From: AAAI Technical Report WS-98-15. Compilation copyright © 1998, AAAI (www.aaai.org). All rights reserved.

Interchangeability for Case Adaptation in Configuration Problems

Rainer Weigel, Boi V. Faltings and **Marc Torrens**
Artificial Intelligence Laboratory
Swiss Federal Institute of Technology in Lausanne (EPFL),
IN-Ecublens, CH-1015 Lausanne, Switzerland

Abstract

We address in this paper the adaptation of a case when a complete constraint model of the underlying problem is given. The idea is to apply methods from constraint based reasoning that allows the detection of "similar" solutions, which can be used to adapt a selected case to a new situation. We consider applications like configuration where a complete constraint model is available. For these applications, it is often advantageous to use a case-based approach because constructing a solution from scratch requires often more interaction than taking a complete solution (case) and adapt it to fit the current requirements.

Keywords: Case-Based Adaptation, Constraint-Based Reasoning

Introduction

We consider a solution space of a discrete CSP which can be nicely partitioned into sets of smaller spaces. Two solutions within the same set are quite "similar" while solutions from different sets are not "similar" where similarity can for example be measured by the number of values that are different in two solutions. Consider an elevator configuration problem where the class of electric elevators is very different from the hydraulic ones. Taking other concepts like building types into account one can synthesize summarized descriptions of the solutions within the different subspaces. The summarized descriptions correspond from a CSP point of view to the solutions of a sub-CSP induced by *primary* variables which represent "abstract" concepts of the underlying model. Similarly the summarized descriptions of a product can be found in every product catalog a salesperson uses to find a first rough description of the product envisaged by the customer. One can now build a case base containing for every partitioning a single complete solution of the CSP. Using such case base a clever salesperson can easily select a case which corresponds roughly to the customer's wish. Such a solution however will rarely satisfy all the requirements and must therefore be adapted. In many applications of case-based reasoning to design, automatic case adaptation relies on constraint satisfaction methods, like for example in Julia (Hinrichs 1988), Cadsyn (Maher *et al.* 1995) or Cadre and Faming (Faltings 1997). In this paper we focus on the case adaptation process by exploiting the concept of interchangeability of values of CSP variables introduced by Freuder in (Freuder 1991); extensions to context dependent interchangeability are described in (Weigel *et al.* 1996) and structuring techniques using meta interchangeability can be found in (Weigel and Faltings 1997). The basic idea is as follows: From the model based description, we extract knowledge of the form "The value a of a variable X can be replaced by a value b of X in a certain context C, but not in general". This knowledge can then be used for the adaptation of a case containing a, to move to a case containing b as long as the case is included in context C. In a car configuration example we have selected a case that contains the air-conditioning system $AC1$ which should be reconfigured because the customer would like to have system $AC2$. $AC1$ and $AC2$ however are interchangeable only when a large battery is available. Thus if the case already contains a large battery we can simply swap $AC1$ and $AC2$, otherwise we should first upgrade the battery and then replace the air-conditioning system. In the following, we recall the definition of a CSP and describe how to build the case base by finding some solutions which are considered as typical cases for the different possible classes of solutions. Then we recall the concepts of interchangeability and show how these concepts can be exploited when moving from one solution which corresponds to the adaptation process of solutions.

Cases and Constraints in Configuration Problems

Configuration is a design activity where the set of available components and their allowed combinations are

known a priori and the goal of the configuration process is to find the sets of components fulfilling the customers wishes and respecting all the compatibility constraints. Given such a problem with a complete constraint model, one could argue that there is no need for using case based methods if one can solve the whole problem indeed solely with CSP methods. However in configuration it can be very advantageous to add case-based methods because customers often cannot completely describe the product they really want. This is due to their incomplete knowledge about the product and also due to the complexity of the problem. The sales person's task is thus to propose a solution which is "close" to the customer's wish. Proposing a solution however can be considered as the case selection and a clever sales person will directly find a set of cases which could satisfy the customer wish to a certain extend. Without a case base the salesperson would have to generate constructively a solution by deciding first on certain parameters and starting a search engine. It could be that given the input parameters the system is over-constraint which is a situation every sales person tries to avoid, or the result is a huge number of possible solutions which are simply hard to overlook. Another good argument for using case based methods is that in some configuration applications it is much easier for a customer to say what he does not want instead of clearly stating what he wants. Thus proposing a solution and modifying it until all requirements are fulfilled is easier that constructing a complete solution from scratch.

