Combining Local Search and Look-Ahead for Scheduling and Constraint Satisfaction Problems

Andrea Schaerf
Dipartimento di Informatica e Sistemistica,
Università di Roma "La Sapienza"
Via Salaria 113, 00198, Rome, Italy
e-mail: aschaerf@dis.uniroma1.it

Abstract

We propose a solution technique for scheduling and constraint satisfaction problems that combines backtracking-free constructive methods and local search techniques. Our technique incrementally constructs the solution, performing a local search on partial solutions each time the construction reaches a dead-end. Local search on the space of partial solutions is guided by a cost function based on three components: the distance to feasibility of the partial solution, a look-ahead factor, and (for optimization problems) a lower bound of the objective function. In order to improve search effectiveness, we make use of an adaptive relaxation of constraints and an interleaving of different look-ahead factors. The new technique has been successfully experimented on two real-life problems: university course scheduling and sport tournament scheduling.

1 Introduction

The solution techniques for scheduling and constraint satisfaction problems appearing in the recent AI literature can be roughly divided into two main branches. On one side, *constructive* methods build the solution step by step by adding at each step a new piece to the partial solution constructed so far. Each step generally consists in assigning a value to a variable out of its domain. On the other side, *selective* methods explore the search space composed only by complete solutions, in which all variables are assigned a value.

Constructive algorithms usually involve some heuristics for making the most promising choice at each step; such heuristics generally include some form of *look-ahead* mechanism, which takes into account the possible repercussions of the current choice on future steps. If the construction reaches a dead-end, i.e. there is no possible value for the current variable, a different partial solution is considered. In particular, backtracking-based methods, e.g. *forward checking* and *backjumping* [Prosser, 1993], may involve the exhaustive exploration of the search space and are usually *complete*; that is, they always find a solution, if it exists, and they find the optimal one, for optimization problems. Backtracking-free constructive algorithms instead make use only of some limited change in the partial solution, based on heuristics, in order to go around dead-end situations.

Among selective methods, one main branch of investigation concerns *local search* techniques. A local search algorithm, starting from an initial solution, which can be obtained with some other technique or generated at random, enters in a loop that *navigates* the search space, stepping from one solution to one of its *neighbors*. The neighborhood is composed by the solutions that can be obtained by a local change from the current solution. Changes, called *moves*, are selected based on the number of violated constraints, i.e. the so-called *distance to feasibility*, and —for optimization problems— based also on the objective function of the problem. The structure of moves depends on the specific problem; for example it can be the change of the value of one single variable or the swap of the values of two different variables. Most common local search procedures are *hill climbing*, *simulated annealing*, and *tabu search*.

In this paper, we propose a technique that combines constructive backtracking-free algorithms and local search methods. Our technique acts like a constructive one until a dead-end is reached. At this point, it performs a local search phase which makes local changes on the current partial solution. Thereafter, the construction continues up to the next dead-end. The whole procedure stops either when a full solution is reached (positive answer) or when a predetermined number of local search phases have being accomplished (negative answer).

The idea of revising the solution while constructing it is definitively not new. In fact, it is both long-standing and well established; for example, it is known in expert system literature under the name of *propose and revise*. Nevertheless, our approach differs from previous ones in at least two aspects:

1. It revises the partial solution by making use of a full run of local search, instead of a fixed number of changes. In addition, it relies on well-studied search procedures, whose good behavior has been verified in various applications, rather than using only *ad*

hoc rules, designed only for a specific problem.

2. Local changes are selected with the additional objective of improving the possibility of the partial solution to be completed. That is, local search is driven not only by the feasibility (and optimality) of the current partial solution, but also by what we call the *look-ahead factor*. Furthermore, in order to make local search effective, the respective weight given to the three different components of the cost function that guides local search —feasibility, optimality, and look-ahead— is dynamically changed.

The paper is organized as follows. Section 2 introduces our search technique and its relevant features. Sections 3 and 4 illustrate our case studies, namely the course timetabling problem and the tournament scheduling problem; moreover, they discuss the application of the technique to these problems and the related experimental results. Section 5 discusses related work. Finally, conclusions are drawn in Section 6.

2 The technique

We assume, for simplicity, that the problem consists in assigning to a set of variables $\{x_1, x_2, \ldots, x_n\}$, with associated domains $D_1, D_2, \ldots, D_n$, a value $v_i \in D_i$ (for $i = 1, \ldots, n$), subject to a set of constraints. The problem may also include an objective function to minimize. Thus, our technique is applicable to both search and optimization problems.

