

---

# PACT-97 Contribution

---

ConPlan/SIEDAplan\*: Personnel Assignment as a Problem  
of Hierarchical Constraint Satisfaction

Harald Meyer auf'm Hofe



**Deutsches Forschungsinstitut für Künstliche Intelligenz**  
**Postfach 2080**  
**D-67608 Kaiserslautern**

\*The ConPlan project has been supported by the foundation "Rheinland-Pfalz für Innovation" under grant number 836-38 62 61/98.

## **Abstract**

The ConPlan project conducted by the German Research Center for Artificial Intelligence in collaboration with the SIEDA Software house aimed at representing and solving nurse scheduling problems as a problem of optimizing constraint satisfaction. Shifts are assigned to the personnel complying with legal regulations, avoiding over-time work as far as possible and meeting as many additional requirements as possible. These requirements are encoded in a *hierarchical constraint satisfaction problem (HCSP)* comprising 600 to 800 variables. This paper explains on the one hand how nurse scheduling has been represented as a problem of optimizing consistency with constraints. On the other hand, the used search algorithm is described.

# 1 Introduction

The ConPlan project conducted by the German Research Center for Artificial Intelligence aimed at representing and solving nurse scheduling problems by generic techniques of partial constraint satisfaction. As a result a C++ library has been developed providing implementations of various search algorithms and constraint propagation techniques. This library is a part of the SIEDAplan nurse scheduling system that has been implemented by the SIEDA software house in Kaiserslautern and is currently tested at the DRK hospital Neuwied. Working shifts are assigned to each nurse on each day of a certain period of time. A typical problem comprises 600 to 800 assignments that have to meet several requirements like

- legal regulations,
- optimized personnel costs,
- flexibility with respect to the actual expenditure of work,
- consideration of special qualities,
- management of vacation and absence,
- consideration of employee's requests,
- preference of working time models, that are common sequences of working shifts.

These requirements can be represented by constraints.

The problem has two characteristics:

1. It is hardly possible to fulfill all the requirements on a nurse schedule simultaneously. Conflicting requirements have to be distinguished due to their importance. While compliance with legal regulations is required, the consideration of employee's requests is optional. Additionally, explicit optimization tasks are part of the problem.

Consequently, a solution is not necessarily consistent with all requirements but satisfies them as good as possible.

2. The satisfiability of requirements strongly depends on parameters like the contracts of the employees, the employees' working time balance, and the schedule of approved vacation. Hospitals are especially interested in opportunities to flexibly react on the actual expenditure of work and to avoid expensive over-time work.

Consequently, it is impossible to determine the most important and satisfiable requirements in advance.

Hence, the opportunity is needed to represent constraints of distinguished importance and to compute a solution complying with the set of most important constraints.

Usually, importance of constraints is stated by a real number representing either a *weight* [5] or a *priority*, a possibility value or a fuzzy membership value [3]. In systems of weighted constraints the problem solver tries to satisfy constraints with a maximal sum of weights. Thus, the simultaneous satisfaction of many less important constraints can be preferred to the satisfaction of a single but more important constraint. In contrast, constraints of larger *priority* are always more important than all constraints of lower *priority* together. Such attributes presuppose “... a *commensurability between preference levels pertaining to different constraints. In other words the user who specifies the constraints must describe them by means of a unique preference scale*” [3]. In the nurse scheduling

problem one needs both methods of combining preference levels of different constraints: weights as a cumulative measure of importance as well as priorities. Hence, both aspects need to be integrated in an understandable way in order to make problem representation as easy as possible. To achieve this goal, ideas of *hierarchical constraint logic programming (HCLP)* [2] have been translated into the context of constraint satisfaction avoiding some disadvantages of the HCLP scheme. The nurse scheduling problem is represented in the resulting scheme — as a *hierarchical constraint satisfaction problem (HCSP)*.

Such problems of *partial constraint satisfaction* may be solved searching either exhaustively by enhanced versions of the *branch&bound* algorithm [5, 11, 10] or locally [17]. The effort of exhaustive search grows in worst cases exponentially with the number of the variables to be labeled with shifts, whereas local search is not sound.

Common constraint logic programming approaches to nurse scheduling base on exhaustive search. None of these approaches manages soft constraints in order to represent optional requirements or optimization criteria. Typically, sound and complete techniques like arc-consistency processing, and numerous application specific heuristics are coupled to achieve an acceptable latency of the system on *most* problem instances. Latency possibly varies strongly between different instances of the problem because of tree search's exponential complexity. Lately, *neighborhood interchangeability* [4] of shifts has been used to simplify nurse scheduling problems [18]. The effect of this method is to unify shifts playing a similar role in the scheduling process during search — in this case several types of idle shifts. This reformulation of the problem is sound and complete but the problem remains to be of exponentially complexity. In contrast, local search algorithms have the advantage of being able to return a solution at any time — the result is of course not necessarily of a reasonable quality. Thus, these algorithms are especially appropriate if some dialog with users is desired e.g. to request some hints for solving the problem.

