Nurse Scheduling using Constraint Logic Programming

Slim Abdennadher
Computer Science Institute, University of Munich
Oettingenstr. 67, 80538 Munich, Germany
Slim.Abdennadher@informatik.uni-muenchen.de

Hans Schlenker
Technical University of Berlin
Franklinstr. 28/29, 10587 Berlin, Germany
hans@cs.tu-berlin.de

Abstract

The nurse scheduling problem consists of assigning working shifts to each nurse on each day of a certain period of time. A typical problem comprises 600 to 800 assignments that have to take into account several requirements such as minimal allocation of a station, legal regulations and wishes of the personnel. This planning is a difficult and time-consuming expert task and is still done manually. INTERDIP[1] is an advanced industrial prototype that supports semi-automatic creation of such rosters. Using the artificial intelligence approach, constraint reasoning and constraint programming, INTERDIP creates a roster interactively within some minutes instead of by hand some hours. Additionally, it mostly produces better results. INTERDIP was developed in collaboration with Siemens Nixdorf. It was presented at the Systems'98 Computer exhibition in Munich and several companies have inquired to market our system.

Introduction

Many real-life problems lead to combinatorial search, computationally a very intensive task. Unfortunately, no general method exists for solving this kind of problems efficiently. The automatic generation of duty rosters for hospital wards falls under this class of problems.

Since the manually generated solution of the nurse scheduling problem usually requires several hours of work, a lot of research has been done to reduce the amount of time needed in the roster development. The most popular technique is based on mathematical programming (War76). The main disadvantage of this approach is the difficulty of incorporating application-specific constraints into the problem formulation. Other methods include goal programming (AR81) and heuristic models (SWB79).

Recently, Constraint Logic Programming (JM94; FA97; MS98) (CLP) has become a promising approach for solving scheduling problems. CLP combines the advantages of two declarative paradigms: logic programming and constraint solving. In logic programming, problems are stated in a declarative way using rules to define relations (predicates). Problems are solved using chronological backtrack search to explore choices. In constraint solving, efficient special-purpose algorithms are employed to solve sub-problems involving distinguished relations referred to as constraints, which can be considered as pieces of partial information. The nurse scheduling problem can be elegantly formalized as a constraint satisfaction problem (Mac92) and implemented by means of specialized constraint solving techniques that are available in CLP languages.

In this paper, the generation of duty rosters for hospitals is tackled using the CLP framework. The System is called INTERDIP and has been successfully tested on a real ward at the "*Klinikum Innenstadt*" hospital in Munich (AS97). INTERDIP has been implemented in collaboration with Siemens-Nixdorf-Informationssysteme AG using IF/Prolog (Sie96b) which includes a constraint package (Sie96a) based on CHIP (DVS$^+$88). This package includes, among others linear equations, constraints over finite domains and boolean constraints.

The nurse scheduling problem consists in assigning a working shift to each nurse on each day of a planning period (usually one month), whereby several requirements must be considered, such as minimal allocation of a ward, legal regulations and wishes of the personnel. Usually not all specified requirements can be fulfilled. The nurse scheduling problem can be modelled as a partial constraint satisfaction problem (FW92). It requires the processing of *hard* and *soft* constraints to cope with. Hard constraints are conditions that must be satisfied, soft constraints may be violated, but should be satisfied as far as possible.

Several approaches have been proposed to deal with soft constraints: Hierarchical constraint logic programming (HCLP) (BFW92) supports a hierarchical organiziaton of constraints, where a constraint on some level is more important than any set of constraints from lower levels. To avoid the so called inter-hierarchy comparison in HCLP, the soft constraints are encoded in a hierarchical constraint satisfaction problem (HCSP) (Mey97). The Conplan/SIEDAplan (Mey97) considers the representation of nurse scheduling as a HCSP, where legal

[1]INTERDIP is an acronym for the German "Interaktiver Dienstplaner".

regulations are hard constraints and wishes of nurses usually have the lowest priority level. The result is also not necessarily of a reasonable quality in respect to the nurse's wishes.

