Constraint Satisfaction with a Multi-Dimensional Domain

Masanumi Yoshikawa  Shin-ichi Wada
C&C Systems Research Laboratories, NEC Corporation 4-1-1 Miyazaki, Miyamae-ku, Kawasaki 216
JAPAN

Abstract

This paper presents a novel approach to a class of constraint satisfaction problems (CSPs). First, it defines Multi-Dimensional Constraint Satisfaction Problem (MCSP), which is a useful model applicable to many scheduling problems. Second, it proposes an approach for MCSPs. The approach employs both a general problem-solving method and an automatic generation method for problem-solving programs. The problem-solving method is a combined method with backtracking and constraint propagation, based on the features of MCSPs. The automatic generation method analyzes the meaning of constraints and generates a problem-solving program, which is especially efficient for the given problem. Finally, the proposed approach is evaluated by several experiments including scheduling applications and well-known toy problems. Employing the two methods enables solving a hard MCSP in a reasonable time, merely by describing it in a declarative form.

1 INTRODUCTION

In recent years, scheduling problems are increasingly being viewed as constraint satisfaction problems (CSP) [Fox 1968]. The CSP has a simple well-defined model and general problem-solving algorithms. Since many scheduling problems can be formulated as CSPs, they can be solved by these algorithms, in a general way.

In order to solve a CSP, two classes of algorithms have been developed. The first group involves backtracking algorithms and the second group involves constraint propagation algorithms. Backtracking algorithms are guaranteed to solve any CSP, but they suffer from thrashing [Nudel 1983]. On the other hand, several network-based constraint propagation algorithms have been developed, using the topology of constraint networks [Mackworth 1977, Freuder 1982, Freuder 1990]. They are guaranteed to solve a class of CSPs with a tree-like constraint network in polynomial time [Mackworth 1988].

Recently, combined algorithms with backtracking and constraint propagation have been developed [Dechter 1988, Dechter 1990]. They use constraint propagation on tree-like subgraphs of a constraint network in order to reduce the search space for backtracking. They can solve any CSP and are efficient for tree-like constraint networks. They have been the most powerful algorithms to solve CSPs in a general way.

However, even with these algorithms, most scheduling problems are hard problems. This is because, they have a disjointness constraint, that prohibits assigning a same value (resource) to more than one variables (task). Since this constraint is concerned with every combination of two variables, their constraint networks become a complete graph, which is the most complex case. In general, a CSP is the more difficult, if the constraint network is the more complex [Zahfred 1990].

This is the most critical difficulty for solving scheduling problems in a general way.

In order to overcome this difficulty, the authors took an approach from two points of view:

1. Since there is no algorithm which solves any CSPs efficiently, a problem-solving method, based on the features of the application problems, is required.

2. The network-based algorithms are efficient only the topological features of the constraints. However, focusing on the meaning of constraints, there may be more efficient ways to process them.

This paper presents a novel approach to a class of CSPs. First, it defines Multi-Dimensional Constraint Satisfaction Problem (MCSP), which is a useful model applicable to many scheduling problems. A declarative framework to describe MCSPs is also presented.

Second, it proposes an approach for MCSPs. The approach employs a general problem-solving method for MCSPs and an automatic generation method for problem-solving programs.

The problem-solving method is a combined method with backtracking and constraint propagation. It is an efficient method, based on the features of MCSPs. The automatic generation method generates a problem-solving program, which is especially efficient for the given problem. It analyzes constraints in a logical form and selects efficient procedures, according to the meaning of constraints.

Finally, the proposed approach is evaluated by experiments including school curriculum scheduling, production scheduling, work assignment problems, and several well-known MCSPs. Employing the two methods enables solving a hard MCSP in a reasonable time, merely by describing it in a declarative form.

The following section defines the MCSP and presents the declarative framework to describe MCSPs. In Section 3, the problem-solving method for MCSPs is proposed. The automatic program generation method is described in Section 4. Section 5 shows and discusses the experimental results. A summary and conclusion are given in Section 6.

2 PROBLEMS

As described in the previous section, most scheduling problems belong to the most difficult class of CSPs. Therefore, the authors focused their attention on a class of CSPs, that is applicable to many scheduling problems. This section defines Multi-Dimensional Constraint Satisfaction Problem (MCSP) and presents a declarative framework to describe MCSPs.

2.1 Multi-Dimensional Constraint Satisfaction Problem

A CSP involves a set of N variables \( v_1, ... , v_N \) having domains \( D_1, ... , D_N \), where each \( D_i \) defines the set of available values for the variable \( v_i \). An MCSP is a CSP, in which all the domains are the same. Namely, \( D_1 = D_2 = ... = D_N \).

