case of preference and weight specifications as captured in Table 3 the utility of the three-member teams is calculated as follows:

$$\begin{aligned}\mathcal{U}(< A, B, C >) &= q_{BC}(A) \cdot g(A) + q_{AC}(B) \cdot g(B) + q_{AB}(C) \cdot g(C) \\ &= (0+1) \cdot 80 + (-1+1) \cdot 60 + (-1+0) \cdot 40 = 40.\end{aligned} \tag{9}$$

The total utility is then the sum of weighted adjacent satisfaction of all the team members. So, the advanced teams are built up for all available senior physicians. For the rest $t = 1, \ldots, d - s$, assistants (where $s = |S|$, $d = |D|$) smaller teams are built in a similar way, in order to facilitate treatment of another (lower) complexity class. The maximum sum reflects the highest total utility of all the teams involved and would be the optimal solution for the problem.

Complexity Analysis Discussion. To achieve the maximum utility value, it is necessary to compare the sums of all the possible team configurations in the shift. The total number of possibilities for a set of s teams of four members in the shift is

$$\sum_{i=0}^{s-1}(s-i)(d-i)(e-i)(n-i), \tag{10}$$

which requires a computational complexity of $O(s \cdot d \cdot e \cdot n)$, or $O(n^4)$.

It is important also to consider what happens when the team size is not fixed. The complexity results are completely different. The problem can be represented as a graph where each vertex represents a team configuration and an edge between two vertices exists in cases where they have at least one common actor. The goal is to find maximum independent set in the graph. An *independent set* in a graph $G = (V, E)$ is a subset $I \subseteq V$, such that $\forall x, y \in I$, $\{x, y\} \notin E$. The independent set problem consists of finding a maximum (largest) independent set in a graph. It is well-known to be NP-complete [7]. In natural integer programming formulation the finding of the optimal solution with team weights w_j

Employee x	**Preference to A** $P_A(x)$	**Preference to B** $P_B(x)$	**Preference to C** $P_C(x)$	**Weight** $g(x)$
A	-	u	w	80
B	$\bar{w}$	-	w	60
C	$\bar{w}$	u	-	40

Table 3. Example for Team Utility Calculation

for $j \in V$ (V is a set of all possible team combinations in a shift), $|V| = s \cdot d \cdot e \cdot n$, is

$$\begin{aligned}
&\text{Maximize } \textstyle\sum_{j=1}^{n} w_j x_j \\
&\text{subject to } x_i + x_j \leq 1 \text{ (for every edge } (i,j) \text{ in the graph)} \\
&\qquad\qquad 0 \leq x_j \leq 1 \quad (j = 1, \ldots, s \cdot d \cdot e \cdot n) \\
&\qquad\qquad x_j \text{ integer} \quad (j = 1, \ldots, s \cdot d \cdot e \cdot n)
\end{aligned}$$

Hochbaum([7]) gives a summarization of all the approximation algorithms for the weighted independent set problem known to date. A guaranteed good approximation is only possible if the graph has some special features such as being planar (in this case it is possible to give an approximation guarantee of $\frac{1}{2}$). In the general case it is of the $\frac{1}{\lceil \frac{\Delta+1}{3} \rceil}$, where Δ is the maximum degree of a vertex in the graph. In the case of a teams graph where teams consist only of four actors, the degree of each vertex is

$$s \cdot d \cdot e \cdot n - (s-1)(d-1)(e-1)(n-1) - 1.$$

This is nothing other than a total amount of all possible combinations of teams, having substituted those combinations, not having at least one common actor with one selected team. If the hospital department consists of 25 members in each position, the vertex degree Δ has the value 58848, considering only teams of four actors. In this case the approximation degree of about 0.00005 or $5 \cdot 10^{-5}$ can be guaranteed with the complexity $O(m\Delta)$, which does not seem to inspire much (m is the number of edges). Guo et al. ([6]) has experimentally compared and found that using the simulated annealing heuristic to solve the set packing problem (which is polynomially equivalent to the independent set problem [7]) outperforms previous approximation methods, and based on that heuristic ILOG CPLEX obtains results within smaller timescales.

Heuristic. Now that the problem formally defined a feasible solution can be introduced. It is guaranteed to be pareto-optimal (it is not possible to increase the utility of one team, without decreasing the utility of another one), but not the optimal in the general case, since long computations are required to handle the very large numbers of different shift team combinations that need to be considered (see Formula 10). An obvious solution, in order to achieve pareto-optimality, is to calculate for each shift the utility for each possible team combination, sort it, and take those with the highest utility value. However, doing this dynamically each time for each shift requires many computations and an extended waiting time before seeing a proposed schedule. It can be improved by calculating the utility for all possible team combinations in the department only once, since the coworker preference bears a static character and is not changed often. The complexity of such a computation is $O(s \cdot d \cdot e \cdot n)$ or $O(n^4)$ since n is the largest of these four numbers. When any actor changes their preferences, the recalculations required are of $O(n^3)$. Having constructed the array of utilities it is sorted into order so that teams with a higher utility are preferred over those with a lower one. The *heap sort* is selected to facilitate the process. It is the slowest of the $O(n \lg n)$ sorting algorithms, but unlike the *merge* and *quick sorts* it does not require massive recursion or multiple arrays to work. This makes it the most attractive option for very large data sets of millions of items.

Every time teams for a shift are built, it is now sufficient to process the sorted array in descending order of team utilities until all the senior physicians (in case of building advanced teams) or assistant doctors (in case of building teams for less complex treatments) are assigned. The heuristic checks if all of the team members considered have a duty in the shift and are not yet members of a team. If this is the case, the actors are assigned to work in one team, since no other combination of available actors can produce a higher current team utility value (remembering that the array is sorted). This yields a pareto-optimal solution. In the worst case there is a need to go through all of the elements in the array.

Furthermore, the solution obtained can be improved with the help of a modified simulated annealing heuristic. It can often be the case that it is not optimal in the sense, that another combination of teams may exist, so that the total shift utility is higher, than that achieved with the pareto-optimal algorithm.