However, in practice nurses' wishes should be considered in order to support the working climate. Furthermore, some wishes of nurses are sometimes more important than some legal regulations. To deal with these requirements, INTERDIP provides a solution technique based on a variant of branch-and-bound search instead of chronological backtracking. This approach starts with a solution and requires the next solution to be better. Quality is measured by a suitable cost function. The cost function depends on the set of satisfied soft constraints.

To improve on the theoretical complexity of the problem, our system is based on an imitation of the human way of solving the problem: A roster is generated with INTERDIP through several *phases*. Additionally, several days in the roster are assigned simultaneously through user defined *patterns*. A pattern describes a preferred sequence of working days.

With INTERDIP, a user who is to some extent familiar with nurse scheduling can interactively generate a roster within minutes.

The paper is organized as follows. The next section introduces the nurse scheduling problem. Then we show how the problem can be modelled as a partial constraint satisfaction problem. In Section 4 and Section 5 we describe the implementation and the user interface. Finally, we conclude with an evaluation of our tool. Portions of this paper were taken from (AS99).

Description of the problem

In a hospital, a new duty roster must be generated for each ward monthly. A hospital ward is an organizational unit that has to fulfil some concrete tasks, and has both rooms and personnel, the nurses, at its disposal. Usually, the wards of a hospital are completely distinct: each has its own rooms and its own personnel. Therefore all rosters of a hospital can be scheduled separately. We consider in the following the scheduling problem for one ward.

A roster of one month is an assignment of the personnel of the ward to the shifts for all the days of the month. A shift is a working unit: in a common working model, each day has the units *morning shift* (e.g. 06:00 to 15:00), *evening shift* (14:00 to 23:00), and *night shift* (22:00 to 07:00) and possibly others. To each shift of every day, personnel has to be assigned.

For the generation of a roster, different kinds of constraints must be taken into account:

- *Legal regulations*, e.g. the maximum working time of a person per day or week, or time off in lieu, or maternity leave. In Germany for example the statutory monthly core working hours for a hospital with a 37.5 hour week is about 160 hours depending on the month. So, with an average shift length of 8 hours, each nurse has to work on average 20 shifts. Another law says that between two (working-) shifts, each nurse has to have a break of at least 11 hours ("11 hours rule"). If a nurse works one day in the night shift, she must therefore not be assigned the morning or evening shift the next day. Also a morning shift must not follow an evening shift.
- *Organizational rules* are those that apply specifically to one particular hospital, a part of a hospital or even only one ward. They are given by the respective management. Those are mainly the number and kind of the shifts and – within statutory limits – the minimum personnel allocation of each ward. In the following we consider a model with three shifts: morning, evening and night shift. To morning shift and evening shift at least three nurses must be assigned, and the night shift requires at least two nurses.
- *Personnel data* define the individual frame for each person. These are mainly the contractually established monthly core working time, pending vacation and accrued hours of overtime. If, for example, a nurse has 16 hours overtime, she might be scheduled two shifts less than average.
- Finally, *wishes* are requirements given by the personnel. These are mostly wishes to have some days off, for example at weekends, holidays, birthdays, or for a vacation period.

Often, there is no duty roster that fulfills all the constraints. Therefore we distinguish two kinds of constraints. Hard constraints must always be satisfied, soft constraints may be violated. Roughly speaking, legal regulations, organizational rules and personnel data determine hard constraints, wishes may be hard or soft constraints. So for example the vacation scheduling might be done for a longer term (some months) apart from the actual roster planning. Then a wish for one day of vacation would be a hard constraint, because it was planned externally. Other wishes are mostly soft constraints. Often the nurses have the opportunity to classify their wishes into some "priority levels". If possible, the wishes in one of those levels will then be regarded as hard constraints.

A roster is correct, iff all hard constraints hold. The quality of a roster results from the number of the fulfilled soft constraints and their priorities.