For example, a Four Color Problem is an MCSP. It is a problem to assign four colors on every bounded area on a plane, satisfying the constraint that no area has the same color as its neighboring area. This problem has variables for every area, and a shared domain that is the set of four colors.

Moreover, the domain may be represented by an F x J array with two (or more) dimensions. Figure 1 illustrates the MCSP variables and domain.

For example, a school curriculum scheduling problem, by which to assign a teacher and a time for every classroom, is a two-dimensional MCSP. In this case, variables are given classrooms. The domain is represented by an array with two dimensions corresponding to teachers and times. Another example is a production scheduling problem, that consists of \( N \) tasks, \( I \) production machines, and \( J \) time intervals in a scheduling period.

Many other scheduling problems, such as work assignment problems, can be formulated as MCSPs. Consequently, MCSP is an important subclass of CSPs for scheduling applications.

2.2 DECLARATIVE DESCRIPTION OF MCSPs

This section presents a declarative framework to describe MCSPs. An MCSP consists of a set of variables, a multi-dimensional domain, and a set of constraints.

For example, the declarative description of a Four Color Problem is presented in Fig. 2. Lines 1-2 define the sets of variables and a domain. Lines 3-4 define a class used in the constraint definition in lines 5-9. The problem is defined in lines 10-13. The body (lines 7-9) of the constraint definition is a logical form. The meaning of the constraint definition is that:
For every pair of different assignments (area color1) and (area color2), let the assignments be (area color1) and (area color2), if area1 and area2 form a pair of neighbors. color1 and color2 must be different, otherwise OK.

Lion 12 defines the shared domain for this MCSF. A Four Color Problem has a one-dimensional domain (a set color).

In case of school curriculum scheduling, the domain may have two dimensions (teachers and times involved). The problem definition may be as follows:

(define-problem school-scheduling (variables classroom) ((domain (teacher time)) ; 2-dimension (constraints ...))

The problem is to assign a value, in the two-dimensional domain made up with the sets teacher and time, to each variable in the set classroom, satisfying the all constraints specified in the constraints option.

3 A METHOD TO SOLVE MCSFPs

As described in Section 1, most scheduling problems belong to the multi-dimensional class of CSPs. Therefore, the authors developed an efficient method for MCSFPs, based on the MCSFP features. This section describes two MCSFP features and proposes a problem-solving method, based on the features.

3.1 FEATURES OF MCSFPs

As mentioned in the preceding section, an MCSFP has a multi-dimensional, single-constraint domain that consists of a set of dimensions. In the case of a multi-dimensional domain, the dimensions have independent meanings in the application problem, e.g., teachers and times. Therefore, many constraints refer only one dimension of the domain.

For example, a school curriculum scheduling problem has the following constraints:

science-classroom-science-teacher
A science teacher must be assigned to a science classroom.
same-class-different-time
Different times must be assigned to two classrooms of the same class.

Constraint science-classroom-science-teacher does not refer to the domain dimension time, but to the other dimension teacher. On the other hand, same-class-different-time refers only to time.

The constraints, which refer to only one domain dimension, are called one-dimensional constraints, while other constraints are called multi-dimensional constraints.

Since domain dimensions have independent meanings, most constraints are one-dimensional. This is an important feature of MCSFPs. The proposed method is based on this dimension independence of MCSFPs.

Another MCSFP feature is problem duality. Since an MCSFP has a two (or N) dimensional domain, the problem is assigning a two-dimensional value (i,j) to each variable vij. Therefore, it can be reformulated into another CSP, in which a (one-dimensional) value i is assigned to a variable vi, and j is assigned to vij. For example, since the school curriculum scheduling problem is assigning a value (teacher, time) to each classroom, it can be reformulated into another CSP with 2N variables, where N variables vij (classroom) for i values (teacher) and N variables vi (classroom) for j values (time).

This problem duality is also used in the problem-solving method for MCSFPs.

3.2 A PROBLEM-SOLVING METHOD BASED ON MCSFP FEATURES

The problem-solving method is based on the MCSFP features, dimension independence and problem duality.

The method decomposes an MCSFP into three (or more) subproblems, using problem duality. The subproblems are:

SP-M A subproblem, which is same as the original MCSFP, except that it has only multi-dimensional constraints.

SP-J A subproblem, which corresponds to the dimension i, with N variables vij, a domain with size Ji, and one-dimensional constraints that refer to i.

