Constraint Modeling and Reformulation in the Context of Academic Task Assignment

Robert Glaubius  
*University of Nebraska-Lincoln, glaubius@cse.unl.edu*

Berthe Y. Choueiry  
*University of Nebraska-Lincoln, choueiry@cse.unl.edu*

Follow this and additional works at: [http://digitalcommons.unl.edu/cseconfwork](http://digitalcommons.unl.edu/cseconfwork)

Part of the [Computer Sciences Commons](http://digitalcommons.unl.edu/cseconfwork)

This Article is brought to you for free and open access by the Computer Science and Engineering, Department of at DigitalCommons@University of Nebraska - Lincoln. It has been accepted for inclusion in CSE Conference and Workshop Papers by an authorized administrator of DigitalCommons@University of Nebraska - Lincoln.
Constraint Modeling and Reformulation in the Context of Academic Task Assignment
Robert Glaubius and Berthe Y. Choueiry

Abstract: We discuss the modeling and reformulation of a resource allocation problem, the assignment of graduate teaching assistants (GTAs) to courses. Our research contributes the following:

- Formulation of the GTA assignment problem as a nonbinary CSP.
- Design of a new convention for consistency checking to deal with over-constrained problem.
- Definition of a new network-decomposable nonbinary confinement constraint.
- Evaluation of the reformulation of confinement and equality constraints on 3 real-world data sets.

Benefits of automation: task previously solved manually, which was costly and time consuming. We have designed and developed a prototype that has been notably beneficial to our department.

- Reduced the number of assignment conflicts.
- Increased course quality.
- Decreased time and effort of finding a solution.

Definitions:
A Constraint Satisfaction Problem (CSP) is a triple \( \mathcal{P} = (V, \mathcal{D}, \mathcal{C}) \), where

- \( V = \{ V_1, V_2, ..., V_n \} \) is a set of variables.
- \( \mathcal{D} = \{ D_1, D_2, ..., D_n \} \) is the set of variable domains.
- \( \mathcal{C} = \{ C_1, C_2, ..., C_n \} \) is a set of constraints on variables in \( V \).

Problem Definition: In a given semester, given a set \( G \) of GTAs, a set \( V \) of courses, and a set of constraints on allowable assignments, find an assignment of GTAs to courses that is:

- Consistent - the assignment breaks no constraints.
- Satisfactory - maximize the number of courses covered and the happiness of the assigned GTAs.

Courses: We model courses as variables in our CSP. There are 3 types of courses offered: lectures, labs, and recitations. Additionally, these courses may be offered during the entire semester, or only during the first or last half. Lectures usually require a GTA grader, while labs and recitations require an instructor.

GTAs: GTAs make up the domains of the variables. A GTA may serve as an instructor only if he or she is ITA certified. Each GTA also specifies his or her preference on a scale from 0 to 5 for each course offered in a given semester.

New Consistency-checking convention: Typically, these problems are overconstrained. We choose to assign null to variables when no GTA can be assigned. A solution is consistent when all non-null assignments satisfy all of the constraints.

Reformulation - confinement: For a given confinement constraint \( C \), we define a set \( S \) called the confinement set. We want the set of GTAs assigned to variables in \( S \) to be disjoint from those assigned to the other variables in \( C \)'s scope. We reformulate each confinement constraint by placing a binary mutex constraint between every variable in \( S \) and every variable in scope\( C \)'s.

Reformulation - equality: Since we allow null assignments, we must decompose the non-binary equality constraint into a clique of binary equality constraints.

Experiments: We experimentally evaluate the value of these reformulations on three data sets. These sets are described below. Our experiments involved four tests on each data set. Each test involved either static or dynamic least domain variable ordering, and processed either the nonbinary model using nonbinary forward checking nF2C [1] or the binary model using FC. Search runs for 1 hour and returns the best solution discovered.

Results: For every pair of tests on the same data set and ordering, the same best solution was found. In fact, the same number of nodes was visited by each search while finding these solutions. An 8% to 22% reduction in CPU time needed to find this solution is observed on the binary decomposed problem. The mean reduction is about 17%. Note that fewer constraints checks are made when searching the binary problem when finding the same solution.

<table>
<thead>
<tr>
<th>Data Set</th>
<th>Number of GTAs</th>
<th>Number of Courses</th>
<th>Total number of tests</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spring 2001</td>
<td>25</td>
<td>44</td>
<td>97</td>
</tr>
<tr>
<td>Fall 2001</td>
<td>34</td>
<td>47</td>
<td>97</td>
</tr>
<tr>
<td>Fall 2000</td>
<td>31</td>
<td>45</td>
<td>97</td>
</tr>
</tbody>
</table>

Constraints: We have elicited 4 unary, 1 binary, and 3 nonbinary constraints:

- ITA Certification - GTA must be ITA certified to teach the constrained course.
- Enrolment - GTA cannot be enrolled in the constrained course.
- Overlap - GTA cannot be assigned to a course that requires an instructor if he or she is enrolled in a course at the same time.
- Zero preference - GTA cannot have a preference of 0 for the course.

Unary:
- Equality - all courses should be assigned the same GTA.
- Capacity - no GTA should be assigned to a workload that exceeds his or her capacity.
- Confinement - assignments to two specific sets of courses should be mutually exclusive.

<table>
<thead>
<tr>
<th>CSP</th>
<th>Search running for one hour</th>
<th>Quality of best solution found</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data</td>
<td>[Vars]</td>
<td>[Vals]</td>
</tr>
<tr>
<td>Spring 2001</td>
<td>69</td>
<td>25</td>
</tr>
<tr>
<td>Fall 2001</td>
<td>65</td>
<td>34</td>
</tr>
<tr>
<td>Fall 2002</td>
<td>71</td>
<td>31</td>
</tr>
</tbody>
</table>

These results reaffirm the superiority of dynamic variable ordering, as dynamic least domain (SLD) consistently finds a better solution than static least domain (SLD) on the same data set.

