2002

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

Constraint Modeling and Reformulation in the Context of Academic Task Assignment

Robert Glaubius and Berthe Y. Choueiry

Constraint Systems Laboratory • Department of Computer Science and Engineering, University of Nebraska-Lincoln • {glaubius|choueiry}@cse.unl.edu

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 noticibly 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 $\mathcal{P}$ = ($\mathcal{V}$, $\mathcal{D}$, $\mathcal{C}$), where
- $\mathcal{V}$ = $\{V_1, V_2, \ldots, V_n\}$, a set of variables.
- $\mathcal{D}$ = $\{D_{V_1}, D_{V_2}, \ldots, D_{V_n}\}$, the set of variable domains.
- $\mathcal{C}$ = $\{C_i, C_{j,k}, \ldots, C_{i,j,\ldots,m}, \ldots, C_n\}$, a set of constraints on variables in $\mathcal{V}$.

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.

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

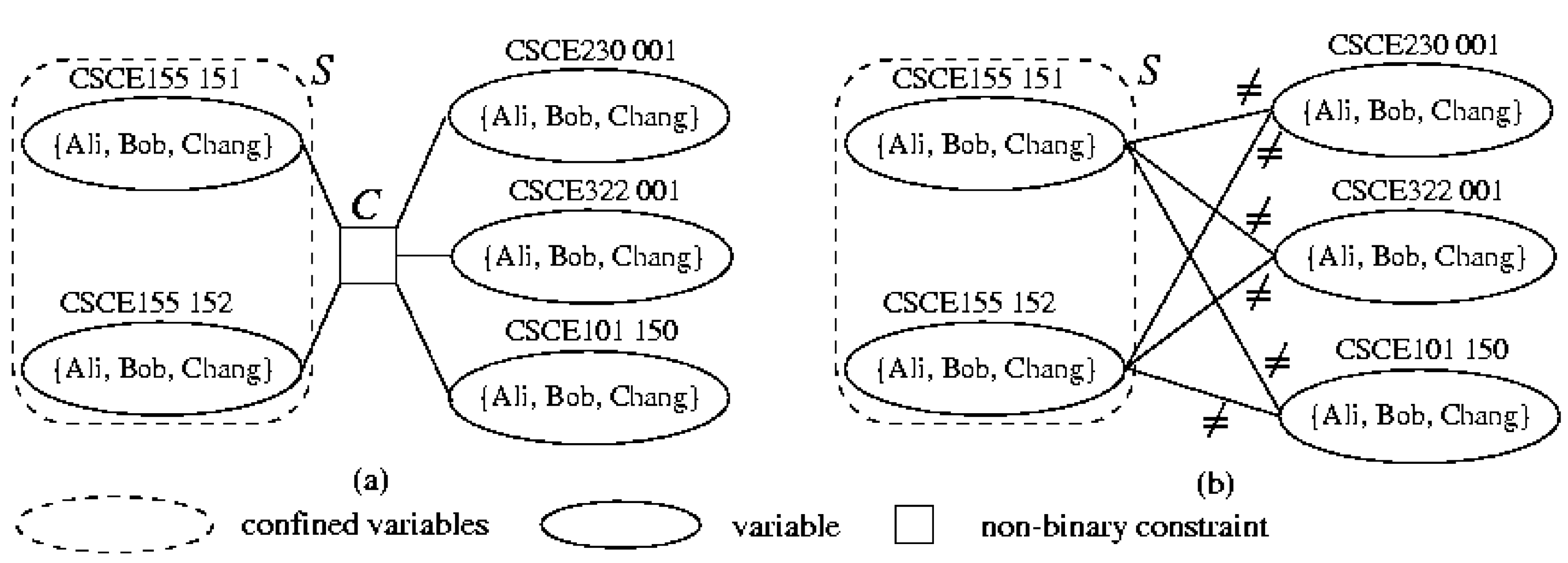
Unary
- *ITA Certification* - GTA must be ITA certified to teach the constrained course.
- *Enrollment* - GTA cannot be enrolled in the constrained course.
- *Overlap* - GTA cannot be assigned to a course that requires an instructor if he or she is enrolled in a course at the same time.
- *Zero preference*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