The maximum value of the team utility function will always be less than the product of the greatest team weight and the number of teams that have to be build ($|D| = d$). However, a better approximation can be reached taking the sum of the first greatest d elements of the sorted teams weight array. Even this yields a value that is not less than the optimal solution (since these d teams are often not disjoint). It serves as a probabilistic termination criteria if the obtained solution is close enough to the calculated bound. The heuristic itself starts on the i-th iteration with the $2i$-th element of the descending sorted array of the actors present in the shift, and moves at first upwards, then downwards from the $2i$-th element, checks whether the current team candidate members are not yet assigned for a team, and in the case of positive outcome, selects them. At the same time the utility function is calculated. If the current utility of the sum of d teams is greater than the known maximum, then the current teams assignment is chosen as a candidate for the shift assignment and it is continued with the next iteration. If no improvement is achieved, the temperature value is decreased.

2.3 Preference-Based Task Assignment to Teams

After the preference-based resource allocation (planning phase, building teams) it is now possible to proceed with scheduling. At first the problem analysis will be given, afterwards the heuristic will be discussed.

Problem Analysis. The problem of scheduling we are faced with can be formulated as follows: given a set of v teams, that have duty in the shift x as well as the set of q tasks, that are planned for that shift. Each task has an approximate duration (since it is often not possible to forecast how long an operation will take due to non-determenistic workflow), contributing to the uncertainty degree of the schedules generated. It is also characterized by its priority in order to distinguish between emergency and regular cases. The main component of each task is a patient, that is characterized by insurance (private or governmental), preferred treatment time as well as a doctor preference. Each team is restricted within the shift by the maximum working time allowed without a break and a minimum break duration as stated in hard-constraints. It is also characterized by the time it finishes its last task as currently scheduled. A graphic representation of the task scheduling problem is captured in Figure 1. The goal is to assign those q tasks to v teams in such a way that no hard-constraint is violated and the objective function gets the highest value.

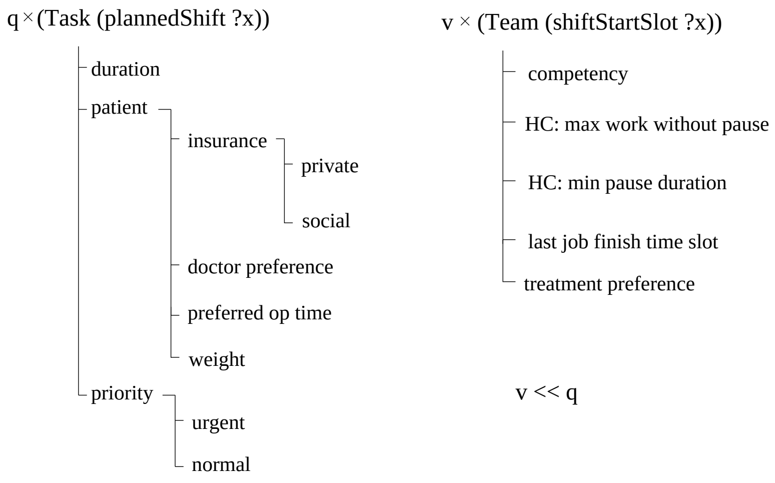

Fig. 1. Task Assignment Problem

The scheduling problem is defined in the terms of the Graham's notation [12]:

Constraint / Preference	Type	Character
max slots without pause	hard-constraint	static
min pause duration	hard-constraint	static
patient doctor preference	preference	static
patient preferred treatment time	preference	dynamic
doctor treatment preference	preference	static
team competency	hard-constraint	static
task priority	hard-constraint	dynamic
patient insurance	soft-constraint	static
shift end time	hard-constraint	static

Table 4. Constraints for Tasks Scheduling

$$Rm \mid r_j, M_j, brkdwn \mid \theta_1 \sum w_j T_j + \theta_2 \sum \hat{w}_j Z_j$$

The first element of the triple - machine environment - refers in the case of the operation theatre scheduling to the team environment. There is no exact environment notation to express the situation faced with in the team scheduling. That is why the most suitable environment is taken with the assumption noted below. *Rm* stands for unrelated machines in parallel. Usually it is used if the machines have varying speeds and that speed of depends on the job being executed. However, the speed of performing an operation is not the selection criterion for the teams, and that is why it does not play any role here. Job j requires a single operation and may be processed on any of the machines belonging subset M_j, however, different machines (teams) yield different values regarding optimality criterion due to the needs of taking into account team and patient preferences. The processing restrictions include release dates (r_j), machine eligibility restrictions (M_j), and breakdowns (*brkdwn*). The first means that the job j can not start before its release date r_j. So, for example, patients, planned to be operated on some specific date, can not be scheduled before that date. Machine eligibility restrictions mean that not all m machines (teams) are capable of processing job j (some specific kind of operation). The set M_j denotes the set of machines that could process the job j. Breakdowns imply that machines (teams) are not continuously available (due to e.g. pauses, times for operation room preparation). As the objective to be minimized is the total weighted tardiness ($\sum w_j T_j$) plus the total weighted team (treatment type) and patient (senior physician) dissatisfaction ($\sum \hat{w}_j Z_j$) chosen, since it conforms the requirements of the operation completion times and team's and patient's wishes in the hospital. The composite objective includes also weights θ_1 and θ_2 for each of the two sub-objectives. So the goal is to minimize the due date violations, where weight w_j may be used to specify the priority of the job e.g. urgent, normal, etc., as well as $\hat{w}_j$ to specify the importance of taking into account wishes of corresponding actors.

After having formally defined the problem, it is now possible to define its complexity. It is made with the help of reduction to the *3-partition problem* [5]. The 3-partition problem is well known to be strongly NP-hard [12]. Strongness means that the problem cannot be optimally solved even by an algorithm with a pseudo polynomial complexity. The proof idea can easily be derived from Pinedo [12].