This paper will firstly consider the representation of nurse scheduling as a HCSP. Then, the problem specific local search procedure is described that has been used to solve the problem. Finally, some findings and directions of further research and development are sketched.

## 2 Representation

The *constraint satisfaction problem (CSP)* is given by a set of variables  $V = \{v_1, \dots, v_n\}$ , each associated with a finite domain  $D$ , and a finite set  $C$  of constraints restricting the assignment of values to some variables. The task is to find a labeling  $\{v_i \leftarrow d_i \mid 1 \leq i \leq n\}$  of all variables  $v_i$  with values  $d_i \in D$  that is consistent with all constraints in  $C$ .

To represent nurse scheduling as CSP one has at first to identify the constraint variables and their domains comprising the values the variable can be labeled with. In our representation a constraint variable is generated for each nurse on each day that is considered by the schedule. Each variable has to be labeled with the shift the nurse has to serve at that particular day. At the moment, most hospitals still use a three-shift model with only one early-morning-shift F1, one day-turn S1, and one night-shift N1. Personnel scheduling is typically done by hand. Due to cost pressure and the deficiency of qualified and experienced personnel it has been recently recognized that working times must be much more flexible and efficient. A reasonable and promising solution seems to be the introduction of additional overlapping shifts (e.g. six- or nine-shift model) with less working hours. Hence, the system is required to integrate new types of shifts flexibly. These new shifts can be scheduled in a way, that the overlapping hours are during very work-intensive periods. Additionally, some kinds of idle shifts (— and \*) as well as holidays UL have to be scheduled. Each of these particular shifts

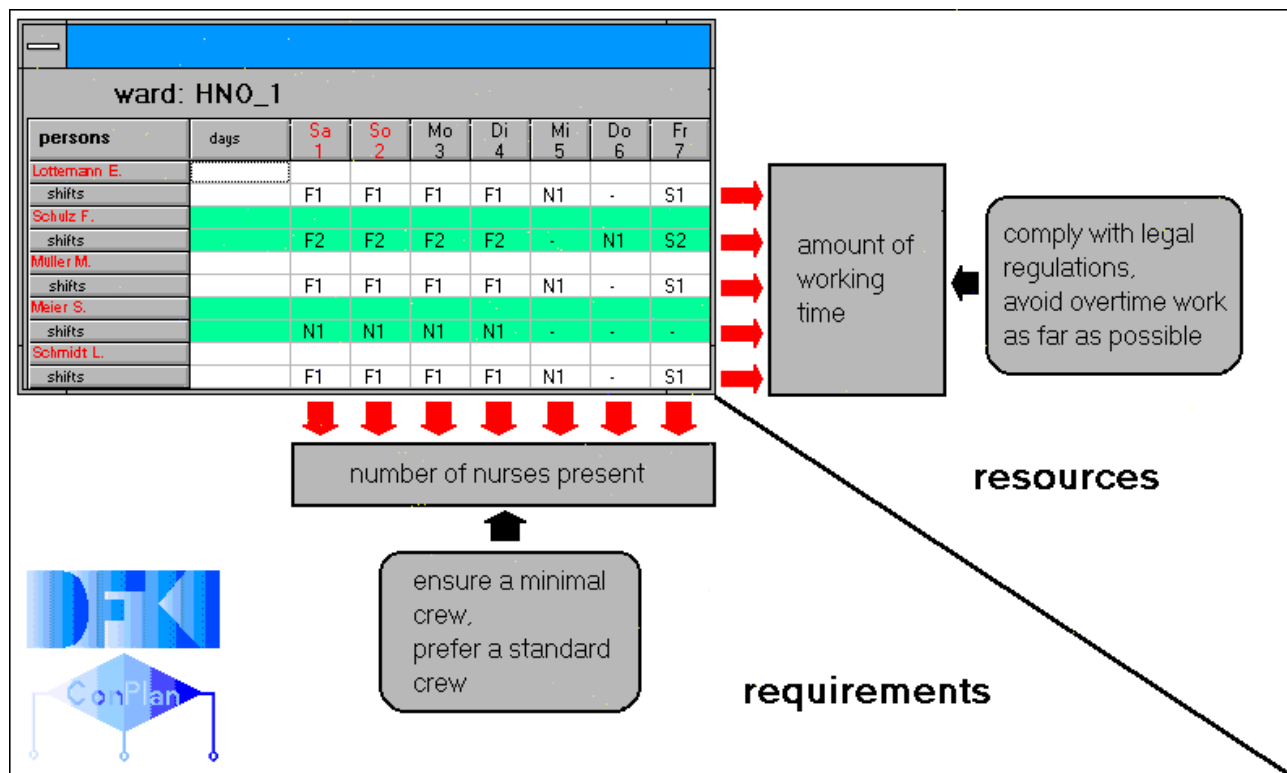


Figure 1: Schema of the requirements in nurse scheduling.