Binary
- *Mutex* - Courses cannot be assigned the same GTA

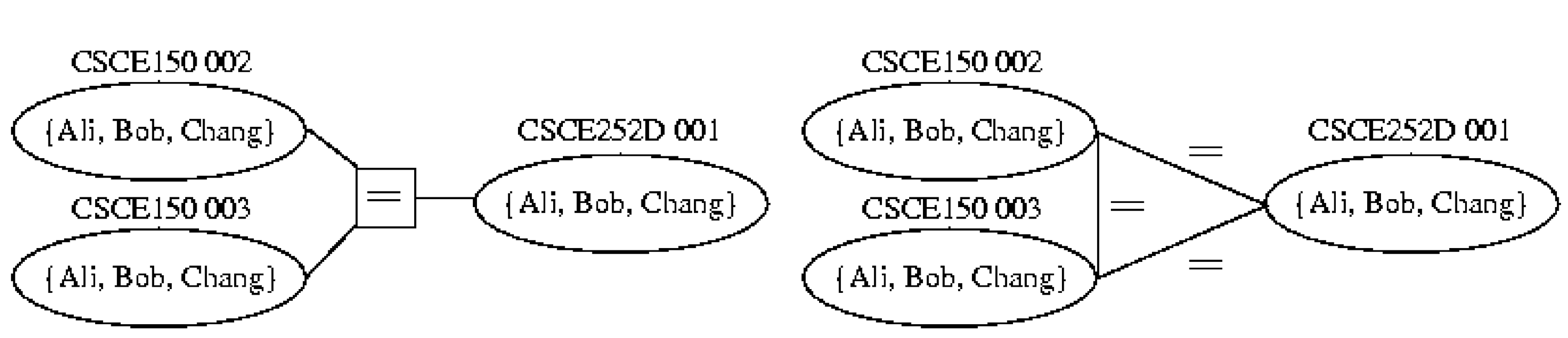
Non-binary
- *Equality* - all courses should be assigned the same GTA.
- *Capacity* - no GTA should be assigned to a workload that exceeds his or her capacity.
- *Confinement* - assignments to two specific sets of courses should be mutually exclusive.

Reformulation of nonbinary constraints: A constraint is network decomposable [2] when it can be represented by an equivalent network of binary constraints. We propose network decompositions for confinement and equality constraints. Under these decompositions, *since we allow* `null` *assignments*, nonbinary forward checking nFC2 [1] collapses to FC on the decomposition.

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 $scope(C) \setminus S$.



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 nFC2, or the binary model using FC. Search runs for 1 hour and returns the best solution discovered.

	Data Set		
	Spring 2001	Fall 2001	Fall 2002
Number of GTAs	25	34	31
Total number of courses	77	81	77
Lecture	44	47	45
Lab	24	24	24
Recitation	3	3	2
Half-semester	6	7	6
Number of equality constraints	3	3	10
Average arity	5	5.667	3.4
Number of capacity constraints	50	68	62
Average arity	63	58	65
Number of confinement constraints	12	16	14
Average arity	63	58	65
Average confinement set size	3.333	4.375	4.857

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.

CSP			Search running for one hour					Quality of best solution found			
Data	|{Vars}|	|{Vals}|	Order	Model	|Sol|	GTA Unused	GTA Available	CC	NV	Time (ms)	GeoMean
Spring 2001	69	25	SLD	binary	49	0	2.5	1208257106	514389	2463680	3.806217
				non-bin	49	0	2.5	1424663866	514389	2848450	3.806217
			DLD	binary	51	0	2.5	400736550	84423	614080	3.673231
				non-bin	51	0	2.5	400998214	84423	673020	3.673231
Fall 2001	65	34	SLD	binary	56	0	1	77809896	112	30630	3.167192
				non-bin	56	0	1	97854466	112	38970	3.167192
			DLD	binary	56	0	1	82827924	64	33360	3.354575
				non-bin	56	0	1	104189982	64	42630	3.354575
Fall 2002	71	31	SLD	binary	54	0	3.6	76231798	70	24570	3.564383
				non-bin	54	0	3.6	92933223	70	31520	3.564383
			DLD	binary	57	0	3.15	225355613	22560	255170	3.451227
				non-bin	57	0	3.15	252293613	22560	295790	3.451227

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

[1] C. Bessière, P. Meseguer, E. Freuder, J. Larrosa. On forward checking for non-binary constraint satisfaction. *Principles and Practice of Constraint Programming* (*CP'99*), pp. 88-102, 1999.

[2] I. Gent, K. Stergiou, and T. Walsh. Decomposable constraints. *Artificial Intelligence*, 12:133-156, 2000.

[3] R. Glaubius and B.Y. Choueiry. Constraint Modeling and Reformulation in the Context of Academic Task Assignment. ECAI 2002 Workshop on Modelling and Solving Problems with Constraints, 2002.

University of Nebraska Lincoln

This work was supported by a NASA Nebraska Space Grant, a Layman award, the Constraint Systems Laboratory, and the Department of Computer Science and Engineering at UNL, We acknowledge the help of Marilyn Augustyn in providing the data, and Christopher Hammack for building the web interface for data collection currently in use.