We will now give a definition of a binary CSP together with a partitioning of the variables into primary and secondary parameters. A CSP defined by $P = (X, D, C, R)$; where $X = \{X_1, \ldots, X_n\}$ is a set of variables, $D = \{D_1, \ldots, D_n\}$ a set of finite domains associated with the variables, $C = \{C_1, \ldots, C_m\}$ a set of constraints, and $R = \{R_{ij} \subset D_i \times D_j$ for a constraint applicable to X_i and $X_j\}$ a set of relations that define the constraints. We divide the variables into a set of primary variables X_p and secondary variables X_s. With primary variables we can for example represent abstract concepts or functions of an artifact. Based on the definition, that design [Configuration] can be defined as the moving from a function description to an attribute description N. Murtagh described in an IMS working paper (GNOSIS Workpackage 4 1993) primary and secondary attributes of an artifact:

- Primary attributes relate directly to the required functions of an artifact, e.g., the primary attributes of a car enable it to perform the functions of transporting people and objects. The desire to have an artifact possessing certain primary attributes is what normally motivates the design in the first place.
- Secondary attributes are normally not essential, e.g. the ease of use of a car or optional features like air-conditioning, sun-roofs etc.

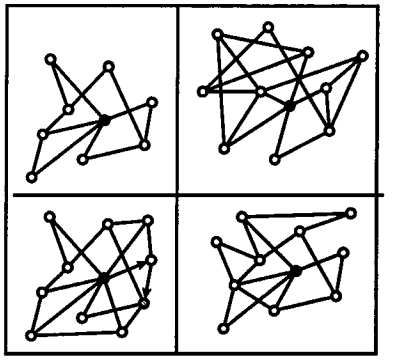

Figure 1: *Partitioning of the solutions in 4 sets. Adaptation of a case correspond to finding a path in the graph.*

One should note that secondary parameters are often used to implement the functions described by primary attributes. Given such a partitioning we can define the CSP P_p as the CSP induced by the primary attributes (variables) of the artifact. P_p contains only the variables X_p and the constraints in between variables in X_p. The solutions of P_p represents principally all possible combinations of functions a product could exhibit, however some of the solutions of P_p, which are in fact partial solutions of the overall problem P might not be extendible to a complete solution, and will thus be removed from the set of possible function configurations of the artifact. Every solution in this set together with a single extension to a global solution represents a case in the configuration case base. In Fig. 1 we see a small example where the solutions of the problem can be partitioned into four solution sets. Every solution is represented by a circle and the solutions that will appear in the case base are the filled ones. Two solutions are linked by an edge if they are "similar", that is they differ only in a small number of values. In the following we assume that a case base as described above is available and the salesperson together with the customer have selected a case which needs now to be modified until all of the customer's requirements are fulfilled. In constraint reasoning one can describe a selection of a case as a "jump" into a region of the search space where one hopes to find the customer's solution. Thus case adaptation can be considered as the reconfiguration of a given product (case). We consider the situation where the customer is not willing to change the values of the primary variables and thus only val-

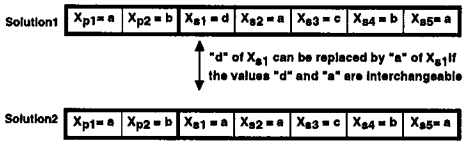

Figure 2: *Adaptation with Neighborhood Interchangeability.*

ues of secondary variables can be changed. Otherwise one would need to start again at the case selection.

Adaptation using Interchangeability

Using a CBR configuration system a clever salesperson can retrieve a case (product) from the case base which is close to the customer's wish and propose this product to the customer. A customer however is rarely satisfied and some modifications to the product need to be made. In this section we describe how make these modifications using model based transformation techniques based on the concepts of interchangeability. We consider a solution S of P which corresponds to the case selected and a CSP P_s which is obtained from the CSP P by removing the variables in X_p, furthermore we remove the values from the variable in X_s that are not compatible with the values of the primary variables X_p in S. We first recall the definition of interchangeability which formalizes equivalence relations between values in a CSP.

Definition 1 (Freuder) : *Neighborhood interchangeability:* A value b for a CSP variable V is neighborhood interchangeable (NI) with a value c for V iff for every constraint C on V:

$$\{i \mid (b,i) \textit{ satisfies } C\} = \{i \mid (c,i) \textit{ satisfies } C\}$$

A polynomial time algorithm ($O(n^2d^2)$) for computing all NI sets is described in (Freuder 1991). In the small example shown in Fig. 2 we can replace the values a and d for variable X_{s1} because the values are neighborhood interchangeable.

Even in structured problems NI sets are rather rare and thus other forms of interchangeability might be useful. In the next two subsections we show how context dependent interchangeability and meta interchangeability can be exploited for case adaptation.