Since even a simplified scheduling problem with only one machine (team) without any preferences and room considerations is NP-hard, the more general problem with many teams with preferences for treatments, coworkers and time consideration is therefore not less difficult.

Heuristic for Preference-Based Scheduling of Tasks. This heuristic selects tasks that are planned for the shift considered and schedules first the urgent ones, and then the regular ones. Scheduling the urgent tasks tasks before the regular ones matches the normal prioritization in a hospital department. The schedule is developed with the help of *dispatching rules.*

The dispatching rule determines which task should be scheduled as next. Every time a team becomes free and the time since the last index update exceeds some Δ, the ranking index is recomputed for each remaining job. Their list is then sorted and the jobs with the highest ranking index are processed first. This ranking index is a function of the time t, at which a team with the earliest last job finish time becomes free, as well a the patient weight w_j, task processing time p_j, task urgency u_j, and the due date d_j of the remaining jobs. The index is based on [12] and defined as

$$I_j(t) = \frac{w_j \cdot i_j}{p_j} \cdot e^{-\frac{d_j - p_j - t}{K\bar{p}}} \cdot e^{u_j} \tag{11}$$

where

w_j is the patient's weight, that is assigned to the task j;
i_j is the insurance type of the patient from the job j;
p_j is the task j processing time (duration);
d_j is the due date - time, until the job has to be done;
t is the time, machine can begin processing at;
K is a scaling parameter;
$\bar{p}$ is the average of processing times of the remaining jobs;
u_j is the urgency of task j.

The first part of the index ($\frac{w_j \cdot i_j}{p_j}$) determines the price that a patient is paying for one slot of time for his task. The second two parts of it have exponential character and represent the task urgency and the slack influence on the index. K is the *scaling parameter* that can be found empirically and determines the influence of the first exponent on the index function. The smaller K is, the higher is the influence of the first exponent on the whole index. Sometimes K is also called *lookahead parameter.* The *slack* of the job j is the time left before the latest time point the job should be started, in order to do not exceed the due date. It is less than zero if the job can no longer be finished before the due date. This means that the smaller the value, the greater the due date violation. If the slack is positive, the first exponent decreases the value of the index function. Task urgency u_j can be e.g. equal zero for regular tasks, have value greater than zero for urgent ones, having the greater value the more urgent concrete task is.

After the weight calculation for tasks planned for the considered shift is completed, the task list is sorted in the descending order of the weight. In this order the tasks are further processed so that those with a higher weight get scheduled first. The task under consideration is tested in order to determine whether the patient has specified the preferences for the operation chief (senior physician in case of teams having four actors, assistant doctor in case of smaller teams consisting of three actors). If it is the case, we check whether the physicians as specified by the patient preference are present on the shift. Those present are then further categorized into sets with preference "willingly", respectively "unwillingly" (all other belong to "undecided"). Further, each of the sets is sorted in the ascending order of the teams' last task finish time and stored to an ordered list. This contributes to the attempt to schedule the currently less occupied teams within a preference value first. Having finished sorting, the lists $\mathcal{L}_w$, $\mathcal{L}_u$, and $\mathcal{L}_{\tilde{w}}$ are concatenated to the list $\mathcal{L}$ in the following order of preference value:

$$\mathcal{L} = \mathcal{CON}\{\mathcal{L}_w, \mathcal{L}_u, \mathcal{L}_{\tilde{w}}\}.$$

If the current task to be scheduled is an urgent one, the earliest possible starting time is calculated without regard for the specified preference. Preferred teams are only then considered if their starting time deviated from it by not more than a small value ε since these tasks have to be executed as soon as possible:

$$currentTask_{ts} - earliestPossible_{ts} < \varepsilon, \qquad \varepsilon > 0. \tag{12}$$

For regular tasks the physician preference has more influence than with the urgent ones, since it is more important to take this preference into account, even if the task will be scheduled later but still within the shift.

Next we process the ordered team list and select a team for the job. An attempt is made to schedule the team for the next possible time. If this fails because the team needs to take a break, the break is scheduled and we retry. If it fails for the second time, the next team is considered. If the list is processed to the end and if no team was chosen - the dialog-based scheduling mechanism has to take over.

In case of success, the selected team is assigned to the appropriate position in the list of shift teams, to keep it in ascending order of the finishing time of the last task. Scheduling proceeds with the next job.

We introduce a function of the following arguments to handle the treatment preferences of the team:

– doctor preference of the patient;
– last task finish time slot of the team;
– treatment preference of the team.

During the task scheduling, the function value is calculated for each team. The teams are considered in order of the returned function value, rather than being divided into three groups with patient preference values from $\mathcal{D}$ and sorting regarding the last task finish time within the group.

Discussion. An important feature of the task scheduling mechanism proposed is that it allows rescheduling on demand. The dynamic character of job arrivals in a hospital environment causes disturbances to pre-existing schedules, making reactive scheduling necessary. The plan is being adapted to new situations as the changes occur. However, the emphasis is to keep as much as possible of the existing plan untouched. Rescheduling also takes place when an urgent task is added to the system. In this case all tasks, that have already begun are kept unchanged. In contrast, all un-started ones have their status changed to *unscheduled* and are re-proceeded by the scheduler. Following, further possible improvements can be considered for the job assignment:

– if a hard-constraint is violated due to the maximum work time without a pause, it is possible to search for a shorter task to fill the gap instead of recommending a pause right away. This helps to minimize unneeded pauses;
– tasks, to which patients did not specified doctor preferences, can be postponed initially. Instead, the number of tasks with explicit preferences for each team should be computed. Scheduling those tasks without preferences to those teams that have less patients in queue would bring better overall satisfiability of the proposed schedules. However, not specifying the preference should not cause longer waiting times before processing;
– *team rotation* is an important feature for long and complicated operations. In real life some treatments last longer than one team can complete either due to the ending of the shift or due to exceeding the maximum allowed hours without a break. In this case a team "handover" must be performed during the task execution.