Our algorithm is shown in figure 1. Intuitively, it is a backtracking-free constructive method which performs a local search whenever it reaches a dead-end.

```
Procedure Solve(ProblemVars)
begin
  current := ∅;
  rest := ProblemVars;
  while rest ≠ ∅
    x = SelectVariable(rest);
    v = AssignValue(x);
    if v ≠ nil
    then
      current := current ∪ {x ← v};
      rest := rest \ {x};
    else
      if LastTrial then return ∅
      best := current;
      repeat
        move := SelectMove(current,rest);
        current := MakeMove(current,move);
        if Improves(current,best)
        then best := current
      until LastIteration or LowerBoundReached
      current := best
    endelse
  endwhile;
  return current
end;
```

Figure 1: The abstract algorithm

A concrete form of the algorithm is obtained by specifying: (*i*) the variable ordering procedure (SelectVariable), (*ii*) the value selection procedure (AssignValue), (*iii*) the local search procedure and the neighborhood relation (SelectMove), and (*iv*) the score function that we use to drive the local search (SelectMove and Improves).

The choices involved at points (*i*)–(*iv*) depend on the problem under investigation. We now discuss some general guidelines which can be stated for all problems.

2.1 Score Function

In general, local search techniques rely on a score function that assesses the quality of each solution. In search problems, such function counts the number of constraint violations, thus measuring the distance to feasibility. For optimization problems, it also takes into account the objective function of the problem. Therefore, in the general case, the score function consists of a weighted sum of these two components.

In our case, the function also includes a look-ahead factor (see Section 2.3). Furthermore, being computed on partial solutions, only constraints regarding the instantiated variables are taken into account. For the same reason, the objective function is not computed exactly, but it is generally estimated using a lower bound (in a similar way as branch-and-bound procedures).

Therefore, our score function is a weighted sum of three components: the number of violations of constraints associated with the instantiated variables, the upper bound of the objective function, and the look-ahead factor.

2.2 Adaptive Relaxation of Constraints

The choice of the weights of the score function is crucial for the effectiveness of local search. For instance, the use of a very high weight for constraint violations ensures that the number of infeasibility is never increased. However, such a choice has the drawback of making the search to be easily trapped on local minima. To overcome this problem, Gendreau *et al.* [1994] employ an adaptive relaxation of constraints. That is, the weight of constraints is adjusted based on the number of their violations in the most recent iterations. More specifically, for each kind of constraints, if there are no violations of constraints of that kind for a given number of iterations its weight is reduced; conversely, if all solutions have that kind of infeasibility, then its weight is increased.

Within our framework, assigning a fixed high value to feasibility would result in moving on feasible partial solutions, since search starts from a feasible one. Experiments show that the navigation through only feasible partial solutions does not effectively visit the search space, and the problem on being trapped on local minima is extremely critical. Therefore, we employ the above idea for our local search phase, and we dynamically adjust the weight of the constraint violations component of the score function. In details, if all solutions are feasible w.r.t. that component for k consecutive iterations, then the weight of the component is halved; conversely, if all k consecutive solutions are infeasible, then the weight is doubled ($k = 10$ in our experiments).

2.3 Look-Ahead Factor

Many constructive techniques guide variable selection and/or value assignment by using a look-ahead mechanism which estimates the likelihood of the remaining subproblem to be solvable (see e.g. [Gent *et al.*, 1996]).

Frost and Dechter [1995] experiment with four different look-ahead heuristics, and they show that *min-conflicts* (MC) is the most effective one. MC consists in simply summing up the number of available values for all the uninstantiated variables.

We consider MC and a variant of it, in which the sum is not extended to all the remaining variables, but only to a limited number of them, specifically those that are going to be scheduled in the immediate future. We call this heuristics *short-term min-conflicts* (SMC).

We therefore consider two forms of look-ahead factor: MC and SMC. Our experiments in Section 3.3 show that the interleaving of the two heuristics gives the most effective results. Intuitively, we start each local search phase by using the "broad-minded" MC, and we move to the "narrow-minded" SMC if the previous one does not succeed to find a value for the next variables to be scheduled.

