Partial Redundant Modeling

Student: Tiziana Ligorio
Supervisor: Susan L. Epstein

The Graduate Center of the City University of New York,
Computer Science Department
365 Fifth Avenue, New York, NY 10016-4309, USA
tligorio@gc.cuny.edu

Abstract. In the context of previous work on redundant modeling, permutation problems and matrix modeling, we introduce the notion of *partial redundant modeling* and *categorical channeling constraints*. To do so, we look at problems with categorical structure. Categorical channeling constraints also introduce the more general notion of channeling constraints over global constraints in redundant models. This paper provides only the motivation for partial redundant modeling and channeling categorical constraints. Future work will include implementation and testing as well as checking for scalability and flexibility of the proposed technique.

1 Introduction

Considering a problem from different points of view may provide additional inference power. Redundant modeling combines different models of the same problem using channeling constraints [1]. Channeling constraints allow different formulations of a problem to interact, propagating the constraints between different formulations. This can result in a significant improvement in performance.

Finding alternative models for a problem is not always a simple task. Nonetheless, some common CSP problem types have been identified for which finding an alternative model is fairly straightforward. One such problem type is a *permutation problem* [5]. This is a problem that has the same number of variables and values, where all the variables have the same domain and no two variables can be assigned the same value. Other constraints in the problem determine which permutations are acceptable as solutions. A permutation problem formulation (the primal model) can be translated into its *dual* by interchanging the variables with the values. The 0/1 version of a permutation problems can also be formulated as a *matrix problem* [3]. It is often the case in CSP that problems can be formulated as a matrix of decision variables. In a matrix problem some (or all) of the rows can be interchanged with some (or all) of the columns, thus different formulations of the same problem are also readily obtained in this generalized case.

Originally, work on redundant modeling assumed that redundant models must fully characterize the problem [1]. Later, Smith argued that only the primal model need fully characterize the problem, while the dual model need only have all the dual variables and channeling constraints between the two models (a *minimal combined model*) [5]. A primal model plus two redundant partial models have also been used to represent all the variables twice [2].

This paper extends the notion of a minimal combined model. It proposes the omission of some dual variables as well as the dual constraints, resulting in what we call *partial redundant modeling.* It also suggests that problem solving may benefit from more than two points of view, as practiced so far in redundant modeling. Partial redundant models originate in problems with a *categorical structure*, where the variables may be subdivided into categories. Often these categories can be identified as groups of variables that fall under n-ary constraints that partition the variables into disjoint sets. Real world problems, such as scheduling and rostering, may also have categorical structure. *Logic puzzles* are a class of problems with a simplified version of categorical structure. The advantage of such simplified (both in size and regularity of structure) problems is in the ease of exposition and investigation for possible future extension to more general categorical structures. We also introduce *categorical channeling constraints*, channeling constraints between global constraints in different partial redundant models.

This paper only provides an exposition and motivation for partial redundant modeling and categorical constraints. Future work includes implementation and testing, as well as the exploration of the scalability and flexibility of the proposed technique. In the next section we introduce logic puzzles, and introduce the *cross-hatch table*, a representation often used for solving logic puzzles. The third section relates the cross-hatch table to permutation problems, as well as matrix and redundant modeling, and introduces partial redundant modeling. Finally we provide thoughts for future work.

2 Logic puzzles

A logic puzzle consists of a set of *objects*, a set of *categories* (same-size disjoint subsets of those objects), a set of *semantic relations*, which specify the relationships that hold between the categories, and a set of clues to the solution. From these clues and the semantic relations, new clues can be inferred, until the problem is solved. A CSP consists of a set of *variables*, a set of possible values for each variable (its *domain*), and a set of *constraints* that restrict the values these variables can assume. Logic puzzles can be modeled as CSPs with a single solution and a categorical structure. In a logic puzzle, each category can be viewed as a subset of the CSP variables under an all-diff constraint [4], or as $n(n\text{-}1) / 2$ not-equal binary constraints, where n is the size of the category. One category can be chosen as the domain; the semantic relations along with the initial clues specify the constraints.

Consider, for example, the Relay Relativity puzzle: “The Sontags and three other father-daughter teams competed in a relay race sponsored by the Garden Spring Girls’ Club. From the following clues, determine the full names of each father-daughter team (one daughter’s name is Lisa) and the order in which the four teams finished. Ed and his daughter did not finish last; the Ahns did not finish first; Joyce finished either third or last. Gary and his daughter, who finished before the Carters, did not finish first; Inez finished either first or second. Hank and his daughter finished neither first nor last; Karen did not finish first. Frank and his daughter finished immediately after the Pizzi team, which does not include Hank. Inez and her father finished immediately before Joyce and her father." This puzzle includes 4 categories (Father, Daughter, Last name, finish Order). Each category has 4 objects, for a total of 16 objects. The implicit semantic relations here are that each team is made up of one father and one daughter, and that they both have the same last name.