differ in start or end time, effect on the working time balance and costs. For a ward with a crew of 15 to 25 nurses and a planning period of one month this representation is a constraint problem of 600 to 800 variables each having a domain of around 10 or more shifts. About  $10^{600}$  schedules are *a priori* possible solutions of the problem. Figure 1 gives an impression of the representation showing the cut of a schedule.

## 2.1 Constraints

These variables are connected by constraints representing demands on the schedule. These constraints are also shown in figure 1. The variables in each row of the schedule are connected by constraints concerning the number of nurses to be present at the ward at a particular time. This number can vary over the time, and both — a minimal and a preferred attendance of crew members at the ward — can be specified. The preferred size of a crew should not be exceeded. Contrarily, the variables in each column are connected by constraints concerning a particular employee. The most important constraints of this group involve the variables of a complete column and enforce a balance between the working hours to be served due to the employment contract and the scheduled working time. However, over-time work can not always be avoided completely. The less over-time work is needed to achieve an acceptable crew attendance the better is the schedule. Additional constraints affect variables in a column of a schedule and concern obligational and preferred breaks between consecutive shifts, and working time models — some preferred sequences of shifts defined for each employee. Working time models are traditionally used by the personnel department to control the deployment of particular employees. The schedule of each employee should be as similar as possible to one of its working time models. Additionally, unary constraints of the form “ $shiftOfAt(person, day) = UL$ ” represent approved holidays. Employee’s requests are translated into constraints like “ $shiftOfAt(person, day)$

*is requested to end before 6 PM.”*

## 2.2 Preference

Up to now preference on constraints has been treated only superficially. Compulsory and optional demands on a schedule have been distinguished. Of course, the optional demands are of different importance. As mentioned above two basic ways have been suggested to represent such different degrees of preference: Weights [5] and priorities [3]. The satisfaction of a constraint with a large weight can be less preferred than the simultaneous satisfaction of many constraints with smaller weights, whereas a constraint of high priority is always more important than all constraints of lower priority together. Both effects are needed in the nurse scheduling domain:

1. For instance cost reduction is always more important than respecting employee’s requests. Thus, keeping the working time in balance has a higher priority than employee’s requests.
2. Contrary, the schedule should respect *as many* requests of employees as possible. Thus, requests have an additional weight.

This problem can be solved distinguishing the following hierarchies of preference:

**hard:** compliance with legal regulations, and hard working time restrictions,

**1:** guarantee minimal crew,

**2:** management of working time, e.g. reduction of overtime work,

**2:** deploy a crew of preferred standard size,

**3:** compatibility of consecutive shifts, working time models,

**4:** consideration of employee’s requests.

Each constraint of a higher hierarchy level is more important than all constraints of lower levels together. Within each category constraints are rated according to a weight. These hierarchies of preference correspond to the hierarchy levels in *hierarchical logic programming (HCLP)* [2]. However, some disadvantages of this formal framework, mainly concerning the so called inter-hierarchy comparison, should be avoided. A result is the definition of *HCSPs* [10].

### 2.2.1 Formalization

A problem with *soft constraints* involves in addition to the typical elements of a *constraint satisfaction problem* — a set of variables  $V$ , their domains  $D$  and a set  $C$  of constraints between them — a preference ordering among sets of constraints  $\succ$ .

A solution of the problem is a labeling of the variables in  $V$  with values from  $D$  that violates a least preferred set  $C' \subseteq C$  of constraints. A constraint set  $C'$  is preferred to be satisfied rather than constraint set  $C''$  if and only if  $C' \succ C''$ . A solution of a problem with soft constraints is a complete labeling of all variables  $V$  violating a minimal constraint set due to the given preference criterion  $\succ$ . Thus, the meaning of  $C' \succ C''$  can be explained in natural language by: *The constraints in  $C'$  are more important than the ones in  $C''$ .*

Preference orderings usually depend on attributes of the constraints like weights or priorities. For instance, if  $\omega(c)$  denotes the weight of constraint  $c$  then weighted constraints define a preference like

the following:

$$C' \succ_{\omega} C'' \text{ iff } \sum_{c' \in C'} \omega(c') > \sum_{c'' \in C''} \omega(c'').$$

The sum of the weights determines which constraint set is preferred to be satisfied. If  $p(c)$  returns the priority of constraint  $c$  then a system of constraints with priorities defines a preference ordering

$$C' \succ_p C'' \text{ iff } \max\{p(c') \mid c' \in C'\} > \max\{p(c'') \mid c'' \in C''\}.$$

A constraint set is as important as the most important violated constraint.