3 Course Timetabling Problem

The course timetabling problem consists in the weekly scheduling for all the lectures of a set of university courses in a given set of classrooms, avoiding the overlaps of lectures having common students. We consider the basic search problem (see [Schaerf, 1997]) which has an integer programming formulation with linear constraints. Many variants of this problem have been proposed in the literature, which involve more complex constraints and usually consider an objective function to minimize.

3.1 Problem Definition

There are q courses $K_1, \ldots, K_q$, and each course K_i consists of k_i lectures, and p periods $1..p$. For all $i \in 1..q$, all lectures $l \in 1..k_i$ must be assigned to a period k in such a way that the following constraints are satisfied:

Conflicts: There are c curricula $S_1, \ldots, S_c$, which are groups of courses that have common students. Lectures of courses in S_l must be all scheduled at different times, for each $l \in 1..c$.

Availabilities: There is an availability binary matrix A of size $q \times p$. If $a_{ij} = 1$ then lectures of course i cannot be scheduled at period j.

Rooms: There are r rooms available. At most r lectures can be scheduled at period k, for each $k \in 1..p$.

The problem can be shown NP-complete through a simple reduction from the graph coloring problem.

3.2 Application of the Technique

In order to apply our technique to this problem, we consider each lecture as a variable whose value represents the period at which the lecture takes place. The number of variables is therefore equal to $\sum_{i=1}^{q} k_i$ and they all have the same domain $1..p$.

Each constructive step consists in assigning a lecture to a period. A local move consists in rescheduling a lecture in a different period to which no other lecture of the same course is assigned. During construction we enforce all constraints, whereas during search we include the count of all their violations in the score function.

Courses are ordered statically based on predetermined criteria. In particular, they are grouped based on the curricula, and groups are ordered by their size and their number of availabilities. We do not use any specific strategy for value selection, but we simply assign the first feasible one, thus delegating all look-ahead activity to the local search phase. Moreover, being a search problem, there is no objective function to minimize.

Regarding the look-ahead factor, the MC heuristic results in counting the number of available periods for all unscheduled lectures, whereas SMC counts them only for the next course to be scheduled.

In addition, if a lecture of a given course cannot be scheduled, all lectures of that course are removed and the search works on the partial solution made only by the lectures of the courses completely scheduled.

3.3 Experimental Results

We experiment on problems of two different sizes. We created two test examples: Test 1 has 20 courses (130 lectures), 5 rooms and 30 periods; Test 2 has 100 courses (580 lectures), 10 rooms and 30 periods. Conflicts among courses in the test examples are taken from real cases. In order to create different problem instances, availabilities and some other conflicts are assigned randomly, still keeping the problem solvable.

We have implemented three versions of our technique based on three different local search techniques: random hill climbing (RHC), steepest hill climbing (SHC), and min-conflict hill climbing (MCHC) [Minton *et al.*, 1992].

RHC draws at each iteration a random move, SHC visits at each iteration the whole neighborhood looking for the move that gives the best improvement (arbitrarily breaking ties), MCHC looks randomly for a lecture that causes at least one conflict and moves it to the period in which it creates the minimum number of conflicts (arbitrarily breaking ties). All three methods accept the selected move only if the cost function is improved or is left at the same value. This means that they never perform worsening moves, although they can perform sideways moves. Therefore, they have the capability of navigating *plateaus*, whereas they are all trapped by strict local minima. For all three of them the stop criterion is based on the number of iterations without improving the value of the best solution. SHC stops also when it reaches a strict local minimum, the other two methods don't do it because they do not recognize such a situation.

We experiment with both the pure local search methods and our technique that combines them with constructive methods. We make use of adaptive relaxation of constraints, in such a way that, the three types of constraints —availabilities, conflicts, and rooms— are associated with an independent dynamic weight which varies as explained in Section 2.2.

More sophisticated local search techniques, like tabu search [Schaerf, 1996b], also accept worsening moves and allow one to escape from local minima (without resorting to relaxation). We do not discuss their use in this paper because their study is still on-going; however, preliminary results with tabu search did not give any improvement on those presented in this section.

We have also implemented two complete backtracking methods: a basic chronological backtracking (CB) and a forward checking with dynamic variable selection mechanisms (FC).

We create 1000 solvable instances for Test 1 and 100 for Test 2. We fix the running time (10 seconds for instances of Test 1 and 200 seconds for instances of Test 2), and we performe as many iterations as possible in that time-frame.

For the look-ahead factor, we use an interleaving of the two methods, as mentioned in Section 2.3. In particular, we use MC for the first time the construction stops at a level; if after local search it is still not possible to schedule all lectures of the next course, for all the subsequent runs at the same level we use SMC.

