Producing and Improving Time Tables by Means of Constraint and Multi-agent Systems

Kay Anke and **Rainer Staudte** and **Werner Dilger**
Fakultät für Informatik,
Technische Universität Chemnitz-Zwickau
D-09107 Chemnitz, Germany
{staudte, dilger}@informatik.tu-chemnitz.de

Abstract

This paper presents a method for solving a specific class of timetabling problems. The solution introduced here proceeds in two phases: the construction of a time table that generally is not optimal, and a subsequent improvement of the time table. The first phase was successful solved using constraint programming (finite domain), the second one was realized as a multi-agent system. Results of the proposed optimization process have been verified by data from a college time table. The language Oz served as programming language and the system DFKI-Oz as convenient implementation tool.

Introduction

The time tabling approach described in this paper was inspired by the way in which school time tables were made in those times when time tabling programs were not yet available. Then, a time table was done by hand by some authorized person which usually was not optimal. Each teacher got his time table and tried to improve his or her individual time table by negotiating with some colleagues for changes of lessons. Improving a time table meant and still means to reduce the number of free periods and to get the lessons at preferred times. In most cases the result was a time table with which all teachers were happy.

With the advent of multi-agent systems it is possible to simulate the process. Clearly, to produce a time table only by negotiations between teachers would be very inefficient. Therefore the authorized person who completes the first version of the time table must also be simulated. The best way to do this seems to be by modeling the problem as a constraint problem and to apply constraint propagation.

The time tables produced in this phase, one for each teacher and for each student group, are handed over to the teachers who are modeled as agents and to the groups which also are modeled as agents. Now the agents start to negotiate with each other to improve their individual time tables. For a successful negotiation it is required that one agent suggests another agent a change. That means, the agent must be able to produce suggestions. Clearly, a suggestion made by an agent will be an improvement of the agent's own time table. But the agent must assume that its suggestion will only be accepted by its negotiation partner if it does not worsen the time table of the partner. In order to produce a suggestion an agent makes use of the constraint system with the actual time table and the agent's preferences as inputs.

To summarize, the whole approach can be regarded as a multi-agent system consisting of a central planning agent, which simulates the authorized person mentioned above and produces the first rough time table, and a number of other agents representing teachers and student groups. Each agent uses the same constraint system for its individual restrictions and preferences. The whole system is implemented in the language Oz which seemed to be well suited for all aspects of our approach.

The paper is organized as follows: In the next section the school time table problem is shortly sketched. Then the constraint system and the solution strategy realized in this system are described. In the next section the architecture of the multi-agent system with its special features and implementation is presented. Finally we present results of our work which show that by means of the multi-agent system a significant improvement of the time table can be achieved.

The School Time Table Problem

There is no definition of a time table that is shared in common. The time table model introduced in the following section differs somewhat from other problems (Scherf 1996; Würtz 1996). For given sets of subjects (that consist of several lessons, groups, teachers, rooms, and times) assignments are to be found such that a well defined set of constraints holds and several quality criteria of the time table are obeyed as far as possible.

Typical constraints are for instance:

- For each subject there is given a set of groups, teachers, rooms, and times to chose from.

- There are times when subjects, groups, teachers, and rooms are unavailable.
- Not more than one lesson at any time for groups, teachers, and rooms.
- Rooms of sufficient size have to be assigned to the lessons.

Quality criteria for a time table can be:

- As few free periods between lessons as possible for teachers and groups.
- Minimal moves between rooms in remote parts of the building.
- The length of a working day for teachers and groups should not exceed a given limit.

The Constraint System

In this section we describe how the first version of the time table is produced by means of a constraint system, thus modeling the central planning agent.

Constraints

The problem to construct a time table where for all subjects the assignments have been made was specified about two years ago (Dilger *et al.* 1995) and successfully solved using constraint programming in Oz later (Anke96a). In this section, we describe the main ideas of this work.

There are two constraint variables for each lesson denoting a time and a room number. We can find these variables in the slots for times and rooms, respectively. Because there are time restrictions for rooms and for lessons the values of the two variables depend on each other. Beginning from initial values in the solution process these constraint variables are to propagate. The solution process starts with the time variables because a time assignment is more restrictive than a room assignment. If one of these two variables cannot get any value then there is no solution in this search tree, and the process stops.