The HCLP scheme [2] introduced lexical combinations of preference criteria. Several preference criteria are treated one by one in a certain order due to their importance. The most important criterion is considered first. If one of the two sets is preferred according to this criterion, this set will also be preferred due to the whole list. Otherwise, the next criterion is considered. This procedure is repeated until either the importance of both sets can be distinguished or all criteria have been treated but failed. The purpose now is to associate each of these criteria with one of the above mentioned hierarchy levels. Lower hierarchy levels will only be considered, if two constraints can not be distinguished according to more important criteria. Thus, this is exactly the formalism that is needed to represent nurse scheduling as an optimization problem.

This idea can be adopted to the previously introduced formalization of soft constraints. A *hierarchical constraint satisfaction problem (HCSP)* is declared dividing the set of constraints  $C$  into a finite number of hierarchy levels  $C_i$  such that each constraint is in exactly one hierarchy level.  $C_0$  is thought to comprise compulsory constraints. Each hierarchy level  $C_i$  except  $C_0$  is associated with a preference ordering  $\succ_i$  on subsets of  $C_i$ . If  $C' \succ_i C''$  is valid for two sets  $C', C'' \subseteq C_i$ , then  $C'$  is said to be preferred to  $C''$  with respect to hierarchy level  $i$ . Each hierarchy  $H$  is said to be respected by exactly one preference ordering  $\succ_H$  according to the following inductive definition:

$$\begin{aligned} \succ_H &\equiv \succ_H^1 && \text{with} \\ C' \succ_H^i C'' && \text{iff} && C' \cap C_i \succ_i C'' \cap C_i \\ && && \text{or} && \neg(C'' \cap C_i \succ_i C' \cap C_i) \\ && && && \text{and } C' \succ_H^{i+1} C''. \end{aligned}$$

The semantics of hierarchy levels is closely related to constraint priorities. As an illustration assign a priority of  $p(c) := \frac{1}{i+1}$  to each constraint  $c \in C_i$  of level  $i$  in an arbitrary hierarchy  $H$ . Then, obviously

$$C' \succ_p C'' \implies C' \succ_H C''$$

holds true. Hence, a constraint hierarchy can be considered as a system of prioritized constraints with some additional preference relations.

### 2.2.2 Application of the proposed formalism

The initial motivation of constraint hierarchies has been the integration of constraint priorities and constraint weights. The previous section showed that hierarchy levels are related to constraint priorities. In the nurse scheduling application the constraints are divided into hierarchy levels as described previously: e.g.  $C_0$  holds legal regulations and compulsory working time restrictions,  $C_1$  demands the attendance of a minimal crew etc. To refine this hierarchy each constraint  $c$  is mapped to a weight  $\omega(c)$  to determine its preference in its hierarchy level. Two schedules are compared respecting the resulting hierarchy due to the following procedure:

- If a schedule violates compulsory conditions then it is unacceptable. Nothing has to be compared.
- In this hierarchy level, the weights are all 1.0. If one of the schedules ensures the minimal crew at more days than the other schedule, then it is preferred. Otherwise, the next hierarchy level will be considered.
- In this level, the weight of each constraint is related to the distance between the number of working hours an employee is required to serve and the number of scheduled working hours. If the sum of these distances is equal in both schedules the next level will be considered.
- Each optional condition on sequences of shifts served by the same nurse is weighted by 1.0. The more conditions are fulfilled, the better is the schedule. Requests of employees will be considered if the number of violated conditions is equal in both schedules.
- Employees have the opportunity to state a certain number of heavier weighted requests and an arbitrary number of requests with normal weights.

As the rating of schedule is now defined the next section will deal with the problem, how to construct good schedules to a given hierarchy of demands.

### 3 The search procedure

Nurse scheduling can be considered as a hard search problem. To solve this problem several constraint processing techniques have been implemented in the ConPlan C++ library that is part of the SIEDAplan system. A heuristic search procedure uses enhanced variants of the *branch&bound* searching algorithm to locally improve a given labeling of the variables with shifts. Prospective constraint processing techniques like *forward checking* and computation of arc consistent domains are available to compute an initial labeling as well as to improve *branch&bound* search. The following sections explain more details of the search procedure and give a small nurse scheduling example.