Table 1 shows the percentage of the instances solved for each method. The results highlight a clear dominance of combined methods over pure local search ones. They also show that the "winning" algorithm for this problem among local search methods is MCHC, and among combined ones is Construction + MCHC. It also shows that SHC works quite well for the Test 1, but not for Test 2. This is due to the fact that for large instances the exploration of the full neighborhood is generally too time consuming. Both of the backtracking-based methods weren't able to produce a solution even in 48 hours for some instances of Test 1.

Tables 2 and 3 focus on specific features of the technique (using Construction + MCHC).

Table 2 illustrates the importance of two types of look-ahead mechanisms. The first entry shows the result without any look-ahead factor. The subsequent two entries regard each method used in isolation. The last line is repeated from Table 1. It emerges that MC performances deteriorate increasing the size of the instance, whereas performances of SMC are not negatively influenced by the number of courses.

Table 3 focuses on the importance of adaptive relaxation. The first line shows the results without adaptive relaxation, with a fixed high weight for infeasibilities. The second line shows the results using one single dynamic weight instead of three independent ones. Table 3 shows that adaptive relaxation is absolutely necessary, whereas for this specific problem, using independent weight gives only a limited advantage.

Search Method	Nr. of instances solved	
	Test 1	Test 2
Pure local search methods		
RHC	46.3%	20%
SHC	15.6%	3%
MCHC	59.0%	45%
Combined methods		
Construction + RHC	88.5%	78%
Construction + SHC	87.0%	21%
Construction + MCHC	97.4%	89%
Pure constructive methods		
CB	0%	0%
FC	0%	0%

Table 1: General results

Look-Ahead Method	Test 1	Test 2
no one	0%	0%
MC	29.2%	11%
SMC	50.1%	58%
MC + SMC	97.4%	89%

Table 2: Look-Ahead Strategy Results

4 Tournament Scheduling

We consider the problem of scheduling the matches of a round robin tournament. The problem consists in assigning matches to rounds in such a way that every team matches every other one, all teams play every round with a different opponent (either home or away), and various other constraints are satisfied.

Constraints involve the availability of stadia in given rounds, forbidden rounds for specific matches, and sharing of stadia. For the sake of brevity, we do not supply their complete definition, that can be found in [Schreuder, 1992]. We just mention that constraints are split into hard (requirements) and soft (wishes) ones: The former ones must be necessarily satisfied by the solution, the latter ones instead can be violated, and they contribute, with their associated penalty, to the *objective function* to minimize.

This problem is tackled relying on a so-called *tournament pattern*, which is a complete tournament where teams are replaced by numbers. Given a pattern, the problem consists in finding the assignment of real teams to distinct numbers in the pattern that satisfies hard constraints and minimizes the total penalty associated

Adaptive weights	Test 1	Test 2
static weights	0%	0%
one dynamic weight	95.2%	72%
three dynamic weights	97.4%	89%

Table 3: Adaptive Relaxation Results

with soft constraints. This problem has been proved NP-complete in [Schaerf, 1996a].

Schreuder [1992] developed an incomplete algorithm that solves an instance of this problem with 18 teams in about 2 minutes of cpu time. Instances of the same size are solved optimally using a diagnostic system in [Bakker *et al.*, 1993] in 25 hours. Using a specialized depth-first branch-and-bound algorithm, Schaerf [1996a] solves optimally instances of such size in about 20 minutes.

Although each instance can be solved optimally in reasonable time, in order to solve a real case, the run must be repeated several times so as to get sensibility on constraints and penalties. Therefore, it is necessary to have a fast method, that allows the user to play interactively with the constraints and the corresponding solutions.

A local search technique can be used to this aim. However, we notice that in real cases, there are a few teams that are highly constrained, and therefore a few assignments that are critical. The problem thus shows a high-degree of asymmetry. In asymmetric cases a constructive method that deals in the beginning with the most difficult assignments is more promising than a local search method. In fact, experiments in Section 4.2 confirm that our technique outperforms hill climbing methods.

4.1 Application

Given an instance with n teams, a solution to this problem is an assignment of a distinct number (i.e. a value) in $1..n$ to each team (i.e. a variable). A constructive step consists in choosing a new team and assigning to it an unused number. Numbers are assigned based on an upper bound of the objective function, which is obtained summing the penalty associated with the soft constraints on already scheduled teams and an estimation of the penalty of the other constraints. A local move on a complete solution consists in swapping the assignments given to two different teams. For partial solutions, a move consists in assigning a different unused number to the selected team.