The following constraints denote a fully assigned time table as generated by the system:

1. Each lesson can be hold only at one time within given limits for this lesson. Furthermore, the time is restricted by the free times of groups, teachers and rooms that are a priori given in the times slots.

   $$U_i.u = \left(\bigcap_{G_j \in G_i} G_j.u\right) \cap \left(\bigcap_{T_j \in T_i} T_j.u\right) \cap \left(\bigcup_{R_j \in R_i} R_j.u\right)$$

   where

$U \supseteq U_i$	...	time i
$G \supseteq G_i$	...	set of groups at time i
$T \supseteq T_i$	...	set of teachers at time i
$R \supseteq R_i$	...	set of possible rooms at time i

2. The number of lessons at the same time is limited by the number of rooms.
3. Groups, teachers and rooms have no more than one lesson at a time.
4. A block consists of several occurrences of a lesson. The following properties hold for blocks:
   - All times of a block are marked and belong to the same day. There are no free periods between the times of a block.
   - Times of different blocks do not belong to the same day.
   - There is an order relation between blocks with the same number of lessons because there is no other result if two such blocks have been changed. The block with the lower number takes earlier place than the other one.
   - There is no order relation between blocks with a different number of lessons.
   - All times of a block have the same room.
5. At any time a room is selected from the appropriate set of rooms.

There are redundant constraints for a time table. Such constraints do not really restrict the solution set, but contribute to a more efficient search.

Distribution

If a constraint system is satisfiable there exists in general more than one solution and constraint propagation is not sufficient to find them. To overcome this problem we solve the given finite domain problem P by choosing a constraint C and solving both $P \cup \{C\}$ and $P \cup \{\neg C\}$. This process is called *distribution*. The combination of propagation and distribution yields a complete solution for finite domain problems.

The distribution strategy determines the selection of the constraint C. In Oz we have two degrees of freedom:

1. choosing a well suited variable Var,
2. choosing a value Val of the domain of Var.

C is obtained by the constraint $Var = Val$ and $\neg C$ by $Var \neq Val$.

We have tested several distribution strategies for timetabling. The following modified first fail strategy turned out as well suited:

Choose from the set of not yet determined variables a variable whose domain in the so called "constraint store" has minimal size. Choose from the possible times one that has a daytime as early as possible. If there is more than one such time choose the one with the lowest number.

Tests showed that this strategy finds a first solution in reasonable time. Furthermore, the lessons are as early as possible during the day and well spread over the week. This already satisfies two important quality criteria in the first phase of timetabling. We observe that certain quality criteria influence the distribution strategy.

Example 1 (Distribution) *In the time table of figure 1 we can find free times, i. e. times that are not yet assigned, as numbers. The most early possible times are the second ones on Thursday and on Saturday. 20 is less than 32, therefore the second time on Thursday (20) will be allocated in the current computation space (C). In the copied computation space this time (20) is excluded ($\neg C$).*

	Mo	Tu	We	Th	Fr	Sa
1						
2				20		32
3			15			33
4	4	10	16	22	28	34
5		11		23		
6	6	12		24	30	

Figure 1: A time table

The Multi-Agent System

In this section we present the architecture of the multi-agent system. The focus is on the optimization of the time table, the communication and some special features of the agents. The structure of the agents is mainly determined by the use of the constraint system.

Optimization

One of the most important quality criteria for timetabling is a minimal number of free time periods. The importance of this criterion for groups and teachers differs somewhat from school to school. Free periods for rooms do not matter. In the following we model the minimization of free periods.

Timetabling knows three kinds of objects: teachers, groups, and rooms. The rooms are regarded as mere objects, the teachers and groups are taken as agents. They are able to improve their respective time tables by negotiating with each other. We distinguish between the two types of agents, the teachers and the groups.

First tests showed that the number of negotiations between time table agents was very high. Therefore the constraint system was used to reduce this communication overhead. In this way small changes in the timetable have been performed efficiently.

The multi-agent system works in rounds. In each round an agent can once improve its time table. There are two possibilities for an agent

1. eliminating free periods in its time table or
2. eliminating free periods in other time tables without increasing the number of free periods in its own.

Eliminating a free period means changing at least one time to fill this gap. As far as possible other times should not be affected.