#### 3.1 Branch&bound

The ConPlan library provides certain constraint processing extensions to the common *branch&bound* algorithm that can be explained according to the structure of *branch&bound* algorithms given in figure 2. The task is to find a labeling

$$S = (v_1 \leftarrow d_1, \dots, v_n \leftarrow d_n)$$

for the variables  $v_1, \dots, v_n \in V$  with values  $d_1, \dots, d_n \in D$ , such that  $S$  violates a set of constraints  $b$  that is minimal according to  $\succ$ . The algorithm completes a partial labeling  $L$  step by step. For each variable all available values are considered (row 4) as labels. The algorithm checks for new constraint violations (row 4a) and maintains the set  $\delta$  that holds the constraints violated by labeling  $L$ . Then, the new branch in the search tree is expanded (row 4b) considering all complete labelings comprising  $L$  and  $v \leftarrow d$ . If all variables have been labeled by  $L$  and  $L$  violates less important constraints than  $S$ , then  $L$  will be taken as new assumption for the solution (row 1). Thus,  $S$  always holds the best labeling found so far. Hence, if all labelings have been searched  $S$  will hold an optimal solution. To be more efficient the algorithm expands only a new branch of the search tree if  $L$  extended by  $v \leftarrow d$  satisfies all hard constraints and violates less important constraints than stored in the bound  $b$  (row 4b).

Figure 2 differs from common representations of the *branch&bound* [14, 5] only in one point: A labeling's merit is not stored as a real number but as a set of violated constraints (distance  $\delta$  and

---

```

branch&bound( $S, L, b, \delta, V, C, D, \succ$ )
1. if  $|L| = |S|$  and  $b \succ \delta$  then
     $b := \delta$ ;  $S := L$ ; return  $S$ ;
2. if  $|L| = |S|$  then
    return  $S$ ;
3. choose a variable  $v \in V$  that is not labeled by  $L$ ;
4. forall values  $d \in D$  in the domain of variable  $v$  do
    (a)  $\delta_{local} = \{c \in C \mid c \text{ is inconsistent with } v \leftarrow d\}$ 
    (b) if  $\delta_{local} \cap C_0 = \emptyset$  and  $\delta \not\succeq b \cup \delta_{local}$  then
         $S \leftarrow \text{branch\&bound}( S,$ 
             $L \cup \{v \leftarrow d\},$ 
             $b,$ 
             $\delta \cup \delta_{local},$ 
             $V, C, D);$ 
    (c) if  $S$  is an acceptable solution then return  $S$ ;
5. return  $S$ ;
initial call:
branch&bound( $\emptyset, \emptyset, C, \emptyset, V, C, D$ )

```

---

Figure 2: *Branch&bound* algorithms.

bound  $b$ ). Two labelings are compared by the preference ordering  $\succ$  (instead of using the natural ordering of reals).

Certain extensions of the *branch&bound* have been proposed employing adoptions of constraint processing techniques in order to increase efficiency<sup>1</sup>. Basically, these extensions concern three rows in figure 2:

- In row 4a the term “inconsistent with” can have several concrete meanings. Naive implementations only check constraints between labeled variables. In contrast, *forward checking* with domains of yet unlabeled variables is possible either with hard and soft constraints [5]. On the one hand, these procedures prune domains of unlabeled variables in advance. On the other hand, an optimistic estimation of the distance  $\delta$  is computed to reduce branching in row 4b of the algorithm. Additionally, arc-consistency with hard constraints [9] and prioritized constraints [16, 3] can be maintained after each assignment for the same purposes. The algorithms for arc-consistency with prioritized constraints are as well appropriate to improve searching constraint hierarchies [11, 10].
- Prospective constraint processing of soft constraints results in an optimistic estimate of the solution quality that can be achieved committing a certain assignment  $v \leftarrow d$ . *Forward checking* identifies constraints that will be violated conducting  $v \leftarrow d$ . Arc-consistency with prioritized constraints delivers an estimate of the level in the constraint hierarchy that can be completely satisfied assigning  $d$  to  $v$ . This estimate can be exploited to consider that values first in row 4 that probably are part of a sufficient solution.
- Certain generic heuristics have been suggested to inform the choice of variables in row 3. Especially, the *minimum remaining values* heuristic, also called *first fail principle*, has been applied beneficially [1]. Two variants of this heuristic have been proposed to additionally exploit results from the propagation of constraint hierarchies [10].

<sup>1</sup>This paper will concentrate on prospective constraint processing.

