

Constraint Satisfaction for Case Adaptation

Nicoleta Neagu and Boi Faltings

Artificial Intelligence Laboratory (LIA),
Computer Science Department, Swiss Federal Institute of Technology (EPFL)
CH-1015 Ecublens, Switzerland
{neagu, faltings}@lia.di.epfl.ch <http://liawww.epfl.ch/>

Abstract. Case adaptation is a complex problem for which no general method has been found. We consider the restricted domain of problems which can be formulated as *constraint satisfaction*, and propose a general method using dimensionality reduction based on constraint solving and interchangeability.

Keywords: case-based reasoning, case adaptation, constraint satisfaction problem, interchangeability

1 Introduction

We are developing a java-based framework for case-based reasoning (CBR) on constraint satisfaction problems (CSP). At the moment, the main focus of this project is on case adaptation. Several researchers have combined CBR and CSP, often with the goal of supporting the adaptation process. A good example is COMPOSER ([1]), a research prototype system aimed at exploring the use of CBR and CSP as applied to engineering design. In COMPOSER, the case-based reasoner was augmented with CSP techniques in order to achieve a more systematic adaptation process.

More similar to our approach is CADRE ([2]), a case-based reasoner for architectural design where constraints restrict numerical relationships among dimensions. CADRE introduced the concept of *dimensionality reduction*: before attempting to adapt a case, it constructs an explicit representation of the degrees of freedom available for adaptation. However, CADRE defined this approach only for numeric constraints.

In this paper, we show how to use the concept of *interchangeability* ([3]) to extend the idea of dimensionality reduction to discrete variables.

2 Framework

As an example of a problem where cases are used to solve constraint satisfaction problems, we consider product configuration ([4]). While admissible configurations can be precisely specified as a constraint satisfaction problem, each customer has particular needs that remain largely unformalized. Case-based reasoning can help to map properties of the customer to the implicit needs.

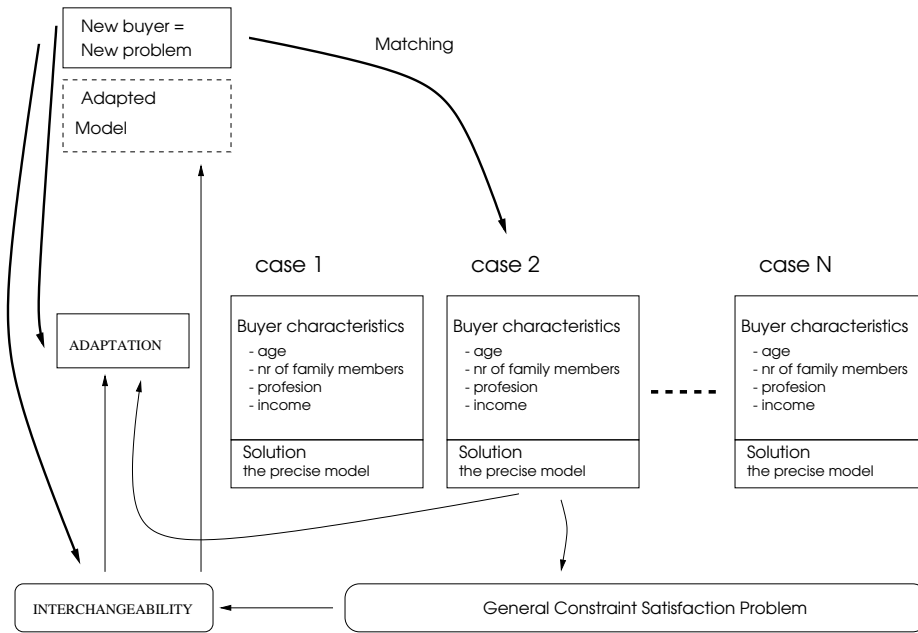


Fig. 1. Graphic representation of the framework

The framework for a system solving such configuration problems is shown in Figure 1. Cases represent past sales and consist of a description of a buyer as well as the product sold. Buyers are characterized by features such as age, profession, number of family members, and income.

Admissible configurations are modelled by a constraint satisfaction problem which is common to all cases. Interchangeability algorithms perform the dimensionality reduction that defines the possible adaptations that can be applied to a case. These are then matched to additional customer requests in order to construct an adapted solution. While this framework is tailored to the precise example of configuration, it is straightforward to adapt it to other constraint satisfaction problems (for example scheduling or planning).

3 Case Adaptation

Many approaches to case adaptation have considered all elements of a case to be variable, thus making adaptation as complex as solving the problem from scratch. Our approach, already used very successfully in CADRE, is to consider adaptation as two steps:

1. construct a model of the case that contains only variables that are both relevant and modifiable, we call this *dimensionality reduction*.
2. search for an adaptation in this reduced model.