The proposed heuristic makes proposals for the task scheduling. However, it is up to the human scheduler whether to accept or to reject it. The goal is to minimize the manual rescheduling so that the proposed schedule is changed, if at all, only slightly. Tasks, that were not able to be scheduled automatically must be proceeded by the human anyway.

2.4 Room Assignment

The last stage in operating theatre scheduling covers the assignment of operating rooms. This approach is subject to the condition that there are enough rooms available in that their number at least equals the number of teams in the shift and furthermore, the room, assigned to the team, has all the equipment and facilities that the team may need for executing its tasks. In another case, if the teams are kept only for operations and no rooms are available, personnel resources are wasted. However, teams often pause between consecutive operations. Each of these time gaps can be too small to reserve the room for another task, but some optimization of these time windows is possible, such as maximizing the available gaps between operations already scheduled. It is assumed that the rooms considered for the optimization are all of the same type, so that it does not matter which room is chosen from the proposed set, and it is not vital for the personnel. Of course the hospital management goal is the full utilization of the resources, but at the same time a reserve capacity is required in order to handle emergencies and to deal with the device/room breakdowns. It is assumed that all the team and time assignments are already made. The algorithm chosen to facilitate the process comes from Kandler [8], who proposed it for scheduling in a virtual enterprise. The important feature for the room assignment is that the jobs are not shifted in time but retain their scheduled time slots unchanged.

3 Analysis and Comparison

In this section the proposed heuristics are evaluated in practice. The most interesting criteria are the time needed to produce a valid schedule proposal as well as the quotient of regarded/disregarded wishes of the personnel, and its comparison to existing schedules produced by human actors (here the random assignment is used because this reflects reasonably well the reality in todays world where preferences are not considered). All evaluations were performed on an Intel Pentium 4 CPU 2.40 GHz with 512 MB RAM running SUSE Linux with default kernel version 2.4.21. The experimental setup was built using the multi-agent system JADE 3.2[5] and the rule-based expert system JESS 6.1[6].

3.1 Shift Assignment Evaluation

The measures for shift assignment are made for 20 different values for the personnel size in the department (from 10 to 200, with the step 10), each with four different required staff quotients (specified as a hard-constraint) in the shift. The quotient is 20%, 30%, 40%, 50% of the overall number of personnel.

The number of actors for each position is based on the figures given in Table 5. For example, in case of 200 actors, 20 are senior physicians, 40 are assistant doctors, 60 anaesthetists, and 80 nurses.

quote	qualification
10%	senior physician
20%	assistant doctor
30%	anaesthetist
40%	nurse

Table 5. Personnel Quote of Specified Position

Shift preferences are generated for all actors and shifts as follows: with the probability $\frac{1}{3}$ the preference for the considered shift is generated by an actor. If it has been generated, it is assigned a random value from the domain $\mathcal{D}$ (each with the equal probability of $\frac{1}{3}$).

The number of required personnel usually varies due to different days and shifts. Thus, we distinguish between a shift {*early, late, night*} as well as the day of the week (regular working day, weekend or holiday). Furthermore, for each of the days and qualifications, hard-constraints regarding the required staff number are usually set flexibly, e.g. between 8 and 10. That is, there must be at least 8, at most 10 actors with some qualification in the shift. Following the heuristic, the maximum number is only taken if all wish to work in the shift, no other hard-constraint is violated and the contractual work hours are not yet exceeded (soft-constraint). Otherwise, the lower number of personnel is selected. For the simulation, however, the upper and lower bound are kept identical.

[5] Java Agent Development Framework, *http : //jade.tilabs.it/*

[6] Java Expert System Shell, *http : //herzberg.ca.sandia.gov/jess/*

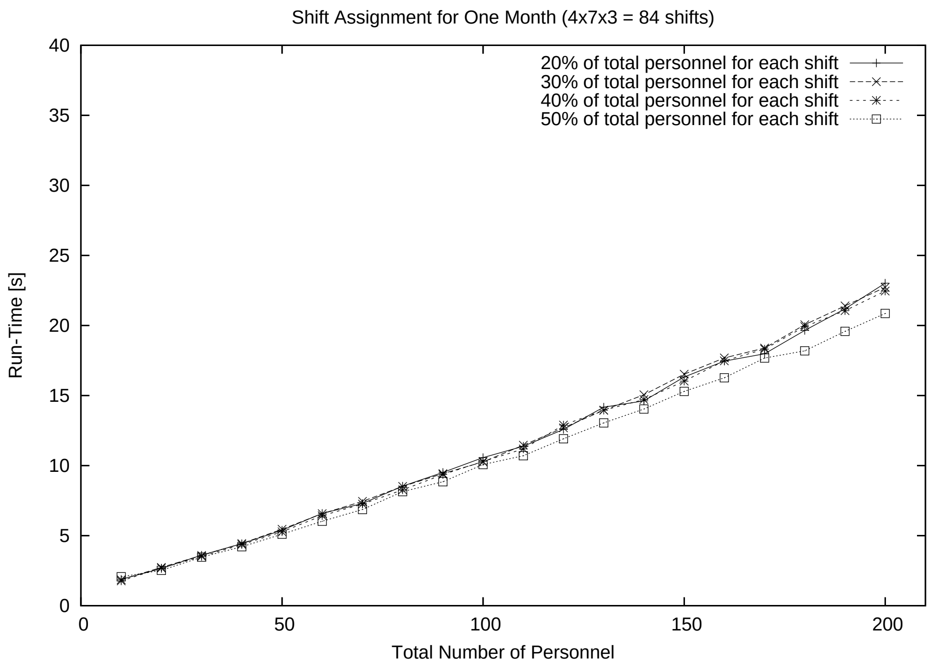

Fig. 2. Shift Assignment Run-Time

For each qualification the maximum duty duration as well as the minimum inter-duty pause hard-constraints were set to be equal to 16 hours. That is, it is not allowed to work more then 16 hours consecutively, and between 2 shifts must not be less than 16 hours. The soft-constraint that represents the contractual work hours is set to 40 hours per week for each employee.