---

*local-search*( $V, H, D$ )

1. compute an initial labeling of all variables in  $V$ ;
  2. choose an arbitrary  $V' \in V$ ;
  3. if  $V' = \emptyset$  then go to 6;
  4. optimize the labels of the variables in  $V'$ ;
  5. go to 2;
  6. result are the labels of  $V$ ;
- 

Figure 3: Local search algorithms.

However, these extensions do not generally suffice to deal with large search problems. Hence, one either has to prune the search space heuristically by additional constraints or explicit choice points, or one tries heuristic search procedures instead that use the *branch&bound* only to optimize a given labeling partially. While the first method requires large effort specific to the concrete domain of application, we have been able to solve the nurse scheduling problem on the latter way with only few application-specific tricks.

### 3.2 Adaptive local search

Heuristic searching of *constraint satisfaction problems* basically follows the scheme drawn in figure 3. An initial labeling is computed and improved continuously changing the labels of a relatively small set of variables  $V' \subset V$ . This optimization in row 4 can either be limited to a local improvement of the labeling [12] or accept a worse labeling with a certain probability [15]. The first method is known to improve labelings rapidly but stick to local minima of the preference criterion. In contrast, the latter techniques converge on globally optimal solutions but are slower in improving a labeling [17].

In the nurse scheduling application rapid improvement is very important, whereas the schedule is not necessarily required to be optimal due to the given problem specification. Hence, the goal of the search procedure in the SIEDAplan system is to improve a schedule of reasonable quality as good as an expert in nurse scheduling would do... but much faster. The *branch&bound* algorithm and its extensions described in section 3.1 are applied to compute the initial labeling of all variables as well as to improve the labels of  $V'$ .

Prospective constraint processing is employed to achieve an initial schedule of a reasonable quality (row 1 in figure 3). The *branch&bound* procedure of figure 2 is called accepting any solution being consistent with compulsory constraints. Maintaining arc-consistency with compulsory constraints after each assignment of a shift avoids unnecessary *backtrackings* in this phase. When the algorithm generated a complete labeling it stops at row 4c. *Forward checking* with soft constraints resp. maintaining arc-consistent compatibility values for each shift ensures that this schedule is not too bad to be used as an initial state for the local search procedure. The results of soft constraint propagation are used to assign promising shifts first.

Now, the local search procedure has to reason about opportunities to improve this initial schedule. To achieve a rapid increase of the schedule's quality the local search procedure will only accept local improvements of the schedule in row 4 of figure 3. The *branch&bound* procedure is appropriate to this task because it computes an *optimal* labeling of some variables  $V'$  that have been recognized as bad regions of the current schedule. In opposition to the initial call of the *branch&bound* given in figure 2 the procedure is called with an initial bound  $b$  being retrieved from the current schedule that is to be improved. Thus, the algorithm is able to prune assignments early off the search space that can not lead to a locally improved schedule by the test in row 4b.

Dienstplangenerierung																			
Station: Innere-01		von: 01.11.1996 bis: 30.11.1996																	
	ZK	Fr 1	Sa 2	So 3	Mo 4	Di 5	Mi 6	Do 7	Fr 8	Sa 9	So 10	Mo 11	Di 12	Mi 13	Do 14	Fr 15	Sa 16	So 17	M 18
Hübner Günther	-15h24	N1	N1	N1	N1	N1	N1	N1	N1	-	-	*	-	-	-	N1	N1	N1	N1
Leibig Markus	-1h24	F1	-	F1	F1	S1	S1	S1	S1	S1	S1	S1	*	F1	F1	F1	*	-	S1
Löffler Rita	-8h54	S1	-	-	F1	F1	F1	F1	F1	F1	UL	UL	UL	UL	UL	UL	-	-	F1
Mischnick Eva	3h06	N1	-	-	-	-	-	-	N1	N1	N1	N1	N1	N1	N1	N1	*	-	N1
Müller Heidrun	-2h24	S1	S1	S1	S1	*	F1	F1	F1	-	-	S1	S1	S1	S1	S1	S1	S1	S1
Schmidt Bettina	-1h24	F1	F1	S1	S1	S1	*	S1	S1	-	-	F1	F1	F1	F1	F1	F1	S1	S1

	ZK	Fr 1	Sa 2	So 3	Mo 4	Di 5	Mi 6	Do 7	Fr 8	Sa 9	So 10	Mo 11	Di 12	Mi 13	Do 14	Fr 15	Sa 16	So 17	M 18
Hübner Günther	-15h24	-	N1	N1	N1	N1	N1	N1	*	-	F1	F1	F1	*	-	-	N1	N1	N1
Leibig Markus	-1h24	F1	-	F1	F1	F1	S1	S1	S1	S1	S1	S1	*	F1	F1	F1	*	-	F1
Löffler Rita	-8h54	N1	-	F1	F1	F1	F1	F1	F1	UL	UL	UL	UL	UL	UL	UL	*	-	F1
Mischnick Eva	3h06	-	-	-	-	-	-	-	N1	N1	N1	N1	N1	N1	N1	N1	*	-	N1
Müller Heidrun	-2h24	S1	S1	S1	S1	*	F1	F1	F1	-	-	S1	S1	S1	S1	S1	S1	S1	S1
Schmidt Bettina	-1h24	F1	F1	S1	S1	S1	*	S1	S1	-	-	F1	F1	F1	F1	F1	F1	F1	S1

Figure 4: The initial and the improved schedule.