Modeling the problem as PCSP

Constraint Satisfaction Problems (CSPs) have been a subject of research in artificial intelligence for many years. A CSP is a pair (V, C), where V is a finite set of variables, each associated with a finite domain, and C is a finite set of constraints. A *solution* of a CSP maps each variable to a value of its domain such that all the constraints are satisfied. A *partial constraint satisfaction problem* (PCSP) (FW92) is a triple (V, C, ω), where (V, C) is a CSP and ω maps constraints to weights. A constraint's weight expresses the importance of its fulfillment, allowing to distinguish *hard constraints*, which must not be violated, from *soft constraints*, which should not be violated, but may be violated in case this is unavoidable. Hard constraints have an infinite weight. The finite weights of soft constraints allow for the specification of priorities among constraints. A *solution* of a PCSP maps each variable to a value of its domain such that all hard constraints are satisfied and the total weight of the violated soft constraints is minimal.

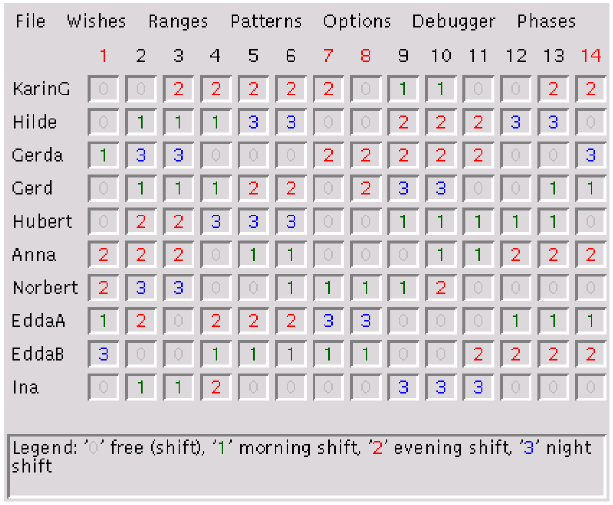
In the representation of nurse scheduling as a PCSP, there is a constraint variable for each nurse on each day. The domains of the variables consist of possible shifts (also comprising vacations, recuperation of a worked public holiday, special leaves, maternity protection, unpaid leave etc.), so they usually consist of 10 values. (HW96) proposed a reduction of variable domains, based on elimination of interchangeable values introduced by Freuder (Fre91). The values of the above mentioned free shifts, e.g. vacations, can be reduced to only one value and each variable takes its values now in $\{0, 1, 2, 3\}$. For a nurse i and a day j a variable V_{ij} may have one of the following values:

- $V_{ij} = 0$: The nurse i is off-duty the day j.
- $V_{ij} = 1$: The nurse i is assigned to the "morning" shift on the day j.
- $V_{ij} = 2$: The nurse i is assigned to the "evening" shift on the day j.
- $V_{ij} = 3$: The nurse i is assigned to the "night" shift on the day j.

Reducing the variable domains from 10 values to 4 considerably improves the efficiency of the solution research. Figure 1 shows a complete schedule for 10 nurses and 14 days. Each row comprises the shifts of a certain nurse. The columns contain the shifts performed on a certain day. So, each square of the chart specifies for each nurse the working days and shifts, and days off. E.g. on the 4th day the second nurse Hilde is scheduled in shift 1, i.e. morning shift.

Now we describe how to express the most important requirements of our application in terms of IF/Prolog-Constraints (Sie96b). In the following, we use a Prolog-like notation with meta-variables. We denote the total number of nurses to be scheduled by s, the total number of days by t and a variable by Vij, where i denotes the number of the nurse or the row in the roster, respectively, and j denotes the number of the day, i.e. the column in the roster. With this notation, we can write down all the variables of this modeling in a list: [V11,V12,...,Vst].

