Arc Consistency for CP-Nets under Constraints

Eisa Alanazi and **Malek Mouhoub**
Department of Computer Science
University of Regina

Abstract

Many real world applications require managing both system requirements and user preferences where the latter are usually provided in a qualitative way. We introduce a new approach to handle these two aspects, in an efficient way, respectively through Constraint Satisfaction Problems (CSPs) and CP-nets. In particular, we use Arc Consistency (AC) in order to reduce the search space needed when looking for the optimal outcome in an acyclic CP-net. More precisely, assuming that there are always some shared variables between the CP-net and the CSP, our approach works by first applying AC to the CSP and then update the CP-net with the remaining variables values. The resulting simplified CP-net will then be used to look for the best outcome. Experimental tests conducted on randomly generated problem instances clearly show the effect of AC on the size of the search space and the time needed to find the best outcome.

Introduction

One of the main issues in building decision-making systems is to represent and handle user preferences (Boutilier et al. 2004; Rossi, Venable, and Walsh 2008; Boutilier et al. 2001). Preferences can take different structures and forms (Mouhoub and Sukpan 2008; 2012). Qualitative user preferences can be represented as preference statements under the ceteris paribus assumption via Conditional Preferences networks (CP-nets) (Brafman and Dimopoulos 2003; Boutilier et al. 2004). In addition to preferences, many decisions take place under hard constraints or problem requirements (Rossi, Venable, and Walsh 2008; Boutilier et al. 2001; Mouhoub and Sukpan 2012). Therefore, reasoning about preferences within a constrained environment is an important step towards building decision support systems. For example, in an online flight reservation system, a user may prefer to have an aisle seat while the hard constraint could be, for instance, the flight departure time. In this paper, a new approach is introduced to determine the optimal outcome for the CP-net with respect to a set of hard constraints, defined through the Constraint Satisfaction Problem (CSP) paradigm.

A CSP is a well known framework for representing and solving problems under constraints. Since solving these problems is in general NP-hard, constraint propagation techniques such as Arc Consistency (AC) have been proposed to reduce the size of the search space before and during the search (Mackworth 1977; Kumar 1992; Dechter 2003). In order to handle constraints and preferences in an efficient way, we use constraint propagation through AC in order to reduce the search space used later by the CP-net for finding the best outcome. More precisely, in our approach preferences represented by an acyclic CP-net are brought forward and ultimately, the impact of adding a set of constraints to the problem is observed. The search for the best feasible outcome rather than simply the best outcome is the focus. That is, the best outcome satisfying the set of constraints. Clearly, adding constraints could result in eliminating several scenarios or outcomes from the corresponding CP-net. For example, assume $x_1y_1 \succ x_1y_2 \succ x_2y_1 \succ x_2y_2$ is the pre order for a CP-net involving the values $\{x_1, x_2\}$ and $\{y_1, y_2\}$ for variables X and Y respectively. Now assume that the constraint $C(X, Y)$ does only allow the tuples (x_i, y_j) when $i \neq j$. Obviously, this makes x_1y_1 and x_2y_2 not feasible according to $C(X, Y)$. Therefore, x_1y_1 is no longer the best outcome but x_1y_2 is. These types of inconsistencies can easily be detected and removed if the CSP with AC is used to manage hard constraints. More precisely, our approach consists of applying AC first in order to remove some of the inconsistent values. The result is a new CP-net where some domains values are removed from the network. Ideally, this can result in a huge decrease in the search space. By discarding these inconsistent assignments for the CP-net, the discovery of the optimal outcome will be obtained in a shorter period of time. In addition, AC is performed in polynomial time which means that the extra cost due to this propagation technique does not affect the overall running time as demonstrated by the experimental tests we conducted on randomly generated instances and reported in this paper. Note that if one of the variables domain becomes empty during the AC process, the CSP is inconsistent in this case and there is no need to look for an optimal solution since a feasible one does not exist.

Recent research work on managing qualitative preferences and constraints include the Constrained CP-net (Prestwich et al. 2004), where the CP-net is first converted into a set of hard constraints. The solution to the new constraint network is then the optimal solution to the CP-net. (Boutilier

et al. 2001) proposed a different approach to handle constraints with CP-nets, however, the problem of pruning variables values and CP statements before searching for best outcome was not discussed. We are not aware of any solving system using constraint propagation when managing qualitative preferences through CP-nets and hard constraints.

The rest of the paper is structured as follows. Literature review on related work is first covered in the next section. Following that, we introduce in the third section our new approach managing CSPs and CP-nets. More precisely we present our approach by applying arc consistency to the CP-net with constraints. Experimental results evaluating the performances of our techniques are reported in the fourth section. Finally, conclusion and possible future works are listed in the last section.