Shift assignment is performed for one month (4 weeks, each with 7 days, each with 3 shifts that results in 84 shifts). The total number of staff in the hospital is chosen as a parameter. Another parameter is the number of required personnel per shift (four different configurations). The time required to produce such a schedule grows linearly with the number of personnel, and the assignment for one month takes less than a half of minute of computing time for 200 actors as captured in Figure 2.

The important criterion for acceptance of the proposed plans by human actors is the degree to which their individual preferences are taken into account. The presented simulations are made for the personnel of 200 actors selecting 10%, 30%, 50%, and 70% of total number for each shift for the period of one month (84 shifts). Figure 3 shows the percentage of preferences met in each shift. For a shift considered, the total number of preferences is calculated as the number of actors that have specified a preference with a value from {*willingly, unwillingly*} for the shift. It is important, that only those actors are playing a

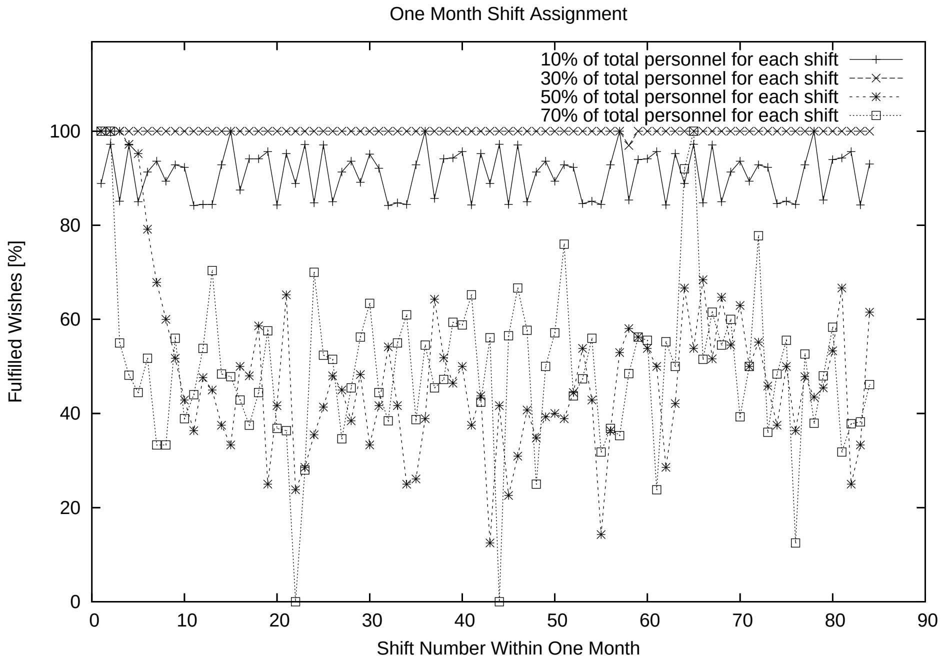

Fig. 3. Shift Assignment Fulfilled Wishes Quota

role here that are allowed to work in the shift (no hard-constraint violation). A preference is treated as respected in case of

– selecting an actor with preference value *willingly*;
– not selecting an actor with preference value *unwillingly*.

For 30% of total personnel in each shift all wishes are almost always fulfilled. In case of selecting 10% usually none of the unwillingly actors is scheduled for work, but not all who would like to work are selected, causing a slightly lower quotient of the preferences met compared to the previous case. Choosing the selection rate to 50% and 70% respectively of total personnel in each shift causes all the willing actors to be selected as well as some undecided and, eventually some unwilling too. Furthermore, selecting these numbers of staff causes violations of hard-constraints in some shifts due to the days off need. This leads to understaffing in these cases. Beyond that, often those who would like to work in the shift can't be even considered due to the above mentioned need for time off. Therefore actors with other preference values are selected. This contributes to the variation of the percentage of the preferences met. The amplitude of this variation is greater when selecting 70% of the entire staff. Nevertheless, the overall quote of preferences met is high and the proposed plans are likely to be accepted by the personnel, as compare to those manual schedules produced by human actors that usually disregard most wishes.

3.2 Team Assignment Evaluation

We need to initialize the team utilities before we can begin with team building. The initialization is done for the teams consisting of four and three actors. Due to the fact that the number of team combinations grows exponentially with the number of personnel, this is the most time consuming procedure in the algorithms. However, in a real life scenario the initialization must be performed only once. The changes are only needed if one decides to change one of his coworker preferences. In this case only one dimension of the utility array has to be recalculated. In Figure 4 the initialization and sorting times for teams consisting of four actors are captured.