If an agent needs not to fill gaps it does not change anything. The optimization ends if the timetable was not changed in the last round. The time table obtained in this way is the best one during the whole process. A time table is considered to be better than another if

1. there are less free periods or
2. less agents have free periods but the number of free periods in both is equal.

Communication Protocol

If an agent A has found a modification by the criteria described above then it tries to realize this proposal in the time table by negotiation. It negotiates with all other involved agents using the Protocol QUERYING as presented in figure 2 to achieve his goal.

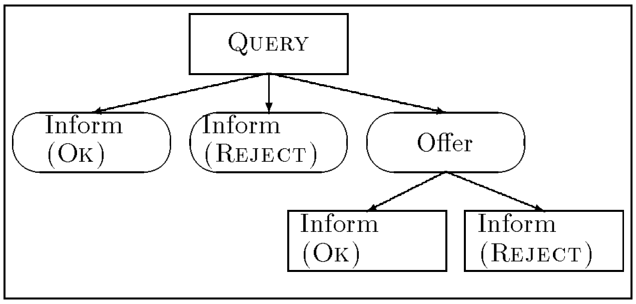

Figure 2: Protocol QUERYING

Let agent A query agent B whether it agrees with the modification. If the new time table of B is not worse than its old one (i. e. it has not more free periods than before) it will agree OK. However, if B got a considerably worse time table it will reject the query and answer REJECT. In this case the change is not realized. Agent A can try a new proposal by querying or it resigns and the next agent gets the control.

If B has only a minor disadvantage it can offer A a deal: If A rewards B balancing that disadvantage B will agree. A on its part can agree or reject this offer. In the second case nothing is changed.

Accounts

Every agent has an account with a limited initial amount. For deals like the one described in the last subsection an agent has to pay the other by transferring a certain sum to the partner's account. The initial amount of an agent equals the number of its free periods plus a basic sum.

If a worse time table is offered to an agent than its previous one, a payment is offered to him by the querying agent. An agent with a low account is not able to make many changes, especially when it tries to worsen many other time tables. But it likely has already a good one. Agents with a high amount can more easily make changes. By this way we have a certain fairness in the negotiating process.

Avoiding Cycles

During a change the total sum of all free periods can increase even though the active agent minimizes its free periods. Principally, there is a danger of cycles if an agent re-changes anything changed before. Although this case is unlikely, it may happen. Therefore the system uses a modified Simulated-Annealing-Algorithm.

Let us consider a natural number T that works as an upper bound of the number of new free periods arising in a negotiation step. With other words, T limits the amount an agent can pay during a round. T is constant during a round, but round by round its value decreases down to zero.

Unfortunately this method does not yet avoid cycles because an agent can re-change the modification made before and the accounts could be the same as before. Therefore the active agent has to pay an extra sum, i. e. a bit more than the receiver gets from it. This rest diminishes and can be viewed as a tax.

Implementation

An agent can change its time table once in a round. Agents without free periods do not change. If the time table at the end of the round is the same as before the minimization process stops. In the other case the upper bound T decreases and a new round starts. The order of agents within a round is determined before the round starts. Agents with only a few free periods become active before agents with many free periods.

If a time table is changed it is compared with the best one found up to now. If it is better it becomes the new time table. The timetabling process starts with the assignment of times to the lessons. In the next phase rooms are assigned. If no more assignment is possible for rooms a new time assignment is required. If there does not exist such a time assignment the time table problem has no solution.

If an agent becomes active it deals as described in the optimization subsection. If it is not able to find a change within a given time, its process stops by *time-out*. It gets no better new time table by his suggestion in the current round.

If an agent finds a better solution for its time table it negotiates with the other agents that are involved in this change. If all agents accept the new variation, this time table becomes the basis for further search. In the other case the active agent can look for a new proposal. The number of these trials is bounded by a given parameter.

If the sum of free periods in a new time table is greater than the sum of free periods in the old one and this difference is greater than the given upper bound T for this round the new time table is rejected. If the difference is small enough then the active agent has to pay for the changes. Any agent which gets a time table worse than before gets a reward. In addition the tax has to be paid. For all these payments the account of the active agent has to be high enough in the beginning. The reader interested in more implementation details can find the complete description in (Anke96b).