One requirement for a correct roster is the minimum personal allocation, i.e. the minimal number of nurses, the ward must be allocated each shift. Actually, the allocation is limited downward *and* upward. Let `Min1` be the lower and `Max1` be the upper allocation limit for the morning shift and `Min2`, `Max2`, `Min3` and `Max3` the



	1	2	3	4	5	6	7	8	9	10	11	12	13	14
KarinG	0	0	2	2	2	2	2	0	1	1	0	0	2	2
Hilde	0	1	1	1	3	3	0	0	2	2	2	3	3	0
Gerda	1	3	3	0	0	0	2	2	2	2	2	0	0	3
Gerd	0	1	1	1	2	2	0	2	3	3	0	0	1	1
Hubert	0	2	2	3	3	3	0	0	1	1	1	1	1	0
Anna	2	2	2	0	1	1	0	0	0	1	1	2	2	2
Norbert	2	3	3	0	0	1	1	1	1	2	0	0	0	0
EddaA	1	2	0	2	2	2	3	3	0	0	0	1	1	1
EddaB	3	0	0	1	1	1	1	1	0	0	2	2	2	2
Ina	0	1	1	2	0	0	0	0	3	3	3	0	0	0

Legend: '0' free (shift), '1' morning shift, '2' evening shift, '3' night shift

Figure 1: A nurse schedule for 10 nurses over a period of 14 days

lower and upper limits for the evening and night shifts, respectively. Therefore a correct roster must not have less than `Min1` and more than `Max1` times the '1' in each column and not less than `Min2` and not more than `Max2` the '2' and so on. So we have to state for each j $(1 \leq j \leq t)$ and each k $(k \in \{1, 2, 3\})$ the following constraint:

cardinality(Min$k$,Max$k$,[V1j=k,V2j=k,...,Vsj=k])

where `cardinality(Lower,Upper,Condition)` is satisfied if at least *Lower* and at most *Upper* conditions in the list *Condition* are satisfied.

Another requirement a schedule has to fulfil is the compliance of the monthly core working hours of each nurse. This means that there is a lower bound and an upper bound of shifts, each nurse is to be assigned in the schedule period. This is the number of all the morning, evening and night shifts. This can be expressed simpler by the number of free shifts. Let for each nurse i $(1 \leq i \leq s)$ the lower bound for the working shifts be given by Min$i$ and the upper bound by Max$i$. Then we can formulate the working hours requirement using the `cardinality` constraint:

cardinality(t-Max$i$,t-Min$i$,[Vi1=0,...,Vit=0])

The "11 hours rule" implies that a nurse must not work a morning shift (the day) after an evening shift and may work (the day) after a night shift only a night shift. We can express the "11 hours rule" by the following expression: If Vij is assigned a specific value, the assignment of V$i(j+1)$ must fulfill a certain condition. This can be expressed directly by the `domain_if` constraint. We state for each i $(1 \leq i \leq s)$ and for each j $(1 \leq j < t)$:

domain_if(Vij = 2, V$i(j+1)$ \= 1) and
domain_if(Vij = 3, V$i(j+1)$ in [0,3]).

The constraint `domain_if(Condition, ThenGoal)` is used to call a goal conditionally. If the arithmetic constraint *Condition* is satisfied, *ThenGoal* is called. If the arithmetic constraint is not satisfiable, `true` is called. The execution of the `domain_if` constraint is delayed as long as the satisfiability of *Condition* has not been determined.

Free shifts, provided they can be considered hard wishes, lead to immediate variable assignments. A wish (e.g. vacation) of nurse i at day j can then be stated as: `V`ij `= 0`.

Soft wishes, like all other soft conditions, can not be stated directly as (IF/Prolog-)constraints, since our constraint solver can only handle hard constraints. We only can use them for optimizing correct rosters. This will be explained in Section .

Planning in INTERDIP

The modeling just described, while being simple and straightforward, is unfortunately very costly: The search space is huge, i.e. 4^{600} for 20 nurses and a period of one month. Therefore we developed a method to prune the search tree which was inspired by the usual manual planning.

Planning by hand

Because of the huge search space a roster is usually generated by hand in two phases. In the first phase we have all liberties for assigning the cells of the roster. Therefore here we do the most complicated assignment (which is tied to most of the conditions): the allocation of the free days or shifts. Those are bound to a lot of constraints: they determine how many shifts a nurse has to work during the scheduled period, how many nurses over all shifts the ward is assigned each day, and not least most of the wishes are to be considered here: the wishes for free shifts (e.g. vacations). Closely connected with the free shifts are the night shifts: the "11 hour rule" enforces for the assignment of shifts to a nurse, that after a night shift there may follow only a night shift or a free shift (free day).