The problem of improving a schedule is reduced to the problem of finding bad regions in a given schedule. One of the most important violated constraints is chosen. Candidate variables for the set  $V'$  are computed according to some heuristics. The following section will draw a more detailed picture of these heuristics. The algorithm tries at first to repair the current schedule changing a single assignment in order to converge quickly on a sufficient schedule. Thus, a single variable is taken from this set of candidates at random as a unique element of  $V'$ . The *branch&bound* procedure is called to improve the labeling. If the schedule has been improved, the next violated constraint will be chosen to improve the schedule. But of course this first try often fails. Two reasons are possible: On the one hand changing the label of a single candidate may be possible but the procedure has chosen a wrong one. On the other hand the procedure may have reached a local minimum that requires to change more than one labeling in order to achieve an improvement. Hence, the algorithm now stores *two* randomly chosen candidates in  $V'$  to improve the schedule in the next loop of the local search procedure. In the first case, the probability of choosing the right labels to change is increased. In the latter case, there is a chance to escape from the local minimum. If this try fails again, *three* variables will be chosen and so on up to a size of  $V'$  of seven. Experience showed that optimizing the labeling of 8 or more variables is not worth its effort. Hence, if this bound is exceeded, the next constraint is chosen to compute a new set of candidates.

This method enables local search to repair easy deficiencies with small effort without getting stuck to local minima that could be improved by a human expert.

### 3.3 Example

The local search algorithm as described in section 3.2 does not depend on characteristics of the application domain. However, the crucial point of finding a set of variables  $V'$  to be optimized has been left over. In fact, this task can not be done without considering additional knowledge of the application domain that goes beyond the representation as a HCSP, yet. The best way to describe this method is to consider a small example.

Figure 4 shows two cuts of a schedule. The upper one represents the initial labeling of the variables by shifts. Forward checking of constraints suffices to compute a schedule of this kind. Bad regions of the

schedule have been shaded. Conspicuously many constraints concerning crew attendance are violated. These constraint have been neglected computing the initial labeling because the heuristics of the local search are especially appropriate to achieve the satisfaction of these constraints. Dark shaded rows represent the violation of a constraint demanding a minimal crew on the ward, light shadings denote a difference from the preferred attendance. For example on Sunday the 10th nobody is at the ward during the early-morning shift F1. On Sunday the 17th two nurses attend during the day-time S1 but nobody serves an early-morning shift. On some days too many nurses are working, e.g. on Friday 1st, and Sunday 3rd. Note, that shifts on weekends have to be compensated by an idle shift (\*) as early as possible.

The labelings to be improved ( $V'$ ) are chosen according to violated constraints. Some of these failures can be repaired very easily like for instance the constraint on crew attendance on Sunday 17th. As the improved schedule in the bottom of figure 4 shows, only the change of a single label in variable

*shiftOfAt(Schmidt, 17th)*

is necessary to satisfy this constraint. In contrast, the preferred crew attendance on Sunday 10th requires a more elaborate heuristic. On this day too few employees are present at the ward. Hence, the algorithm chooses the variable concerning an employee not working on this day (in this case *shiftOfAt(Hübner, 10th)*) together with a variable in the same row at a day, when too many nurses are present (*shiftOfAt(Hübner, 8th)*) for  $V'$ . The *branch&bound* is called with this input to fill in better shifts. These heuristics consider the characteristic of the available working time to be a more or less fixed resource. A request for more working time on a certain day has to be compensated at another day<sup>2</sup>.

The schedule in the bottom of figure 4 is the finally returned solution of the scheduling process. Even this small example of only five employees shows that generally not all demands on the schedule can be fulfilled. On some days more employees work than required. This is because the available resources are typically not fully compatible with the expenditure of work. Thus, the representation of the application as an optimization problem is justified.

## 4 Experiences

As mentioned above the SIEDAplan system is currently tested at the DRK hospital Neuwied. The system manages databases holding all the personnel data, working time accounts, and working time models of the hospital staff and of course the properties of currently valid shifts. All previously mentioned constraints are derived from this data.