Results

Although the time table problem and its model used in this contribution have been developed before constraint and agents came into power both served as convenient starting point for investigations. While working on the subject it turned out that several a priori made assumptions could not be sticked to any longer.

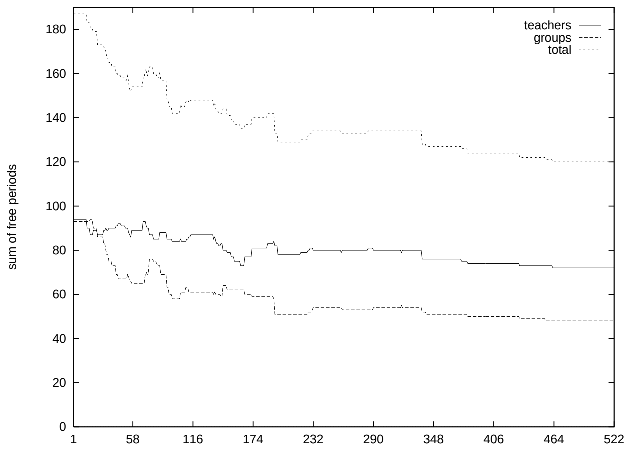

Figure 3: Total number of free periods during the optimization process

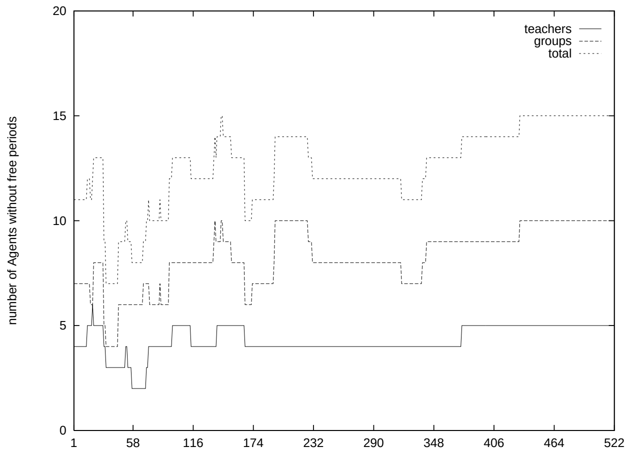

Figure 4: Number of agents without free periods during the optimization process

The attentive reader has probably noticed that the decision to look first for a full assigned time table and only afterwards to improve the most important quality criteria was not strongly obeyed. Obviously there are two classes of such criteria. A criterion of the first class is to be observed during the first phase (i. e. the lessons spread over one week). It makes no sense to take into account a criterion of the second class too early (i. e. minimizing of free periods) because this can prevent from finding any solution.

Fig. 3 shows the progress made by the multi-agent system during the solution process. The x-axis indicates the rounds made by the agents, and the y-axis the sum of free periods in all time tables for teachers and for groups during optimization. As it was to be expected, in a first period the optimization process worked well. After a certain time only modest progress concerning minimization of free periods can be observed. Any further trials to improve the time table needed much computation time. These experiences are similar to those of other heuristic methods. Fig 4 confirms these results showing the number of agents without any free period in their time tables.

	Mo	Tu	We	Th	Fr
1			**Info** 9w		**Info** 7r
2			**Info** 9w	**Info** 9m	*
3	**Info** 10w		*	*	*
4	**Info** 10w		*	**GK** 10w	*
5	*		**GK** 10t	**Werken** 5b	*
6	**GK** 9m	**GK** 10t		**Werken** 5b	**GK** 9m
7	**GK** 8h			*	**GK** 10w
8				**GK** 8r	

	Mo	Tu	We	Th	Fr
1					
2					
3	**Info** 10w				**Info** 7r
4	**Info** 10w			**Info** 9m	**GK** 9m
5	*		**GK** 10w	**Werken** 5b	**GK** 10t
6	**GK** 9m			**Werken** 5b	**GK** 10w
7	*	**GK** 8r		**Info** 9w	**GK** 8h
8	**GK** 10t			**Info** 9w	

Table 1: A teacher time table before and after optimization by the multi agent system. Free periods are marked by an asterisk.