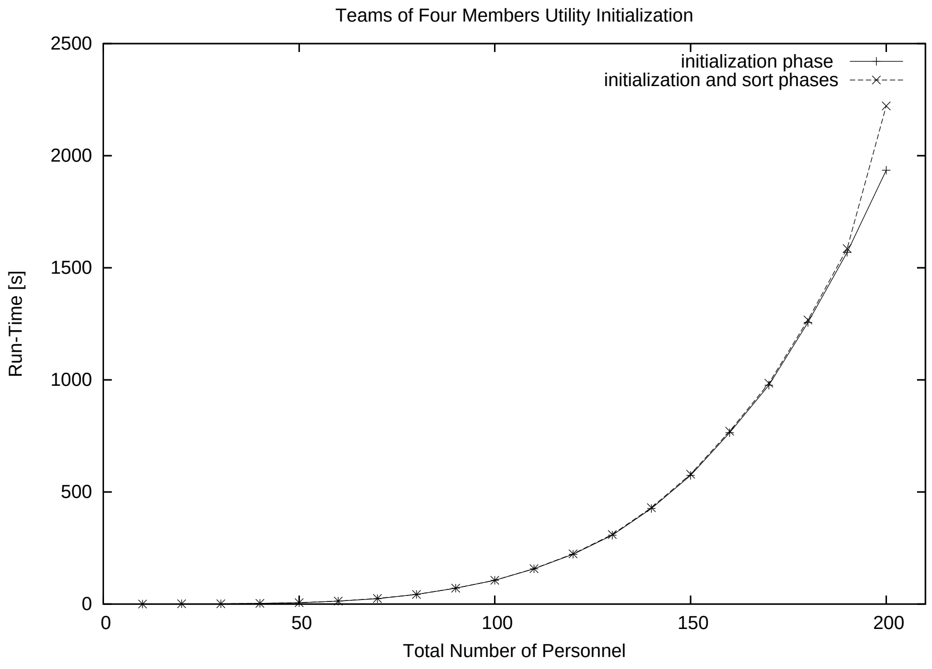

Fig. 4. Teams of Four Members Utility Initialization Run-Time

As can be seen from the figure, most time is spent on the initialization of the the team utilities and not on sorting. It is due to the need to query the knowledge base for each of the possible team combinations for the preference values of each actor regarding his coworkers. In case of three actors these are $3 \cdot 2 = 6$ queries and $4 \cdot 3 = 12$ queries for teams of four actors. For teams of four members, however, a significant deviation of the initialization time from initialization and sorting time is visible if the personnel size is 200 (the number of different team combinations is then $20 \cdot 40 \cdot 60 \cdot 80 = 3.840.000$). Repeated simulations provided the same outcome. Detailed analysis has shown that with this number of combinations swapping occurred.

Simulation was performed in Java, instructing the Java Virtual Machine to initially assign 72MB allowing up to 1GB. The total number of staff was chosen analogously to the case of shift assignment. Coworker preferences are generated as follows: about 50% of the actors are selected randomly to specify a coworker preference. Each selected actor specifies preferences to roughly 20% of his random coworkers. These preferences have values either *willingly* or *unwillingly* each with the probability of $\frac{1}{2}$. For team utilities calculation we used the weight values from Table 6. For a ward with 200 actors the initialization and sorting time of the

weight	qualification
100	senior physician
80	assistant doctor
60	anaesthetist
40	nurse

Table 6. Team Assignment Personnel Weight

team utility array for teams of 4 actors takes less then 40 minutes, for teams of 3 actors less then one and a half minute.

In contrast to the team initialization and sorting that has to be performed only once, team building (assignment) is executed for each shift. In Figure 5 the time needed to build teams of four actors for one week is captured. The worst case occurs if there is a need to go down to the last element in the list of teams, sorted in descending order of team utilities. The other measures are made for the actual time needed to assign teams, selecting 20%, 30%, 40%, and 50% of the total personnel in each shift. Interesting questions that arise analyzing the graphs are why the actual measures are well below the worst case, and why it takes longer to assign less (selecting 20% of total) than more teams (50%). The importance of a senior physician (represented by the weight) is clearly greater than that of the other team members, leading to positioning team configurations with his favored coworkers in the beginning of the sorted team array. The more personnel is present in the shift, the higher the chances that the people he prefers and that prefer him are also in the shift. The search ends sooner due to the availability of teams that are located in the beginning of the array. Even increasing the number of teams (the number of senior physicians increases as well) does not contribute to longer run-times since in the beginning senior physicians are iterated in the array due to their great influence.

Next, the question arises how much improvement the proposed team assignment heuristic brings. It is not very obvious what kind of data should be taken as a reference to compare with. On the one hand it is important to satisfy as many coworker wishes as possible. On the other hand, the hierarchy has to be taken into account, because the proposed team building heuristic may not be

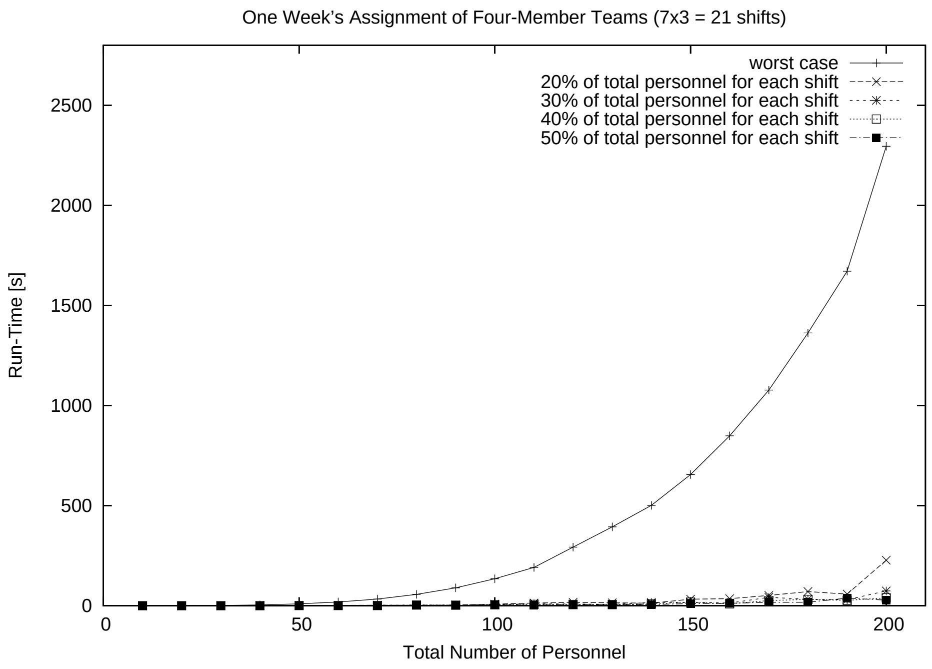

Fig. 5. Assignment Time of Four-Member Teams (With Worst Case)

accepted in the ward if the wishes of e.g. two nurses have always more influence on the decision than that of one senior physician. For demonstration purposes, however, in order to be able to compare the number of preferences taken into account in the teams, the weight of each preference, independently of the position and qualification of an actor, is set to be equal one:

$$\forall a \in S \cup D \cup E \cup N, \qquad g(a) = 1.$$

As a reference for comparison, randomly built teams are chosen. Measures are made for one week (21 shifts) and show the sum of team utilities within the shift for the heuristic and random team building. The average values as well as the number of teams in each shift are also captured in the diagram. Figure 6 shows the simulation results selecting 40% of total personnel in each shift for building teams of four actors, with the size of the ward equal to 200 actors.