In the school curriculum scheduling problem, it has N classrooms as variables, I teachers as values, and one-dimensional constraints that refer to teachers, e.g., science-classroom-science-teacher.

SP-J A subproblem, which corresponds to the dimension j, with N variables vij, a domain with size Jj, and one-dimensional constraints that refer to only Jj.

In the example, it has N classrooms as variables, J times as values, and one-dimensional constraints that refer to only time, e.g., same-class-different-time.

Then, the method assigns values to variables using a backtracking algorithm and a constraint propagation algorithm, as follows:

Figure 3: A Problem-Solving Method for MCSFPs

1. A backtracking algorithm creates assignment on subproblem SP-M, checking multi-dimensional constraints.
2. A constraint propagation algorithm processes one-dimensional constraints on subproblems SP-J and SP-I.

In the example, one-dimensional constraints about times are propagated in SP-J.

The constraint propagation is triggered when the backtracking assigns a value to a variable. The backtracking selects the candidate values for a variable according to the constraint propagation results. Figure 3 illustrates the constraint propagation method.

Here, it should be noticed that the problem-solving method does not specify a certain backtracking algorithm or a constraint propagation algorithm. Existing backtracking algorithms can be joined into this method with small modification. Also, several constraint propagation algorithms can be combined in the problem-solving method. Current experimentation uses a most-constraint mini-conflicts backtracking algorithm, as described in [Kong 1989], and a naive constraint propagation algorithm [Kach 1977].

3.3 DISCUSSION

In order to evaluate the problem-solving method, how existing CSP algorithms work on an MCSFP must be considered. Several efficient algorithms have been developed using the topology of constraint networks, e.g., [Freuder 1982], [Mitchell 1985], [Dechter 1986], and [Dechter 1990]. They are applicable and efficient with tree-structure constraint networks. However, most scheduling problems have a disjointness constraint that makes a constraint network form a complete graph. Since they belong to the most difficult class of CSPs [Zaib 1990], these algorithms have few merits.

On the other hand, SP-I and SP-J in the proposed problem-solving method have small domain sizes I and J, while the domain sizes for the original MCSFP is IXJ. If we propagate e one-dimensional constraint edges on an MCSFP, using AC-3, then the complexity is O(eI+J+P) (See [Dechter 1988]). On the other hand, the complexity on SP-J and SP-I is O(eI+P+J), where e and P are number of edges on SP-I and SP-J, namely e = I+J. Consequently, the proposed method dramatically decreases the computational time.

Here, it must be considered carefully that SP-I and SP-J have only one-dimensional constraints. Since MCSFPs have dimension independence, most constraints are one-dimensional. However, if there are heavy multi-dimensional constraints in an MCSFP, the proposed method has few merits. This is the limitation of this method. An example of heavy multi-dimensional constraints is the disjointness constraint, which can be checked in a cheaper manner, using an IxJ array, as described in the next section.

In addition, the combination of local propagation and backtracking (LBSP) in [Craig 1988] is similar to the proposed method, except that it does not use domain dimensions.

4 AN AUTOMATIC PROGRAM GENERATION METHOD

This section describes an automatic program generation method. The method analyses the meaning of given constraints and generates appropriate procedures to process them. Then, it integrates them into a program to solve the problem. First, a naive program generation method is described. Then, a method to refine a constraint process is proposed.

4.1 A NAIVE PROGRAM GENERATION METHOD

The naive program generation method analyzes a given constraint and generates a constraint process procedure, as follows.

Step 1: A constraint is defined with a logical form. For example, the exclusive-color constraint for a Four Color Problem has the following form:

\[(\text{if } \overline{(\text{in-same-set-of-neighbors area, area})} \text{ and } \overline{(\text{area area})} \text{ then } \overline{(\text{neighbors pair})}) \text{ and } \overline{(\text{color color})})\]

...
4.2.2 Global Refinement Method

The method described above refines a propagation process for one constraint edge. On the other hand, it is possible to refine the propagation process for a set of constraint edges into an efficient procedure. This refinement is accomplished in almost the same manner, but it uses variable-forms, as well as value-forms.

Consider the same-class-different-time constraint for the school curriculum scheduling in Section 3. It has a value-form and a variable-form as follows:

Variable-form:

(value set-of classrooms-for-the-same-class classroom1 classroom2)

Value-form:

(value set-of classrooms-for-the-same-class classroom1 classroom2)

The constraint propagation from one variable to all the other variables in a classroom-for-the-same-class set is refined into a procedure:

PROC-3: Check whether or not the available value (value1) is unique. If unique, remove the unique value from the domain for all the other variables in a set (classrooms-for-the-same-class), otherwise do nothing.