Background

Conditional Preferences networks (CP-nets)

A Conditional Preferences network (CP-net) (Boutilier et al. 2004; Brafman and Dimopoulos 2003) is a graphical model to represent qualitative preferences statements including conditional preferences such as: "*I prefer A to B when X holds*". A CP-net works by exploiting the notion of preferential independency based on the *ceteris paribus* (with all other things being without change) assumption. Ceteris Paribus (CP) assumption gives us a clear way to interpret the user preferences. For instance, I prefer A more than B means I prefer A more than B if there was no change in the main characterstics of the objects. A CP-net can be represented by a directed graph where nodes represent features (or variables) along with their possible values (variables domains) and arcs represent preference independencies among features. Each variable X is associated with a ceteris paribus table (denoted as $CPT(X)$) expressing the order ranking over different values of X given the set of parents $Pa(X)$. An outcome for a CP-net is an assignment for each variable from its domain. Given a CP-net, the users usually have some queries about the set of preferences represented. One of the main queries is the best outcome given the set of preferences. We say outcome o_i is better than outcome o_j if there is a sequence of worsening flips going from o_i to o_j (Prestwich et al. 2004). A Worsening flip is a change in the variable value to a less preferred value according to the variable's CPT.

Constraint Satisfaction Problems (CSPs)

A Constraint Satisfaction Problem (CSP) (Dechter 2003) is a well-known framework for constraint problems. More formally, a CSP consists of a set of variables each defined on a set of possible values (variable domain) and a set of relations restricting the values that each variable can take. A solution to a CSP is a complete assignment of values to variables such that all the constraints are satisfied.

Arc Consistency

A CSP is known to be an NP-Hard problem. In order to overcome this difficulty in practice, several constraint propagation techniques have been proposed (Dechter 2003; Kumar 1992). The goal of these techniques is to reduce the size of the search space before and during the search for the solution to the CSP. One of the well-known constraint propagation techniques is called Arc Consistency (AC) (Mackworth 1977). The aim of AC is to enforce a 2 consistency over the constraint problem. More precisely, the 2 consistency consists in making sure that for each pair of variables (X, Y) sharing a constraint, every value a from X's domain has a corresponding value in Y's domain such that the constraint between X and Y is satisfied, otherwise a is eliminated.

Arc Consistency for the CP-net under Constraints

While a CP-net is a powerful model for representing qualitative preferences (Rossi, Venable, and Walsh 2008), managing both hard constraints and preferences is required in many real world applications (Mouhoub and Sukpan 2008; 2012; Ghavamifar, Sadaoui, and Mouhoub 2011). In these situations, it is important to determine the best solution to the CP-net with respect to the set of hard constraints that we represent with a CSP. It should be noted that satisfying a set of hard constraints is often more important than satisfying the user's preferences statements due to the nature of the preferences and constraints. Constraints mostly represent strict system requirements while preferences represent a pre-order likelihood over a set of features. As a result, in our proposed approach, with respect to any CP-net there is a CSP behind it representing the hard constraints. Following this representation, when looking for the best outcome we will first run arc consistency in order to remove some inconsistencies (which will reduce the size of the search space) and then look for the best outcome in the simplified CP-net.

The result of updating the CP-net with the AC changes is a new CP-net that we call Arc Consistent CP-net (ACCP-net). In order to update the CP-net (into the ACCP-net) after arc consistency, we propose an algorithm that traverses over the shared variables (variables that belong to the CSP and the CP-net) in the CP-net and removes the domain values that have been eliminated during the AC process. As a consequence for removing a domain value x from the CP-net variable X, we remove x from the $CPT(X)$. We also remove any cp-statement that contains x in the set of children of X. For each shared variable, we use X to refer to the variable in the CP-net and $\mathbf{X}$ to refer to the corresponding variable in the CSP. The following pseudocode illustrates our proposed algorithm. Checking the consistency of a variable X is straightforward. We simply check the cardinality of the domains, i.e. $|dom(X)| \neq |dom(\mathbf{X})|$. Since for each CP-net variable X there is a set of parents $Pa(X)$, we refer to the set of different instantiations of the parents in CPT(X) as $pa(X)$. The time complexity of the this approach is as follows assuming N is the number of variables, e the number of constraints, d the largest domain size of the variables and m the largest CPT in the network. We use AC-3 (Bessière et al. 2005) for the arc consistency and the corresponding

Algorithm AC for the CP-net under Constraints

```
procedure generateACCP−net
 input
  CSP C after applying AC
  CP−net N
 output
  CP−net reflecting AC changes
begin
 for each shared variable X in N
  if ¬isConsistent(X, X)
   for each x_i ∈ dom(X)
    if x_i ∉ dom(X)
     remove x_i from dom(X) and CPT(X)
   for each Y ∈ Children(X)
    for each statment S ∈ CPT(Y)
     if x_i ∈ pa(S)
      remove S
end
```

cost is $O(ed^2)$. In the worst case scenario, the cost of the ACCP-net algorithm is $N(d+m)$.