Context Dependent Interchangeability (CDI)

Here, we recall the notion of context dependent interchangeable sets which is defined in terms of maximal cliques of a graph obtained from the microstructure of a CSP. This graph is called the *modified* microstructure μ_{mod} and is obtained by *adding* an edge in the

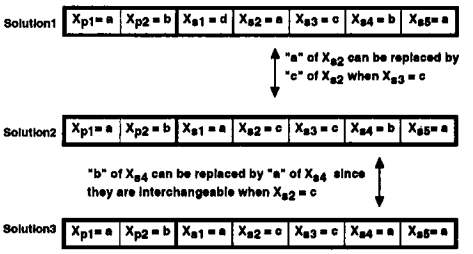

Figure 3: . *Adaptation with Context Dependent Interchangeability.*

original microstructure between all variable-value pairs of the same CSP variable:

$$\mu_{\text{mod}}(P) = (X_D, C_R \cup E_{\text{add}}) \text{ with}$$

$$E_{\text{add}} = \{ \text{ Edges between } (X_i, v_k) \text{ and } (X_i, v_l) | \; \forall (X_i, v_k), (X_i, v_l) \in X_D \text{ and } v_k \neq v_l \}$$

In the modified microstructure, maximal cliques of size bigger than n can be found. Maximal cliques that are of size bigger than or equal to n and contain variable-value pairs covering all variables, contain necessarily a set of solutions to the CSP and thus are called *solution cliques*[1].

Definition 2 *Context dependent interchangeability:* A value a for a CSP variable V is *context dependent interchangeable* with a value b for V, if and only if there exists a solution clique in the modified microstructure of the CSP that contain both nodes (V, a) and (V, b).

Unfortunately finding all solution cliques is computationally intractable, however approximations of CDI sets as described in (Weigel *et al.* 1996) can be found in polynomial time. The basic idea is to decompose iteratively the CSP and run Freuders algorithm on the subproblems. As a result one can identify interchangeability of the following form: The value b of variable X_{s4} is interchangeable with the value a if X_{s2} is either c and X_{s1} is a or b. Exploiting CDI is depicted in fig 3. Given *Solution*1 and the customer wants to replace X_{s4} a by b we realize that replacing the values can be done when X_{s2} is c which is currently not the case. Thus we first exchange b by c of X_{s2} which are CDI when $X_{s3} = c$ and then we swap a by b of X_{s4}. Adaptation can now be described as a transformation of a solution into another solution by successively exchanging values assigned to single variables. This could be accomplished using the solution graph $G = (V, E)$

[1]Note that two arbitrary solution cliques may contain some identical solutions but they differ in at least one solution.

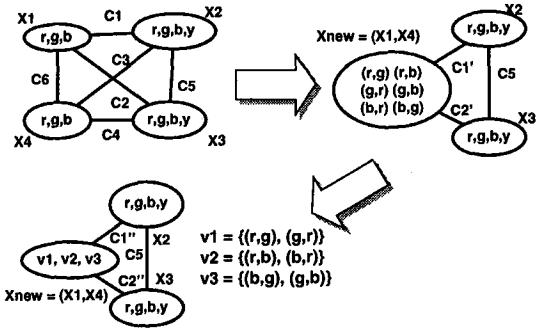

Figure 4: *Coloring example*

where V is the set of solutions of the CSP and there is an edge between two solutions if they differ in exactly one value (see Fig. 1). Each connected component G_i of G is a subset of solutions and any two solutions of G_i can be mapped onto each other by stepwise exchanging m values, where m is the length of the shortest path between the solutions in G_i. Unfortunately, building this graph requires us to enumerate all solutions. However given a single solution S' (the filled nodes in Fig. 1) and the sets of context interchangeable values, then we can build a connected subgraph $G_{cdi} = (V', E_{cdi})$ of G with: $S' \in V' \subseteq V$ and $E_{cdi} \subseteq E$ and any edge in E_{cdi} allows us to move in between the solutions connected by the edge simply by exchanging CDI values. Exploiting CDI for adaptation of solutions can now be described as follows: Given a single case S', construct the connected component of G_{cdi} containing S' which yields $m-1$ possible adaptations, where m is the number of nodes in the connected component.