Therefore, when manually scheduling, the free and the night shifts are allocated in the first phase. In the second phase, the morning and the evening shifts are distributed among the not yet allocated cells of the roster.

The obvious advantage of the scheduling in two phases over the scheduling in one phase is the reduction of complexity: in each phase there have to be considered fewer constraints and, above all, fewer assignments[2].

[2]The assignment of the first phase is normally not changed within the second unless it is then impossible to get a solution and a change in the free and night shifts will probably enable one. The extent of those changes can be neglected: we never observed more than 10 changes.

Phasewise plan generation

In 1993, (van93) presented a partial automatic solution to the nurse scheduling problem that used two very different phases. It flexibly generated good rosters but did not handle night shifts. INTERDIP uses more than two phases which are performed in the same manner by one constraint solver.

We wanted to reduce the search space even further than (van93) did. The idea is to furthermore decompose the problem. We use three phases instead of two:

1st Phase Distribution of the free shifts.

2nd Phase Distribution of the night shifts.

3rd Phase Distribution of the morning and the evening shifts.

With this modeling, in each phase for every cell of the roster, only the minimal decision between two possibilities has to be made. This reduces the search space. We will see how we obtain a complete roster after the three phases.

In each phase, every variable is assigned a value out of the boolean domain $\{0, 1\}$. Depending on the phase, the values 0 and 1 have different meanings. If a variable in the first phase is assigned the value one, this means that the roster gets a free shift in the appropriate cell. The cells whose variables are bound to 0 remain undecided. The second phase only treats the undecided cells: if a variable gets the value 1, the cell is assigned a night shift. The rest remains undecided. In the third phase each still not decided cell is filled with either morning or evening shift, depending on whether the variable was assigned a 1 or a 0, respectively (see Figure 2, the meaning of the bold numbers is just as in Figure 1.). A complete roster results from all three phases.

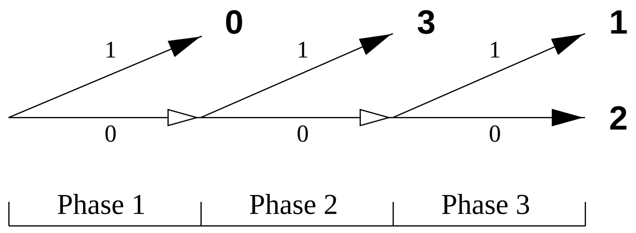


Figure 2: Allocation of the cells in three phases.

Assignment patterns

Because of the incomplete constraint propagation methods used for scheduling problems, the application programmer often has to explicitly use a labeling phase in which a backtracking search blindly tries different values for the variables. Since labeling is expensive, the programmer needs to employ techniques for reducing the search space. There is a variety of techniques to do this. For our application we add domain information about presumably good solutions by introducing *patterns*. A pattern describes a preferred sequence of working days. Coherent cells of the roster are allocated along user defined patterns.

As shown above, the variables are declared in each phase to range over the values 0 and 1 and the appropriate shifts are registered into the roster. Patterns are then meaningful combinations of roster entries, whereby a combination stands for successive days. A large number of these patterns is known. For example, we consider meaningful the combination of five days work and two days free. The appropriate pattern for the first phase, in which the working days are determined, is then: (?, ?, ?, ?, ?, 0, 0). If we assume that it is better to work on three successive days in the same shift than in different ones, we formulate for the second phase and thus for the night shifts: (3,3,3). Each phase has its own set of patterns. The patterns of a phase have an order in which they are selected: first, the ones which result in a *good* solution, since the nurses are accustomed to this pattern, and at the end trivial patterns which are necessary to generate solutions, if they exist. For filling the roster, the given patterns are translated into appropriate variable assignments which are then tried in each row from left to right.