	Mo	Tu	We	Th	Fr
1	**Mathe**	**Mathe**	**Engl**	**Engl**	**Mathe**
2	**Engl**	**Engl**	**Deu**	**Mathe**	**Engl**
3	**Bio**	**Deu**	**Mathe**	**Deu**	**Deu**
4	**Deu**	**Musik**	**Ges**	**Geo**	**Geo**
5	**Werken**	**Bio**	**Phy**	**Ethik**	**Ges**
6	**Werken**	*	**Kunst**	**Sportm**	**Sportw**
7	**Sportm**	**Sportw**		**Sportm**	**Sportw**
8					**Phy**

	Mo	Tu	We	Th	Fr
1	**Kunst**	**Mathe**	**Mathe**	**Engl**	**Deu**
2	**Engl**	**Engl**	**Deu**	**Mathe**	**Mathe**
3	**Mathe**	**Bio**	**Engl**	**Deu**	**Engl**
4	**Deu**	**Musik**	**Bio**	**Phy**	**Ges**
5	**Werken**	**Ges**	**Geo**	**Geo**	**Ethik**
6	**Werken**	**Deu**		**Sportm**	**Sportw**
7	**Sportm**	**Sportw**		**Sportm**	**Sportw**
8					**Phy**

Table 2: A group time table before and after optimization by the multi agent system. Again a free period is marked by an asterisk. Room numbers are omitted.

Results of the proposed optimization process have been verified by data from a college time table (Anke96b). In tab. 1 we can compare a teacher time table of this data set before and after optimization by the multi-agent system and observe the reduction of free periods. Tab. 2 shows a group time table. The German abbreviations of subjects are explained as follows: Bio – Biology, Deu – German, Engl – English, Ethik – Ethics, Geo – Geography, Ges – History, GK – Social studies, Info – Computer science, Kunst – Arts, Mathe – Mathematics, Musik – Music, Phy – Physics, Sportm and Sportw – Sports (male and female), Werken – Handycrafts.

Oz served as programming language and the system DFKI-Oz as convenient implementation tool. Especially the flexible constraint handling facilities in Oz allowed to implement a problem specific distribution strategy as described in the above section.

The work was started with Oz version 1.9.13 and finally transferred to 2.02. The fact that the Oz system is still under development did not sensibly slow down research and evaluation work. As other works show (Würtz 1997) the framework presented here still does not use all Oz features using constraints to tackle time table problems.

Conclusion

The paper describes the architecture of a combined constraint and multi-agent system. The combination is done in such a way that each agent makes use of the constraint system to solve its individual timetabling problem and that the whole timetabling process starts with a first run of the constraint system with global values. this first phase can be viewed as the activity of a central planning agent. We showed that by means of this approach a considerable improvement of the initial time table can be achieved.

The time tabling process can be speeded up if several processors are at hand to do real parallel processing. Our approach gives a key how to parallelize the negotiation process. In each round of this process each agent negotiates with only a small number of other agents. Therefore the set of agents could be divided into several classes before each round and the negotiation would be done only within the group but in parallel for all groups.

Acknowledgments

The authors thank Tobias Müller and all other colleagues from the DFKI in Saarbrücken for their helpful advices with respect to master the Oz system.

References

Scherf, A. 1996. Tabu Search Techniques for Large High-School Timetabling Problems. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence and the Eighth Innovative Applications of Artificial Intelligence Conference*, 363–368

Henz,M; and Würtz, J. 1996. Using Oz for College Time Tabling. The Practice and Theory of Automated Time Tabling. In *Practice and Theory of Automated Time Tabling*, 162-177

Dilger, W.; Staudte, R.; and Anke, K. 1995. Testing Oz Facilities in Distributed Problem Solving. In *WOz'95, International Workshop on Oz Programming*, 45–49. Martigny, Switzerland.

Anke, K. 1996a. Test der Programmiersprache Oz bei der Lösung von Problemen der Künstlichen Intelligenz. Projektarbeit, Technische Universität Chemnitz–Zwickau, Germany.

Anke, K. 1996b. Verbesserung von Stundenplänen durch ein Multiagentensystem. Diplomarbeit, Technische Universität Chemnitz–Zwickau, Germany.

Würtz, J. 1997. Constraint-Based Scheduling in Oz. *Symposium on Operations Research*, Braunschweig, Germany, to appear