Testing Optimality

Consideration is given to the query of finding the best assignment with respect to a set of constraints. The optimal solution for an acyclic CP-net N is the one with the minimum worsening flips according to N (Boutilier et al. 2004; Brafman and Dimopoulos 2003). The best outcome can be computed by assigning each variable to its most preferred value throughout the network . Note that in the presence of constraints, an optimal assignment A can be infeasible. In this case, a solution should be investigated where A satisfies the set of constraints C while minimizing the number of worsening flips.

Experimentation

In order to evaluate the performances of our proposed techniques, we have conducted several experiments on randomly generated instances. All the experiments are conducted using our solver integrating the algorithms managing the CSP and the CP-net. The solver is coded in Java programming language, under NetBeans 6.9.1 environment. The operating system used in the experiments is Mac OS X, version 10.6.7. The computer specifications are 2 GHz Intel Core i7 and 4 GB of RAM. Each experiment consists in running the solving technique on 100 instances and then take the average time needed to find the best outcome. The problem instances are randomly generated using the Model RB (Xu and Li 2000). The reason for choosing this model is that it has exact phase transition and the ability to generate asymptotically hard instances. The constraint tightness is defined as the ratio of the number of allowed tuples to the total number of possible combinations (cartesian product) (Beek and Dechter 1997). The tightness for a given problem is the ratio over all constraints tightness in the problem. The following two experiments are performed, given a tightness range, to evaluate the response time needed to find the best outcome. In each experiment, the tightness is altered and the average time over 100 runs needed to find the best solution is calculated. Figure 1 shows the time needed to find the best outcome when the ratio of the constraint density for each problem is 50%. On the other hand, Figure 2 represents the results when the instances are generated using the Model RB without restrictions. Thus, the tightness ratio is randomly selected for each problem. As we can easily see in both figures, arc consistency has a great effect on the response time especially with a large number of variables.

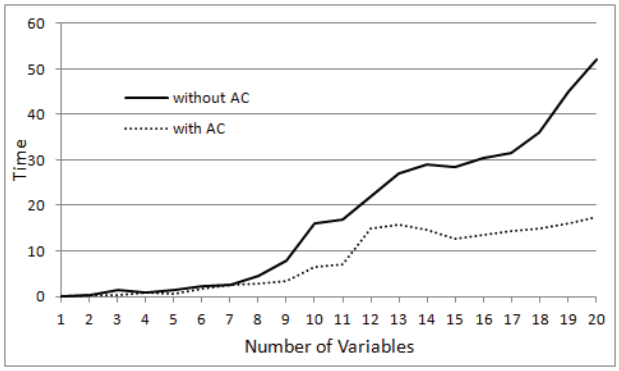

Figure 1: Determining the Best Outcome when tightness=50%

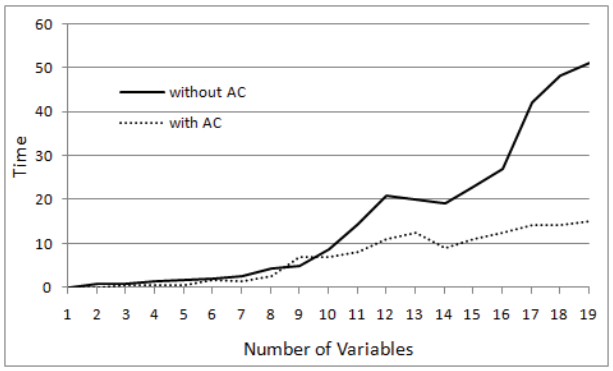

Figure 2: Determining the Best Outcome

The following two experiments are performed with different constraint tightness ratio. The goal here is to show the effect of applying arc consistency in terms of the total number of possibilities. Figure 3 shows the number of possibilities for CP-nets and CSPs with 50% tightness ratio. Eliminating some domain variables will indeed result in reducing the number of possibilities. Figure 4 shows the number of possibilities when each CSP associated with a CP-net has a tightness ratio equal to 75%. Here, the number of eliminations for domain values has been reduced. Thus, the number of possibilities in the new induced CP-nets is larger than those in Figure 3.