In order to compute a schedule the relevant data is retrieved from the database. SIEDAplan builds up and solves the corresponding HCSP problem calling methods provided by the ConPlan C++ library. For a ward of 15 to 25 nurses the search for a schedule of sufficient quality takes about 5 to 20 minutes depending on the adequacy of the employed local search heuristics. Mostly, compensation of large working time credits resp. debts is responsible for worse performance. However, these situations are difficult also for human experts which need hours instead of minutes to generate one schedule.

If scheduling results do yet not suffice, the expert is allowed to change the whole problem description including single shifts in the schedule, working time models, attendance requirements, or even shift types. Another user interface for labeling bad regions in the schedule before restarting local search is in preparation. However, these opportunities have hardly been needed, yet.

---

<sup>2</sup>By the way, a similar procedure is well known in the context of resource oriented configuration systems [6].

## Acknowledgment

I would like to thank Enno Tolzmann from the SIEDA company for his patience and his good ideas.

## Bibliography

- [1] Fahiem Bacchus and Paul van Run. Dynamic variable ordering in CSPs. In [13], pages 258–275, 1995.
- [2] Alan Borning, Bjorn Freeman-Benson, and Molly Wilson. Constraint hierarchies. *Lisp and Symbolic Computation*, 5:233–270, 1992.
- [3] Didier Dubois, Hélène Fargier, and Henri Prade. Propagation and satisfaction of flexible constraints. In R. Yager and L. A. Zadeh, editors, *Fuzzy Sets, Neural Networks and Soft Computing*, pages 166–187, New York, 1994. Van Nostrand Reinhold.
- [4] Eugene C. Freuder and Daniel Sabin. Interchangeability supports abstraction and reformulation for constraint satisfaction. In *SARA-95: Symposium on Abstraction, Reformulation and Approximation*, 1995.
- [5] Eugene C. Freuder and Rick J. Wallace. Partial constraint satisfaction. *Artificial Intelligence*, 58:21–70, 1992.
- [6] M. Heinrich and E.-W. Jüngst. A resource-based paradigm for the configuring of technical systems from modular components. In *CAIA-91: Proc. of the 7th IEEE Conference on AI Applications*, 1991.
- [7] Walter Hower and Zsófia Ruttkay, editors. *Non-Standard Constraint Processing*, Working Notes of the ECAI'96 Workshop W27, Budapest, Hungary, 1996.
- [8] Michael Jampel, Eugene Freuder, and Michael Maher, editors. *Workshop Notes CP95 Workshop on Over-Constrained Systems*, Cassis, France, 1995.
- [9] Alan Mackworth. Consistency in networks of relations. *Artificial Intelligence*, 8(1):99–118, 1977.
- [10] Harald Meyer auf'm Hofe. Partial satisfaction of constraint hierarchies in reactive and interactive configuration. In [7], pages 61–72, 1996.
- [11] Harald Meyer auf'm Hofe and B. Tschaitshian. PCSPs with hierarchical constraint orderings in real world scheduling applications. In [8], pages 69–76, 1995.
- [12] Steven Minton, Mark Johnston, Andrew Philips, and Philip Laird. Minimizing conflicts: a heuristic repair method for constraint satisfaction problem and scheduling problems. *Artificial Intelligence*, 58:161–205, 1992.
- [13] Ugo Montanari and Francesca Rossi, editors. *CP-95: Proceedings of the First International Conference on Principles and Practice of Constraint Programming*, LNCS-976. Springer-Verlag, 1995.
- [14] E. M. Reingold, J. Nievergeld, and N. Deo. *Combinatorial algorithms: theory and practice*. Prentice Hall, Englewood Cliffs, New Jersey, 1977.
- [15] Bart Selman, Hector Levesque, and David Mitchell. A new method for solving hard satisfiability problems. In *AAAI-92: Proceedings of the 10th National Conference on AI*, pages 440–446, 1992.
- [16] Paul Snow and Eugene C. Freuder. Improved relaxation and search methods for approximate constraint satisfaction with a maximin criterion. In *Proc. of the 8<sup>th</sup> biennial conf. of the canadian society for comput. studies of intelligence*, pages 227–230, May 1990.
- [17] Rick J. Wallace and Eugene C. Freuder. Heuristic methods for over-constrained constraint satisfaction problems. In [8], pages 97–101, 1995.
- [18] Georges Weil and Kamel Heus. Eliminating interchangeable values in the nurse scheduling problem formulated as a constraint satisfaction problem. In *CONSTRAINT-95: The FLAIRS-95 international workshop on constraint-based reasoning*, Melbourne Beach, FL, USA, April 1995.