3.3 Task Assignment Evaluation

Last, but not least, job assignment is performed and evaluated for different numbers of staff in the shift, for the period of one week. Tasks are generated in such a way, that there are more jobs than the teams can perform in each of the

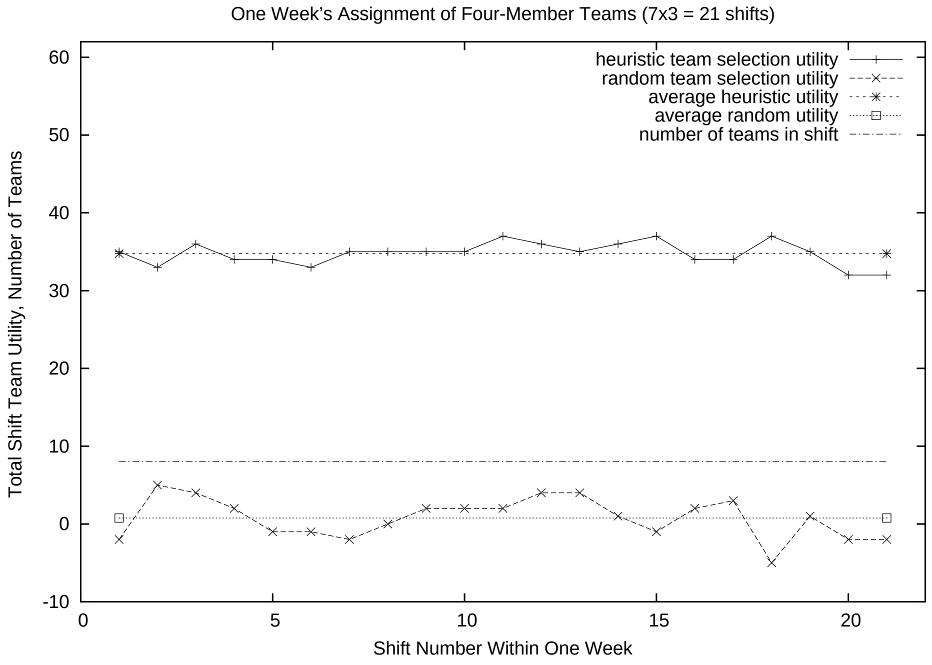

Fig. 6. Total Utility of Four-Member Teams (40% of staff for each shift) Comparison of Heuristic and Random Assignment

shifts. In case of dealing with teams consisting of four actors, there are 10% of senior physicians in the department. Each shift has 32 slots (15 minutes one slot, eight hours per shift), each operation or treatment lasts at least one time slot (however, the duration of the operation is randomly generated from one to five time slots long). The product of the senior physician number and the number of slots within the shift gives the number of patients in the shift, for each of whom a random task is generated. With the probability 0.2 the task is urgent. Measures for the job assignment are captured in Figure 7 and performed for different numbers of staff and shift selection quotient. Each team is restricted by hard-constraints to operate a maximum of 4 slots consecutively (one hour) and has to take at least 4 slots off afterwards. The more teams in the shift, the longer the task assignment lasts. The higher number of possibilities for the teams, task can be proceeded at, causes the increase in required calculations. For 200 personnel in the ward the job assignment for one week (21 shifts) takes less than 6 minutes with the proposed heuristic.

4 Discussion

All the techniques described are heuristics that do not guarantee to find an optimal result. Instead, they aim to find a reasonably good solution in a relatively

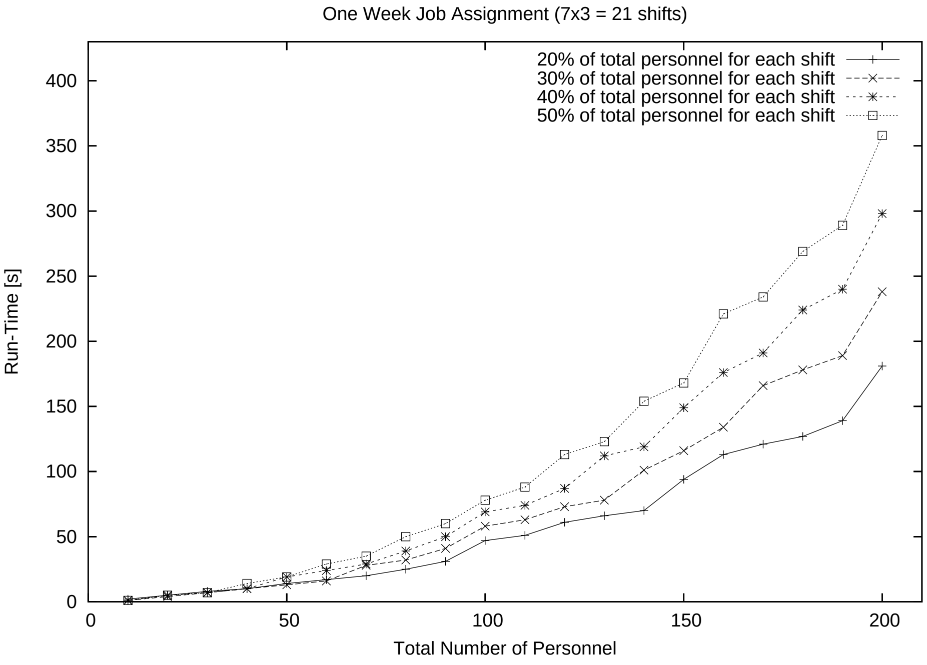

Fig. 7. Job Assignment Time

short time. The presented algorithms facilitate shift assignment (choosing personnel to work in a shift), team assignment, task assignment as well as room assignment. An important attribute of the heuristics is the consideration of the preferences of the involved actors as well as fairness due to the introduced weight functions.