The closer the case is to the current problem, the smaller the model for adaptation will be. Thus, adaptation will be much more manageable than if the base model was used. Furthermore, explicit construction of an adaptation model allows reusing cases in novel and creative ways.

The constraint satisfaction problem underlying all cases provides the knowledge of admissible modifications of a case. The difficulty is how to exploit it for computing models for adaptation. It is useful to consider separate mechanisms for *continuous* and *discrete* variables.

3.1 Constraints on continuous variables

Adaptation using constraints for continuous variables had been studied during the CADRE ([2]) project. Cases are instances of buildings along with a set of constraints on their well-formedness. CADRE introduced the idea of dimensionality reduction, which proved to give better and faster results than propagating changes locally. For dimensional adaptation, dimensionality reduction can be implemented using constraint solving (such as Gaussian elimination for linear constraints). A space of n parameters with m independent equality constraints then reduces to $(n - m)$ non-conflicting parameters.

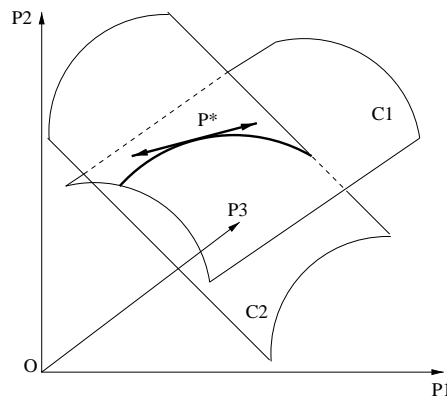


Fig. 2. Two constraints reduce to a single parameter P^*

As shown in Figure 2, there are three parameters and two constraints reduced to a single parameter P^* which models the intersection of the two constraints. Any modification by varying P^* is guaranteed to be consistent with all constraints.

3.2 Constraints on discrete variables

In the domain of discrete variables, the adaptation space is less obvious. In [5], they looked for interchangeability in subproblems induced from the original CSP by reducing the domains of a selected set of variables. Usually it is not possible to compute a single dimensionality reduction which is valid across the entire design space. We show

now how the concept of interchangeability allows us to nevertheless compute a dimensionality reduction for discrete variables.

The concept of Interchangeability formalises equivalence relations among objects or between values in a CSP problem. The concept of interchangeability was first introduced by Freuder in [3]. Among others, Freuder defines three kinds of interchangeability:

- values $x = a$ and $x = b$ are *fully* interchangeable if for any solution where $x = a$, there is an otherwise identical solution where $x = b$, and vice versa.
- values $x = a$ and $x = b$ are *neighbourhood* interchangeable if for every constraint involving x , for every tuple that admits $x = a$ there is otherwise an identical tuple that admits $x = b$, and vice-versa.
- values $x = a$ and $x = b$ are *partially* interchangeable with respect to a set of variables S if for any solution where $x = a$, there is another solution where $x = b$ which otherwise differs only in values assigned to variables in S , and vice versa.

Identifying of fully interchangeable values in principle requires computing all solutions of the CSP. However, values cannot be fully interchangeable without also being neighbourhood interchangeable. Neighbourhood Interchangeability (NI) only considers local interactions, and can be efficiently computed. Rather than computing full interchangeability, it is more efficient to first compute neighbourhood interchangeabilities, and filter those which are not actually valid in solutions. This is particularly applicable in our framework, since the cases will always give us a first solution and it is easy to test whether changing an interchangeable value gives us another solution. To identify neighbourhood interchangeable values we can construct discrimination trees. The leaves of the trees will be the equivalence classes of neighbourhood interchangeable values [6].

The adaptation model for a case thus consists of all variables which the customer might have to change, plus the interchangeabilities which are applicable to the case and define what can be changed.

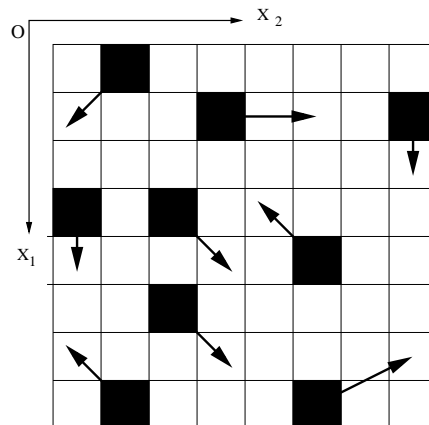


Fig. 3. Graphic representation of two dimensional discrete space

We take as the example a problem with two discrete variables X_1 and X_2 as in Figure 3. Black squares represent solutions given as cases. By applying neighbourhood interchangeability, we can find similar solutions which differ in only one dimension, shown graphically by horizontal or vertical arrows.

However, this severely restricts the possibilities: we might also want to adapt along a diagonal direction. This requires *partial* interchangeability: changes to a variable value which also requires changes in another one.