The patterns can be considered as requirements of minor priority (soft constraints) as well as probable parts of solutions. Schedules that comply with the given patterns are explored first. Applying this specialized labeling method reorganizes the search space.

Additionally, each pattern is assigned a cost value so that for example a nurse whose wishes could not be fully fulfilled, more likely gets "better" work patterns assigned.

Optimal rosters

A roster that satisfies all hard constraints is considered feasible but this does not necessarily mean that it is sufficiently good to be used by a hospital ward.

The concept of an optimal roster is hard to define. Generally, roster quality is a subjective matter and its definition changes from problem to problem. We apply the usual measure which is common to all applications in the field of scheduling. It is given in terms of the number and the priority of soft constraints that are violated.

A popular approach consists in using a branch and bound search instead of chronological backtracking. Branch and bound starts out from a solution and requires the next solution to be better. Quality is measured by a suitable cost function. The cost function depends on the set of satisfied soft constraints. With this approach, however, soft constraints are only part of the cost function but play no role in selecting variables and values. In our multiphase method, branch and bound search is performed three times to improve the roster generated so far.

Costs arise separately for each nurse and the algorithm tries to minimize the maximum of these. This means that we have a separate cost function for each of the nurses and the maximum value of all the functions is minimized. So, INTERDIP tries to achieve that no nurse gets a much worse allocation (e.g. no wishes satisfied) than the others.

Using the system interactively

For a nurse scheduling system to be complete, a flexible user interface should be provided, so that the specific requirements of the problem can be stated easily. INTERDIP provides such an interface.

The INTERDIP user interface has been developed using the Tcl/Tk extension of IF/Prolog. Figure 1 shows a snapshot of the top-level graphical user interface to our nurse scheduling program with a generated roster.

The interface allows the user to define the system parameters as preferred. All parameters like minimal and maximal allocation of the ward for each phase, wishes or patterns can be given graphically or in a spreadsheet.

The wishes are given in three categories: imperative, important and less important wishes. We call them *red*, *black* and *white* wishes, respectively. This naming goes back to how the wishes were actually formulated in the hospital where we tested INTERDIP: They were filled into a plan using red and black pencils. The white wishes are to some extent standard wishes, like not to work on weekends. Red wishes (like vacation) are later treated as hard constraints and all the others as soft constraints. A single wish always relates to exactly one nurse and one day.

Usually the generation of a roster runs as follows. After the user has specified all the conditions he will trigger the phases. A phase starts with generating the constraints and testing their consistency. Then, according to the above method, an optimal solution is computed. After a phase is finished, the next one is started initialized with the best result of the preceding phase, and so on. This is the automatic generation.

It may happen that there exists not even one roster that complies with all the given hard constraints. Then the problem is called *over-constrained.* INTERDIP may detect this while generating the IF/Prolog constraints and then gives the user hints which of the conditions led to the inconsistency. However, there are kinds of contradictions that are not automatically detected. Therefore we built a debugger into INTERDIP.

Being an interactive tool, INTERDIP lets the user take part in the generation in different ways. Firstly the user usually has some freedom in specifying the problem conditions. He can directly influence the planning by giving some red wishes which directly lead to variable bindings. But the user can also interfere with a concrete process of allocation: He can use the debugger to break the computation manually or to set breakpoints. At the breakpoint (a cell of the roster), he is given all the possible patterns out of which he can choose one. The computation then continues with the selected allocation. In the single-step-mode the computation is stopped after each single allocation. Additionally the user can undo allocations already made.

With the debugger, the user can manually allocate parts of the roster in order to improve automatically presented solutions on the one hand and, in case the generator did not find a solution at all, enable one on the other hand.

In addition, the user can manually alter a completely generated roster and let it check by INTERDIP. The system then tries to state all the constraints for the given variable assignments, and if one fails, it gives the user hints about the contradictions.

Conclusion

In this paper, the nurse scheduling problem is discussed and a specific system, INTERDIP, is presented, that assists a human planner in scheduling the nurse working shifts for a hospital ward. We think that our approach can be applied to many applications in the field of personnel assignment. It is quite obvious that the current implementation might even be used "as is" for every duty rota problem and therefore solves this whole problem class.