Meta Interchangeability (MI)

Again, we are in the situation where a case has been selected and must be modified by changing values of the secondary parameters until the customer's requirements are fulfilled. Within this subsection we show that interchangeability might appear for combinations of several secondary variables. We call such combinations *meta variables.* The coloring example in Fig. 4 illustrates this fact. Assume that variables $X1$ and $X4$ are clustered into a new meta variable X_{new}. Then there are 6 values for X_{new} , namely the solutions of the subproblem induced by $X1$ and $X4$. $C1$, $C3$ and $C2$, $C4$ are merged into $C1'$ and $C2'$ respectively. The constraint $C6$ will no longer appear since it is already accounted in the value combinations allowed for X_{new}. The new structured value $(X1 = r, X4 = g)$ for X_{new} is then compatible with b and y for variable $X3$. It turns out that by calculating neighborhood interchangeability for values of X_{new}, we can find three equivalence classes each of size two, namely $v1 = \{(r,g),(g,r)\}$, $v2 = \{(r,b),(b,r)\}$ and $v3 = \{(b,g),(g,b)\}$. These are the meta values for X_{new}. One solution of the transformed CSP is $(X_{new} = v1, X2 = b, X3 = y)$ and this solution represents the two detailed solutions $(X1 = r, X2 = b, X3 = y, X4 = g)$ and $(X1 = g, X2 = b, X3 = y, X4 = r)$. This small example shows that one can find interchangeability in problems after solving a subproblem, although no neighborhood interchangeability could be found in the original problem formulation. Case adaptation is rather straight forward given meta interchangeable values to move from one solution to another. Of course if our current solution is $(X1 = r, X2 = b, X3 = y, X4 = g)$ and the customer wants to replace $X1 = r$ with $X1 = g$ without changing the value for $X4$ then replacing the meta values is not possible. However by applying the clustering several times we can build meta variables, which correspond to subparts of a case, that are rather independent. Thus one can replace parts of a solution by interchangeable parts without affecting the rest of the case. If one builds again a solution graph as done in the subsection above one might link two solution that differ in more that only a single value. Therefor using meta-interchangeability seems to be more appropriate for case adaptation than context dependent interchangeability.

Details of the clustering algorithm can be found in (Weigel and Faltings 1997). We simply give an example of the clustering in the constraint graph of a CSP with 16 variables shown in Fig.5. Every variable name is followed by its domain size and every constraint is labelled with the number of tuples allowed by the constraint. A greedy clique clustering algorithm determines 5 subproblems with complete graphs indicated by the bold edges in Fig. 6.1. Every subproblem is solved individually and appears in Fig. 6.2 as a new meta variable. The variables $V13$, $V14$, $V15$ and $V16$ for example are clustered together into the meta variable, named "$V16V13V14V15$". The last number of the node label is the number of solutions of the subproblem, i.e., 30 in this case. Interchangeability identifies 8 equivalence classes of values each of size 2. Thus the domain size can be reduced to 22. As an example of a set of interchangeable values consider $S1 = (V9 = 0, V10 = 3, V11 = 3, V12 = 1)$ and $S2 = (V9 = 0, V10 = 3, V11 = 1, V12 = 2)$. Renaming $V9V10V11V12$ to $AV2$ (AV stands for Abstract Variable) and applying arc-consistency to the CSP is shown in Fig. 6.3. With arc-consistency one can remove 5 values from the domain of $AV4$. Running the clustering algorithm again we end up with three variables $AV2AV4$, $AV1AV5$ and $AV3$.