This refinement reduces the complexity of O(m^2) into O(m), where m is the size of a set.

As described in Section 3, thorough checking of a disjoint constraint has large costs in the problem-solving method. The constraint checking can be replaced by the following procedure.

PROC-4: Provide an I x J Boolean array that expresses the domain. Look up an array entry in order to check the disjointness constraint. When backtracking assigns a value, mark a corresponding array entry in order to specify that the value is unavailable.

If there were no backtracking to assign values to all variables, PROC-4 reduces the total checking cost from O(n^4) into O(n^2). This refinement is more effective in general cases.

4.3 DISCUSSION

The effectiveness of the refinement method has already been shown. Here, the novelty and limitation for this method are discussed.

Guesgen’s CONSAT also provides a constraint description language and a constraint compiler, which improves constraint propagation processes [Guesgen 1987]. The compiler generates checking variables, which has no relation to a given constraint. However, it does not refine the constraint process with the related variables. The improvement of CONSAT compiler is similar to Step 4 of the naive program generation method in Section 4.1.

The most close research to the refinement method is an acc-consistency algorithm within AC-4 in [DeVito 1991]. AC-5 uses the features of functional and monotonous constraints in order to reduce the complexity. It is based on almost the same idea as the refinement method for constraint propagation, except that it does not handle disjunction constraints, such as disjointness. Moreover, AC-5 does not include the global refinement method or refinements for domain-value removal and constraint checking.

It is trivial that this refinement method is effective only when a value-form matches a provided pattern. This is the limitations of the method.

5 EXPERIMENTAL RESULTS

This section evaluates the proposed approach with several experiments including school curriculum scheduling, production scheduling, work assignment, and several well-known CSP problems (See Appendix).

The computational times used for a school curriculum scheduling and a production scheduling problem are shown in Table 1. For each problem, both a two-dimensional and a one-dimensional formalization are examined. A one-dimensional formalization represents the same problem as a two-dimensional one, except that the two-dimensional domain is elided into a one-dimensional domain. This elision causes that the proposed problem-solving method works in the same way as LPB in [Guesgen 1980] (See Section 3.3). Therefore, a comparison between one-dimensional and two-dimensional formalization shows the improvement by using domain dimensions. Here, the constraint process refinement method, proposed in Section 4, is not used, except PROC-4 for the disjointness constraint.

In the case of Problem A, using a multi-dimensional domain causes almost 80 times the previous efficiency. This marked result indicates the great effect of the proposed problem-solving method. On the other hand, the ratio is 1.78 for Problem B. This is because that Problem B is smaller than Problem A. As discussed in Section 3, the complexity of propagating one-dimensional constraints is O(n^2P^2) vs. O(nP^2) for the two-dimensional. Therefore, the method is more effective for a larger problem, namely larger values of I, I, and f. Consequently, the problem-solving method is more effective for a larger and more tightly constrained problem.

1 This is because the one-dimensional curriculum scheduling problem (Problem A^0) cannot be solved in four days, without PROC-4.
Table 1: Experimental Results of the Problem-Solving Method

<table>
<thead>
<tr>
<th>Problem</th>
<th>a. Two-dims. (seconds)</th>
<th>b. One-dim. (seconds)</th>
<th>Ratio (b/a)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Curriculum Scheduling</td>
<td>1,921</td>
<td>151,900</td>
</tr>
<tr>
<td>B</td>
<td>Production Scheduling</td>
<td>5.94</td>
<td>7.01</td>
</tr>
</tbody>
</table>

Table 2: Experimental Results of the Constraint Process Refinement Method

<table>
<thead>
<tr>
<th>Problem</th>
<th>a. Refined (seconds)</th>
<th>b. Naive (seconds)</th>
<th>Ratio (b/a)</th>
<th>Procedures</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Curriculum Scheduling</td>
<td>856.60</td>
<td>50,490</td>
<td>1.70</td>
</tr>
<tr>
<td>B</td>
<td>Production Scheduling</td>
<td>3.73</td>
<td>47.71</td>
<td>12.79</td>
</tr>
<tr>
<td>C</td>
<td>Work Assignment</td>
<td>82.48</td>
<td>87.62</td>
<td>1.06</td>
</tr>
<tr>
<td>D</td>
<td>WorkPatterns Assignment</td>
<td>1.96</td>
<td>2.38</td>
<td>1.22</td>
</tr>
<tr>
<td>E</td>
<td>S-R-Queue</td>
<td>148.29</td>
<td>1,110</td>
<td>7.49</td>
</tr>
<tr>
<td>F</td>
<td>Four Color Problem</td>
<td>217.60</td>
<td>598.60</td>
<td>2.80</td>
</tr>
<tr>
<td>G</td>
<td>Four Color Problem</td>
<td>0.42</td>
<td>10.34</td>
<td>24.60</td>
</tr>
<tr>
<td>H</td>
<td>Zebra Problem</td>
<td>0.41</td>
<td>0.60</td>
<td>1.44</td>
</tr>
</tbody>
</table>