Casting a logic puzzle as a CSP means that it can be modeled as a matrix. However, with such a representation, the concept of category is easily lost, since all the variables occupy the rows of the matrix without clustering them by category. The all-diff constraints suggest the category subdivision, but do not capture the relationships between the categories.

3 Cross-Hatch Table

A cross-hatch table is a common representation used to solve logic puzzles. Assume that objects in a problem are partitioned into m categories of size n. A *submatrix* is a matrix whose rows and columns are labeled by the objects in a category. The dual of a submatrix A is a new submatrix B in which the rows and columns of A have been interchanged. A *category matrix* is an $m \times m$ matrix each of whose entries is a submatrix. A *cross-hatch table* is a partial category matrix; it includes only those submatrices whose rows and columns are labeled by different categories, and it excludes the duals of its submatrices. The cross-hatch table T for our example appears in Figure 1.

		FATHER				DAUGHTER				ORDER			
		Ed	Frank	Gary	Hank	Inez	Joyce	Karen	Lisa	first	second	third	fourth
LAST	Ahn									0			
	Carter			0						0			
	Pizzi		0		0								0
	Sontag												
ORDER	first		0	0	0		0	0					
	second					1	0						
	third					0	1						
	fourth	0		0	0	0							
DAUGHTE	Inez												
	Joyce												
	Karen												
	Lisa												

Fig. 1. Cross-hatch table T for the Relay Relativity logic puzzle with initial entries from the clues in the problem definition.

In the remainder of this paper we will refer to individual entries in the submatrices of T as T(a,b), where a is the submatrix-row label and b is the submatrix-column label. A 1 entry in a submatrix indicates that the objects labeling the corresponding row and column are related to each other. For example, in Figure 1, T(Third, Joyce) = 1 indicates that Joyce and her father arrived in the third position. Two types of inference are intrinsic to the cross-hatch table representation. The first type is *exclusion/elimination*. Since each row and column in a submatrix must have exactly one 1 entry, zeros can be inserted in the remaining entries in that row and column (within the submatrix) by elimination. Similarly, in an $n \times n$ submatrix, if there are n-1 zeroes in a row or column, the remaining entry must be a 1 by exclusion. The second type of inference inherent in the cross-hatch table representation is *submatrix row/column duplication*. T(A, B) = 1 means that A and B correspond to each other. Therefore, everything that is known about A can be inferred for B. This is easily reflected by duplicating every submatrix row or column corresponding to A into every row or column corresponding to B in another submatrix and vice versa. For example, if T(First, Lisa) = 1, we can duplicate the row or column for First in the Order-X submatrix, where X is some category other than Daughter, into the row or column for Lisa in the Daughter-X submatrix. Figure 2 shows T after all exclusion/elimination and submatrix row/column duplication on Figure 1. Note the amount of added information in the table obtained only through inference.

4 Cross-Hatch Table and CP Modeling

An intuitive CSP model for our problem would be to select the objects in one category as the domain values, and take all other objects in the other categories to be the variables. Four different models can be obtained this way, each choosing a different category as the domain values. We will refer to one of these models arbitrarily as the

		FATHER				DAUGHTER				ORDER			
		Ed	Frank	Gary	Hank	Inez	Joyce	Karen	Lisa	first	second	third	fourth
LAST	Ahn	**0**							**0**	0			
	Carter	**0**		0					**0**	0			
	Pizzi		0		0			**0**					0
	Sontag												
ORDER	first	*1*	0	0	0	*0*	0	0	*1*				
	second	*0*	*0*			1	0	*0*	*0*				
	third	*0*	*0*			0	1	*0*	*0*				
	fourth	0	*1*	0	0	0	*0*	*1*	*0*				
DAUGHTER	Inez	**0**	**0**										
	Joyce	**0**	**0**										
	Karen	**0**	**1**	**0**	**0**								
	Lisa	**1**	**0**	**0**	**0**								

Fig. 2. Cross-hatch table for the problem of Figure 1 after exclusion/elimination (italics) and submatrix row/column duplication (bold) inference.

primal model, on which all the constraints of the problem are expressed. Four additional models of the problem can be obtained by converting each of these into their 0/1 equivalent. The sub-matrices in the first row of the cross-hatch table correspond to a CSP 0/1 matrix representation of the problem, where the objects in the Father, Daughter and Order categories are the variables, and the objects in Last are the values. In the full category matrix, the middle row and the bottom row each correspond to alternative but equivalent 0/1 models of the same problem.