Regarding the look-ahead factor, the MC function is the sum of the available numbers among the unused ones for each of the remaining teams. The SMC counts only the available numbers for the next team. The look-ahead factor is assigned a high weight w.r.t. soft constraints.

Teams are statically split into two groups: (*i*) *top teams*: the strongest teams, upon whose matches various kinds of constraints exists; (*ii*) *regular teams*, upon which regular availability constraints are posed. Inside each group, teams are also statically grouped based on the effective number of constraints in the instance under examination.

4.2 Experimental Results

In the same way as Section 3, we consider data from a real case (18 teams) to which we apply some random perturbations in order to obtain different instances.

Table 4 shows the results for 1000 instances. For each method parameters are set in such a way that each run lasts for 10 seconds on each instance.

Search method	Nr. of instances solved	Average value objective function
Pure local search methods		
RHC	38.3%	26.4
SHC	40.1%	27.7
MCHC	42.3%	27.3
Combined methods		
Cst. + RHC	80.3%	29.1
Cst. + SHC	34.3%	25.2
Cst. + MCHC	82.3%	27.3

Table 4: General results

These results show a clear dominance of our technique upon hill climbing methods for this problem. It also shows, however, that the constructive techniques do not necessarily find better solutions. This is due to the fact that they focus more on the look-ahead factor, rather then on the objective function.

5 Related Work

Zhang and Zhang [1996] combine constructive and local search methods in a different way. Their method finds a partial solution using local search, and then explores all its possible completions using a backtracking-based algorithm. Therefore, they also make use of local search on partial solutions, but they have no notion similar to our look-ahead factor.

Glover *et al.* [1996] use a similar approach to solve the graph coloring problem. Their method starts with a complete solution, found with a high-quality heuristics. Thereafter, it alternates a destructive phase, in which some nodes are *uncolored*, and a constructive one, in which nodes are *colored*. The uncoloring is guided by an estimation of the depth and the width of the current local minimum.

Solotorevsky *et al.* [1994] employ a propose and revise rule-based approach to the course timetabling problem. When the construction reaches a dead-end, the so-called *Local_Change Rules* come into play so as to find a possible assignment for the activity unscheduled. However, they perform only a single step before restarting the construction, and their aim is only to accommodate the pending activity, without any look-ahead mechanism.

A common approach is to use a heuristic constructive method for finding the initial solution and a local search technique to improve it. For example, Yoshikawa *et al.* [1996] solve a timetabling problem combining a constructive method based on arc-consistency with a min-conflict hill climbing phase. The construction goes all the way to the complete solution, accepting also constraint violations. The hill climbing phase improves the objective function of the problem reducing the overall penalty.

Regarding the adaptive relaxation, Selman and Kautz [1993] use a similar scheme for the satisfiability problem. In their work, an independent dynamic weight is given to each single clause in the formula. Such fine grain weighting is necessary for dealing with problems with strong asymmetries. In our technique however, asymmetries are dealt with by variable ordering in the constructive phase. Additional experiments show that if we distribute constraints in the course scheduling problem in an asymmetric way among different courses, performances of pure local search methods drastically deteriorate, whereas performances of combined methods remain (if courses are suitably ordered) at a comparable level.

Morris [1993] and Yugami *et al.* [1994], use local search techniques, relying on relaxing some constraints when a local minimum is reached. Differently from their work, our adaptive relaxation scheme is not a method to deal with local minima (which are not necessarily perceived), but is a general search guide, which makes the overall search more effective.

6 Discussion and Conclusions

We have proposed a solution technique for scheduling and constraint satisfaction problems based in the combination of backtracking-free construction and local search revision with look-ahead capabilities. Our method arises from well-established and long-standing ideas, bringing them together in a principled way, in order to achieve a general purpose technique. The technique turned out to be very effective for two NP-complete scheduling problems. In addition, it has shown a good behavior in highly asymmetric problem instances, which are generally difficult to solve for local search methods.

One of the main advantages of local search techniques is that, giving the possibility to start the search from any solution, they easily allow for interactive maintenance of solutions. In fact, once a solution as been generated, it can be used as the starting point for a new search after some constraints have been (manually) modified. In order to retain such capability, we have included in our technique the ability to run local search on complete solutions. For the course scheduling problem, we have also added the possibility to manually unschedule some lectures and restart the process from the such partial solution in the new constraint setting.