Table 2 compares the computational times for the same problem in two cases: one is when the constraint process refinement method is applied, and only the naive program generation method is used in the other case. These results for each problem are taken from exactly the same problem definition. The procedures used in the refinement method are also listed in the table.

One of the most remarkable results is that the PROC-4, which refines checking the disjunctive constraint, has a great advantage (in Problem A, B, and E). Comparing the results of D and F, since PROC-4 (in Problem II) is a global refinement procedure, the gains are larger than a local refinement procedure PROC-1 in Problem D. As discussed in Section 4.3, AC-5 does not include refinement of constraint checking and global refinement method. Consequently, the proposed refinement method is more effective than merely employing AC-5.

Consider the Four Color Problems (F and G). Since the unique constraint is refined by PROC-4, the gains for the refinement method are comparatively large. In the case of Problem G, an arc consistency algorithm as AC-3 proves that it has no solution without any backtracking. Namely, almost all the time is spent in constraint propagation. Therefore, the refinement method causes a high efficiency.

In addition, it should be mentioned that the same procedures are used in several problems involving different application fields. This means that the constraint process refinement method is applicable for many applications.

Although the constraint process refinement method refines only a part of constraint processes, it has considerable merits for many application problems. Moreover, it is very effective, especially for highly constrained problems. Consequently, two proposed methods enable solving a hard problem in a dramatically efficient way.

6 CONCLUSION

In this paper, a novel approach to a class of CSPs is presented. First, it defines Multi-Dimensional Constraint Satisfaction Problem (MCSP) which is a useful model applicable to many scheduling problems. Second, it proposes an approach for MCSPs, focusing on the features of MCSPs and the meaning of constraints. The approach employs a general problem-solving method for MCSPs and an automatic generation method of problem-solving programs. Finally, the proposed approach is evaluated by several experiments including scheduling applications and well-known toy problems. Employing these methods enables solving a hard MCSP in a reasonable time, merely by describing it in a declarative form.

Appendix: Test Problems

A school curriculum scheduling problem is presented as an example. It has 160 variables (32 classrooms for every 5 classes) and 352 values (11 teachers × 32 times). The constraints are:

- science-classroom-science-teacher,
- same-class-different-time, disjunctive, a constraint specifying continuous classrooms, a constraint specifies that classrooms in the same class and the same subjects must not be assigned in the same day, etc.

A' The same problem as A, except that the two-dimensional domain (11 × 32) is dissolved into a one-dimensional domain (352).

B A job-shop production scheduling problem with 44 variables (tasks), 6 production machines × 10 time-intervals. It is a very simple test problem developed for experimental purpose. It includes specification regarding off-days and scheduled machine maintenance, relation among a task and a machine, due dates, task orderings, and a few constraints.

B' The same problem as B, except that the two-dimensional domain (6 × 10) is dissolved into a one-dimensional domain (60).

C A work assignment problem with 114 variables (works) and one-dimensional domain (21 workers). Since the time of a work is given, there are no time dimension for the domain. Constraints are, exclusive assignments for the same work time, specification of workers' available times, license for workers, standard working time length, etc.

D The same problem as C, except that the 114 works are preprocessed and combined into 23 work-patterns.

E N-five problems. They have N variables (rows) and a one-dimensional domain with N values (columns).

F A Four Color Problem with 560 variables (areas), 4 values (colors), and 1,563 pairs of neighboring areas. It is a one-dimensional problem.

G A Four Color Problem. It has 152 variables (areas) and 413 neighborhoods. Since the available set of colors for each area is restricted, it has no solution.

H Zebra Problem in [Dechter 1990]. It is a one-dimensional problem with 25 variables (5 cigarettes, 5 pets, 5 persons, 5 houses, and 5 druids) and 3 values (positions). It has only one solution.

Acknowledgment

The authors would like to express their thanks to Tatsuo Ishiguro, Masahiro Yamamoto, Takeshi Yoshizumi, Masaharu Watanabe and Tomyo Kai for their encouragement in this work. Further, they also thank Yoshiyuki Koseki for his valuable advice.

References