The cross-hatch table closely resembles CSP matrix redundant modeling. However, redundant modeling typically uses one primal model on which all the constraints are specified, and one dual model with channeling constraints. The cross-hatch table maintains multiple points of view on the problem to retain information about the categories and their relationship. In our problem there are four 0/1 alternatives to the primal model. Using all of them would, however, be overly redundant; too much repeated information would add nothing to the inference process; it would simply increase the expense of propagating channeling constraints over all these models. Given the categorical structure of the problem, we can use portions of multiple redundant models to gain propagation efficiency. The cross-hatch table representation shows that redundant information is useful if we include only sub-matrices whose rows and columns are labeled by different categories and not include submatrix duals. In this way we retain information about the relationship between the categories, and obtain greater propagation power via channeling constraints between the submatrices while keeping redundancy to a minimum.

The exclusion/elimination inference process of the cross-hatch table amounts to propagating an all-diff constraint over the variables in a category (the subproblems). Although all-diff constraints must be included in all the models (primal as well as partial redundant models), these are optimized constraints.

Walsh shows that all-diff constraints in the primal model have greater propagation power than arc consistency on the channeling constraints [6]. He also demonstrates that arc consistency on channeling constraints has greater propagation power than arc consistency on not-equal constraints in the primal model. Thus, in situations where the efficiency of global constraints cannot be exploited, redundant modeling and channeling constraints can be an improvement.

Given the categorical structure of logic problems, we can exploit both these results through partial redundant modeling. We can use the all-diff constraint on the submatrices (the partial redundant models), but use redundancy and the power of channeling constraints between these partial submodels. The added power of the partial redundant model is that of exploiting submatrix row/column duplication borrowed from the cross-hatch table. In the CSP version, the all-diff constraint determines the category subdivision. Therefore, submatrix row/column duplication in CSP amounts to adding channeling constraints between the all-diff constraints in the redundant partial models. These channeling constraints make certain that value assignments under the all-diff constraint in one partial model are reflected under an all-diff constraint in some other partial model. We call these *categorical channeling constraints*.

In problems with a categorical structure determined by all-diff constraints on sub-problems, we can exploit the propagation power of partial redundant models by imposing categorical channeling constraints over the all-diff constraints of the primal and partial redundant models. Implementation and complexity details of categorical channeling constraints will be considered in future work.

5 Future Work

We have defined partial redundant modeling for problems with categorical structure. Partial redundant modeling introduces the idea of using categorical channeling constraints between the all-diff constraints. This concept can be generalized to explore channeling constraints over other global constraints. This paper provides only an initial insight on possible work to be done. Future work includes the implementation and testing of partial redundant modeling and categorical channeling constraints for logic puzzles. Implementation and scalability of categorical channeling constraints needs to be tested. Channeling constraints over other kinds of global constraints will also be sought.

References

1. Cheng, B. M. W.,Choi, K. M. F.,Lee, J. H. M.,Wu, J. C. K.: Increasing Constraint Propagation by Redundant Modeling: an Experience Report. In: Constraints Vol. 4. (1999) 167-192.
2. Dotú, I.,del Val, A.,Cebrián, M.: Redundant Modeling for the QuasiGroup Completion Problem. In: Principles and Practice of Constraint Programming - CP 2003, Kinsale, Ireland (2003) 288-302.
3. Flener, P.,Frisch, A. M.,Hnich, B.,Kiziltan, Z.,Miguel, I.,Walsh, T.: Matrix modelling: Exploiting common patterns in constraint programming. In: International Workshop on Reformulating Constraint Satisfaction Problems, Cornell University, Ithaca, NY, USA (2002) 27-41.
4. Régin, J.-C.: A filtering algorithm for constraints of difference in CSPs. In: AAAI-94, Seattle, WA, USA (1994) 362-367.
5. Smith, B. M.: Modeling a Permutation Problem. In: ECAI'2000 Workshop on Modeling and Solving Problems with Constraints, Berlin, Humboldt University (2000).
6. Walsh, T.: Permutation Problems and Channelling Constraints. In: 8th International Conference on Logic for Programming, Artificial Intelligence and Reasoning (LPAR 2001), Havana, Cuba (2001) 377-391.