For the future, we plan to experiment our technique on other problems, in order to better validate the overall idea and to gain a clearer understanding on which local search strategies fit best within the construction process in our approach.

Acknowledgements

I wish to thank Bruno Errico for useful comments on an earlier draft of the paper.

This paper has been supported by ASI (Italian Space Agency) and by ESPRIT Project n. 22469, DWQ (Foundations of Data Warehouse Quality).

References

[Bakker *et al.*, 1993] R. R. Bakker, F. Dikker, F. Tempelman, and P. M. Wognum. Diagnosing and solving over-determined constraints satisfaction problems. In *Proc. of IJCAI-93*, pages 276–281. Morgan Kaufmann, 1993.

[Frost and Dechter, 1995] D. Frost and R. Dechter. Look-ahead value ordering for constraint satisfaction problems. In *Proc. of IJCAI-95*, pages 572–578. Morgan Kaufmann, 1995.

[Gendreau *et al.*, 1994] M. Gendreau, A. Hertz, and G. Laporte. A tabu search heuristic for the vehicle routing problem. *Management Science*, 40(10):1276–1290, 1994.

[Gent *et al.*, 1996] I. P. Gent, E. MacIntyre, P. Prosser, and T. Walsh. The constrainedness of search. In *Proc. of AAAI-96*, pages 246–252. AAAI Press/MIT Press, 1996.

[Glover *et al.*, 1996] F. Glover, M. Parker, and J. Ryan. Coloring by tabu branch and bound. In D. S. Johnson and M. A. Trick, editors, *Cliques, Coloring, and Satisfiability. Second DIMACS Implementation Challenge.* American Mathematical Society, 1996.

[Minton *et al.*, 1992] S. Minton, M. D. Johnston, A. B. Philips, and P. Laird. Minimizing conflicts: a heuristic repair method for constraint satisfaction and scheduling problems. *Artificial Intelligence*, 58:161–205, 1992.

[Morris, 1993] P. Morris. The breakout method for escaping from local minima. In *Proc. of AAAI-93*, pages 40–45. AAAI Press/MIT Press, 1993.

[Prosser, 1993] P. Prosser. Hybrid algorithms for the constraint satisfaction problem. *Computational Intelligence*, 9:268–299, 1993.

[Schaerf, 1996a] A. Schaerf. Scheduling sport tournaments using constraint logic programming. In *Proc. of ECAI-96*, pages 634–639. John Wiley & Sons, 1996.

[Schaerf, 1996b] A. Schaerf. Tabu search techniques for large high-school timetabling problems. In *Proc. of AAAI-96*, pages 363–368. AAAI Press/MIT Press, 1996.

[Schaerf, 1997] A. Schaerf. A survey of automated timetabling, 1997. To appear in *Artificial Intelligence Review*.

[Schreuder, 1992] J. A. M. Schreuder. Combinatorial aspects of construction of competition dutch professional football leagues. *Discrete Applied Mathematics*, 35:301–312, 1992.

[Selman and Kautz, 1993] B. Selman and H. A. Kautz. Domain-independent extensions to GSAT: Solving large structured satisfiability problems. In *Proc. of IJCAI-93*, pages 290–295. Morgan Kaufmann, 1993.

[Solotorevsky *et al.*, 1994] G. Solotorevsky, E. Gudes, and A. Meisels. RAPS: A rule-based language specifying resource allocation and time-tabling problems. *IEEE Transactions on Knowledge and Data Engineering*, 6(5):681–697, 1994.

[Yoshikawa *et al.*, 1996] M. Yoshikawa, K. Kaneko, T. Yamanouchi, and M. Watanabe. A constraint-based high school scheduling system. *IEEE Expert*, 11(1):63–72, 1996.

[Yugami *et al.*, 1994] N. Yugami, Y. Ohta, and H. Hara. Improving repair-based constraint satisfaction methods by value propagation. In *Proc. of AAAI-94*, pages 344–349. AAAI Press/MIT Press, 1994.

[Zhang and Zhang, 1996] J. Zhang and H. Zhang. Combining local search and backtracking techniques for constraint satisfaction. In *Proc. of AAAI-96*, pages 369–374. AAAI Press/MIT Press, 1996.