Partial interchangeability can be considered as full/neighbourhood interchangeability on a *set* of variables. Thus, it can be computed using the same algorithms. The difficulty, however, is to know what sets of variables to consider. In fact, the number of possibilities grows exponentially with the size of the considered sets, and a general search is clearly unmanageable.

A variable V affects the problem through the constraints that link V to other variables in the problem, thus through V 's neighbourhood $N(V)$. As observed in [6], if V is partially interchangeable with respect to S , then S must include at least one node of $N(V)$. Thus, it makes sense to compute partial interchangeability by considering increasingly large neighbourhoods of V . [3] gives an efficient algorithm for computing neighbourhood interchangeabilities in this context.

We will consider in the following a simple example of car configuration. The variables which describe the car and the constraints between them are shown in the figure below.

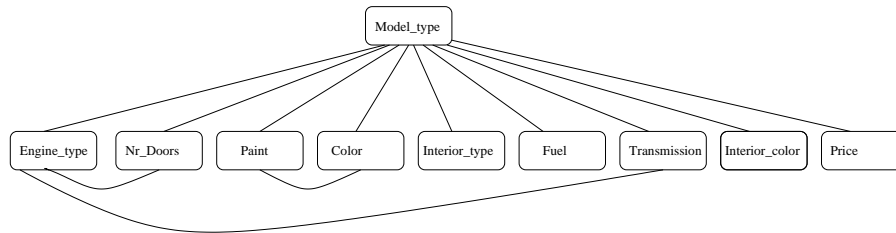


Fig. 4. Graph of the constraint network

Coming back to our application, a case is constructed by assigning one solution of the car configuration CSP to a well-defined buyer profile. When a new buyer manifests a request, the case database is browsed and the new buyer profile is matched to the closest one. The corresponding solution of the selected case is presented to the new buyer. For example, the system will the case 15(see table1). Possible changes for the current solution are presented in terms of interchangeable values applied to the car configuration variables.

In Table 1 we will present some results obtained for the constraint network considered. By applying the Neighbourhood Interchangeability algorithm for each variable we will get the corresponding neighbourhood interchangeable values. For example, for the variable Model-type, the values Opel-Tigra, Opel-Corsa and Opel-Sintra are neighbourhood interchangeable; that means that all the solutions which contains one of these

values will still remain solutions by exchanging with one of the other two values. So, a buyer has the option to choose the desired model form the displayed interchangeable values.

Attributes	Values	Neigh. Interch. Values	Partial Interch. Values
Model-type	Opel-Tigra	Opel-Corsa, Opel-Sintra	Opel-Omega/Engine-type, Opel-Astra/Color
Engine-type	E-75HorseP	–	E-90HorseP/Model-type
Nr-Doors	3Doors	–	–
Paint	Metallic	–	–
Color	Beige	Prune, Blue-Ber, Rouge	Gris-Quarts/Model-type, Blanc/Model-type Gris-London/Model-type
Interior-type	Standard	-	-
Transmission	Manual	-	-
Interior-color	Diasit-Rouge	Diasit-Blue,Houston-Blue Takana-Impala	-
Price	83000	–	–

Table 1. Case 15 with interchangeabilities

The results table also shows the partially interchangeable values obtained for variable Model-type. To obtain these values, the Partial Interchangeability algorithm has been applied on the set $S = \{\text{Model-type, Engine-type}\}$. In this case, when the buyer will change the value of a variable, the values of all the variables from the set have to change. So if our buyer would prefer the model Opel-Omega instead of Opel-Tigra the type of the engine will change.

4 Conclusions and Further work

We are implementing a prototype system which uses interchangeability to extend the dimensionality reduction idea to discrete-valued variables. We thus obtain a general and efficient case adaptation framework for the domain of constraint satisfaction problems.

In the future, we consider computing interchangeabilities specifically for a particular case adaptation. It is then possible to change the CSP and thus its interchangeabilities depending on the problem being solved. This general framework then allows very flexible open-world problem solving.

References

1. L. Purvis. *Intelligent Design Problem Solving Using Case-Based and Constraint-Based Techniques*. PhD thesis, University of Connecticut, 1995.
2. Kefeng Hua, Boi Faltings, and Ian Smith. Cadre: case-based geometric design. *Artificial Intelligence in Engineering*, pages 171–183, 1996.

3. Eugene C. Freuder. Eliminating interchangeable values in constraint satisfaction problems. In *In Proc. of AAAI-91*, pages 227–233, Anaheim, CA, 1991.
4. Rainer Weigel and Boi Faltings. Interchangeability for case adaptation in configuration problems.
5. Rainer Weigel, Boi Faltings, and Berthe Y. Choueiry. Context in discrete satisfaction problems. In *12th ECAIA*, pages 205–209, Budapest, Hungary, 1996.
6. Berthe Y. Choueiry and Guevara Noubir. On the computation of local interchangeability in discrete constraint satisfaction problems. In *Proc. of AAAI-98*, pages 326–333, Madison, Wisconsin, 1998.