Every meta-value in a solution of the resulting CSP

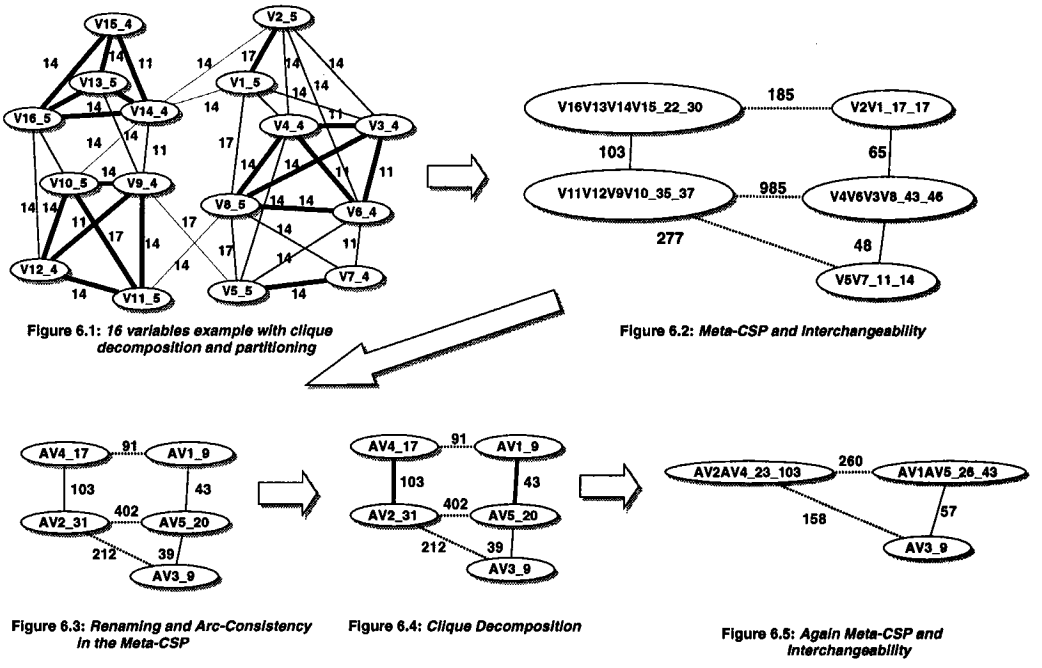

Figure 6.1: *16 variables example with clique decomposition and partitioning*

Figure 6.2: *Meta-CSP and Interchangeability*

Figure 6.3: *Renaming and Arc-Consistency in the Meta-CSP*

Figure 6.4: *Clique Decomposition*

Figure 6.5: *Again Meta-CSP and Interchangeability*

Figure 5: *Precompiling P_S to find MI values used for the adaptation*

can be replaced by the set of values it represents. Thus adaptation of a case solution is accomplished by replacing values that belongs to the same meta value. One should note that changing values within a cluster that influence values in another cluster cannot be done using interchangeability. In this case it would be necessary to apply for example context dependent interchangeability on the clustered CSP or simply start an exhaustive search to find a "repair" of the current case. This situation however might be rare if we can cluster variables into sets of relatively independent meta variables. In fact, aggregating variables into clusters allows us to detect parts of a case that are relatively independent from other parts. Consider an elevator example where a minor change in the "cabin cluster" will not influence the values in the "counterweight cluster". Thus such adaptations can then be done locally.

Conclusion

We have shown how to use various concepts based on interchangeability in CSPs for case adaptation. Precompilation of the constraint problem to compute interchangeability is the key for effective case adaptation or similarly reconfiguration of a given product. Future work should answer the question how much preprocessing needs to be done so that all possible adaptations of a given case can be found with interchangeability.

References

B. Faltings. Case Reuse by Model-Based Interpretation. In M.L. Maher and P. Pu, editors, *Issues and Applications od Case-Based Reasoning in Design*, pages 39 – 60. Lawrence Earlbaum Associates, Hillsdale, NJ., 1997.

Eugene C. Freuder. Eliminating Interchangeable Values in Constraint Satisfaction Problems. In *Proc. of AAAI-91*, pages 227–233, Anaheim, CA, 1991.

GNOSIS Workpackage 4. Soft Machinery. Report GNOSIS/TW4/All/R/D/20Oct93/C, IMS-GNOSIS, available from ftp.cpsc.ucalgary.ca in directory pub/KSI/GNOSIS/tw4/, 1993.

T.R. Hinrichs. Towards an Architecture of Open World Problem Solving . In *Proc. CBR Workshop*, pages 182 – 189, Morgan Kaufmann, San Francisco, 1988.

M.L. Maher, B. Balachandran, and D.M. Zhang. *Case-Based Reasoning in Design.* Lawrence Erlbaum Associates, 1995.

R. Weigel and B.V. Faltings. Structuring Techniques for Constraint Satisfaction Problems. In *Proc. of the 16 th IJCAI*, pages 418 – 423, Nagoya, Japan, 1997.

R. Weigel, B.V. Faltings, and B. Y. Choueiry. Context in discrete Constraint Satisfaction Problems. In *Proc. of the 12 th ECAI*, pages 205–209, Budapest, Hungary, 1996.

Appendix

1. **Integration name/category**: Constraint-Based Reasoning (CSP)/CBR
2. **Performance Task**: Product Configuration
3. **Integration Objective**:
   - possibility of creating the case-base using CSP model
   - runtime efficiency because of might be easier to adapt a closeby solution than solving the CSP from scratch
4. **Reasoning Components**: Interchangeability of values for CSP variables for case adaptation
5. **Control Architecture**: Unified
6. **CBR Cycle Step(s) Supported**:
   - CSP supports CBR for case adaptation
   - CBR supports CSP by providing a first solution
7. **Representations**: Constraint model used to build the case base
8. **Additional Components**: None
9. **Integration Status**: Proposed/Applied
10. **Priority future work**: Applications in configuration and design. Theoretical Analysis: How many cases are covered by adapting a case with interchangeability ? In which way can we relax constraints in order to adapt a case ? Which are the good heuristics for determining the "similar" cases using constraint model ?