Different heuristics could be developed in order to facilitate the scheduling process. For example, another possible approach for team building could be to use the features of an expert system by defining queries in order to find tuples of team member that match with a given satisfiability value and are on duty in the shift. The disadvantage of this approach is the impossibility to directly search (match) for the team with the highest utility, since only preference values have an influence on such a kind of pattern, not the weight an actor places on the concrete preference. The number of queries required to fetch all possible team preference combinations would be n^m, where n is the number of team members, and m is the number of possible preference values.

5 Conclusion

Staff timetables in medical departments are subject to lots of constraints, restrictions, and preferences [3]. Scheduling of hospital personnel is particularly challenging because of different staffing needs on different days and shifts, uncertainty between the offered capacity and the true demand. Furthermore, it is impossible to predefine a treatment's workflow. As emergency cases occur (causing disturbance in existing schedules), there is a need of adaptation to situation changes. Due to the complexity and uncertainty the applicability of traditional (operations research and AI) methods from industrial scheduling to the operation theatres scheduling is problematic [11, 2, 10]. Usually a specialized person is in charge of this task (medical director). Yet, this often does not takes into account preferences of individual actors.

We have split the original problem into sub-problems and provided a preference-based adaptive heuristics for each of them. The system makes a schedule proposal and it is up to the responsible human actor either to accept, accept parts of the proposition, or to reject the schedule. In this paper we show, that the required time for shift, team, job and room assignment is within acceptable ranges for a real-world ward size [10]. The comparison of the heuristic approach to the random assignment was given. This allows to conclude that the proposed algorithms bring a substantial improvement regarding the number of fulfilled wishes of the actors, while planning and scheduling, and helps to save expensive human resources that are currently used in hospitals for the manual scheduling process. However, the preferences of the involved actors were randomly generated. It can often be the case that e.g. some actors are preferred by most others. Heuristic behavior with such a preference distribution is not analyzed yet. Evaluations in a real-world setting would be of a great interest and will be made in cooperation with the university hospital.

Acknowledgments

The authors would like to thank Sumedha Widyadharma for helping to develop the scheduler GUI, Martin Beer, Kai Jakobs, and the anonymous reviewers for their valuable comments and corrections.

References

1. P. Baptiste, C. Le Pape, and W. Nuijten. *Constraint-Based Scheduling: Applying Constraint Programming to Scheduling Problems.* Kluwer Academic Publishers Group, PO Box 322, 3300 AH Dordrecht, The Netherlands, 2001.
2. M. Becker, K.-H. Krempels, M. Navarro, and A. Panchenko. Agent Based Scheduling of Operation Theatres. *EU-LAT eHealth Workshop, Cuernavaca, Mexico*, pages 220–227, December 2003.
3. Edmund Kieran Burke, Patrick De Causmaecker, and Greet Vanden Berghe. Novel Meta-heuristic Approaches to Nurse Rostering Problems in Belgian Hospitals. In *Handbook of Scheduling: Algorithms, Models and Performance Analysis*, pages 44.1–44.18. A CRC Press Company, 2004.
4. Stefan J. Darmoni, Alain Fajner, Nathalie Mahe, Arnaud Leforestier, Marc Vondracek, Olivier Stelian, and Michel Baldenweck. Horoplan: computer-assisted nurse scheduling using constraint based programming. In *Journal of the Society for Health Systems*, volume 5, pages 41–54, 1995.

5. Michael R. Garey and David S. Johnson. *Computers and intractability: A Guide to the theory of NP-completeness.* A Series of Books in the Mathematical Sciences. Freeman and Co., San Francisco, California, USA, 1979.
6. Y. Guo, A. Lim, B. Rodrigues, and Y. Zhu. Heuristics for a Brokering Set Packing Problem. In *Proceedings of 8th International Symposium on Artificial Intelligence and Mathematics in Ft. Lauderdale, Florida*, 2004.
7. D. Hochbaum. *Approximation Algorithms for NP-hard problems.* PWS Publishing Company, 20 Park Plaza, Boston, MA 02116-4324, 1995.
8. Florian Kandler. Scheduling in a Virtual Enterprise in the Service Sector. In Juergen Sauer, editor, *Proceedings of the KI-2001 Workshop "AI in Planning, Scheduling, Configuration and Design", PuK 2001 Vienna, Austria, September 18, 2001*, pages 32–42, 2001.
9. Lars Kragelund and Torben Kabel. *Employee Timetabling: An Emperical Study of Solving Real Life Multi-Objective Combinatorial Optimization Problems by means of Constraint-Based Heuristic Search Methods.* Master thesis in cs, 1998.
10. Andriy Panchenko. *Preference-Based Scheduling of Operation Theaters.* Master Thesis in Computer Science, Department of Computer Science IV, RWTH Aachen University, Germany, 2005.
11. T.O. Paulussen, N.R. Jennings, K.S. Decker, and A. Heinzl. Distributed Patient Scheduling in Hospitals. In *Proceedings of 18th International Conference on AI, Acapulco, Mexico, 09.-15. August*, 2003.
12. M. Pinedo. *Scheduling: Theory, Algorithms, and Systems.* Prentice Hall, Upper Saddle River, New Jersey 07458, 2002.
13. D.G. Rajpathak. Intelligent Scheduling - A Literature Review. Technical Report KMI-TR-119, Knowledge Media Institute, The Open University, Walton Hall, Milton Keynes, MK7 6AA, UK, August 2001.
14. J. Sauer. *Multi-Site Scheduling: Hierarchisch koordinierte Ablaufplannung auf mehreren Ebenen.* PhD thesis, Fachbereich Informatik, Universität Oldenburg, 2002.