It was possible to build this planning system for nurse scheduling within a few man months using a given commercial constraint solver, IF/Prolog from Siemens Nixdorf. The CLP code is just about 4000 lines with more than half of it for user interface. INTERDIP illustrates the important potentials of constraint logic programming for the implementation of real-life applications.

INTERDIP was presented at the Systems'98 Computer exhibition in Munich and several companies are interested to market it. INTERDIP is currently tested at the "*Klinikum Innenstadt*" hospital in Munich. Typically, for 20 nurses and a period of one month, INTERDIP generates a satisfying (not optimal) schedule within a few minutes. The schedules generated by INTERDIP are comparable to those manually generated by a well experienced head nurse, sometimes even better than those. Of course this can not be guaranteed for every possible problem instance since, in general, the scheduling problem is NP-complete.

References

J. L. Arthur and A. Ravindran. A multiple objective nurse scheduling model. In *AIIE Transactions*, volume 13, 1981.

S. Abdennadher and H. Schlenker. INTERDIP – Ein Interaktiver Constraint-basierter Dienstplaner für Krankenstationen. In F. Bry, B. Freitag, and D. Seipel, editors, *12th Workshop on Logic Programming WLP'97*, September 1997.

S. Abdennadher and H. Schlenker. INTERDIP – an interactive constraint based nurse. In *Proceedings of the First International Conference and Exhibition on the Practical Application of Constraint Technologies and Logic Programming*, 1999.

A. Borning, B. N. Freeman-Benson, and M. Wilson. Constraint hierarchies. *Lisp and Symbolic Computation*, 5(3):223–270, 1992.

M. Dincbas, P. Van Hentenryck, H. Simonis, A. Aggoun, T. Graf, and F. Berthier. The Constraint Logic Programming Language CHIP. Technical Report TR-LP-37, ECRC, Munich, 1988.

T. Frühwirth and S. Abdennadher. *Constraint-Programmierung: Grundlagen und Anwendungen.* Springer-Verlag, September 1997.

E. C. Freuder. Eliminating interchangeable values in constraint satisfaction problems. In *AAAI-91 – Proceedings of the 9th national conference on artificial intelligence*, pages 227–233, 1991.

E. C. Freuder and R. J. Wallace. Partial constraint satisfaction. *Artificial Intelligence*, 58(1-3):21–70, 1992.

K. Heus and G. Weil. Constraint programming a nurse scheduling application. In *Proceedings of the Second International Conference on the Practical Application of Constraint Technology*, pages 115–127, 1996.

J. Jaffar and M. J. Maher. Constraint logic programming: A survey. *Journal of Logic Programming*, 20:503–581, 1994.

A. Mackworth. Constraint satisfaction. In Stuart C. Shapiro, editor, *Encyclopedia of Artificial Intelligence.* Wiley, 1992. Volume 1, second edition.

H. Meyer auf'm Hofe. ConPlan/SIEDAplan: Personnel assignment as a problem of hierarchical constraint satisfaction. In *Proceedings of the Third International Conference on the Practical Application of Constraint Technology*, 1997.

K. Marriott and P. Stuckey. *Programming with Constraints: An Introduction.* The MIT Press, 1998.

Siemens Nixdorf Informationssysteme AG. *IF/Prolog Constraint Problem Solver*, 1996.

Siemens Nixdorf Informationssysteme AG. *IF/Prolog Users Guide*, 1996.

L. D. Smith, A. Wiggins, and D. Bird. Post-implementation experience with computer-assisted nurse scheduling in a large hospital. In *Information Systems and Operational Research*, volume 17, 1979.

B. van den Bosch. Implementation of a CLP library and an application in nurse scheduling. Master's thesis, Katholieke Universiteit Leuven, Belgium, 1993.

D. M. Warner. Scheduling nursing personnel according to nursing preference: A mathematical programming approach. In *Operations Research*, volume 24, 1976.