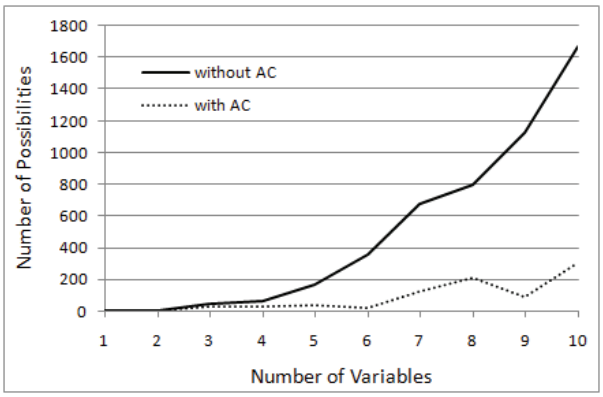

Figure 3: Number of Possibilities when Tightness=50%

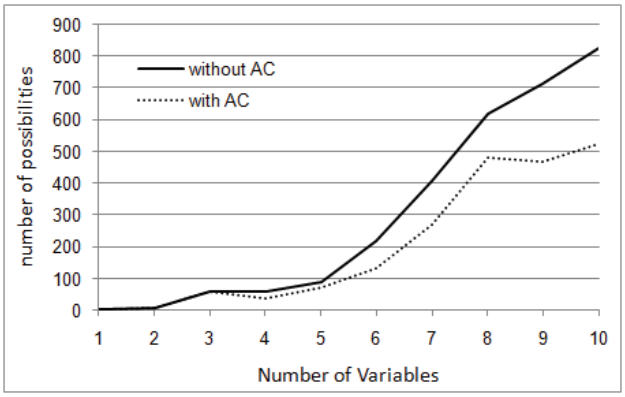

Figure 4: Number of Possibilities when Tightness=75%

Conclusion and Future Work

In this paper, a discussion was initiated regarding the relation between the CSP and the CP-net. Then, the consistency aspect of the CP-net was introduced along with a method to apply arc consistency to the CP-net. The ACCP-net algorithm with which to update the CP-net to reflect the new domains in the CSP was introduced and discussed. It was shown that applying arc consistency to the CP-net can reduce the search space and the time needed for finding the best outcome. This was also demonstrated through the experimental tests we conducted on randomly generated instances. Indeed, when using arc consistency the running time needed for finding the best outcome and the number of possible scenarios are drastically reduced. In the near future we plan to extend this framework in order to include quantitative preferences and uncertainty. One possible way to do that is to handle these two types of information using the C-semiring model. The C-semiring (or constraint-based semiring) (Bistarelli, Montanari, and Rossi 1997; Bistarelli and Rossi 2008) is a general formalism for handling different types of soft and hard constraints problems. Through its instances (Fuzzy CSPs, Probabilistic CSPs and Weighted CSPs), the C-semiring can be used to manage constraints, quantitative preferences and uncertainty. Another future work is to manage preferences in the presence of dynamic hard constraints. In this case, we can use a dynamic variant of the arc consistency algorithm (Mouhoub 2003) in the preprocessing phase.

References

Beek, P. V., and Dechter, R. 1997. Constraint tightness and looseness versus local and global consistency. *Journal of the ACM* 44.

Bessière, C.; Régin, J.-C.; Yap, R. H. C.; and Zhang, Y. 2005. An optimal coarse-grained arc consistency algorithm. *Artif. Intell.* 165(2):165–185.

Bistarelli, S., and Rossi, F. 2008. Semiring-based soft constraints. In *Concurrency, Graphs and Models*, 155–173.

Bistarelli, S.; Montanari, U.; and Rossi, F. 1997. Semiring-based constraint satisfaction and optimization. *JOURNAL OF THE ACM* 44(2):201–236.

Boutilier, C.; Brafman, R. I.; Hoos, H. H.; and Poole, D. 2001. Preference-based constrained optimization with cp-nets. *Computational Intelligence* 20:137–157.

Boutilier, C.; Brafman, R. I.; Domshlak, C.; Hoos, H. H.; and Poole, D. 2004. Cp-nets: A tool for representing and reasoning with conditional ceteris paribus preference statements. *J. Artif. Intell. Res. (JAIR)* 21:135–191.

Brafman, R. I., and Dimopoulos, Y. 2003. A new look at the semantics and optimization methods of cp-networks. In *IJCAI*, 1033–1038.

Dechter, R. 2003. *Constraint processing*. Elsevier Morgan Kaufmann.

Ghavamifar, F.; Sadaoui, S.; and Mouhoub, M. 2011. Preference elicitation and winner determination in multi-attribute auctions. In Murray, R. C., and McCarthy, P. M., eds., *FLAIRS Conference*. AAAI Press.

Kumar, V. 1992. Algorithms for constraint-satisfaction problems: a survey. *AI Mag.* 13:32–44.

Mackworth, A. K. 1977. Consistency in networks of relations. *Artificial Intelligence* 8(1):99 – 118.

Mouhoub, M., and Sukpan, A. 2008. Managing temporal constraints with preferences. *Spatial Cognition & Computation* 8(1-2):131–149.

Mouhoub, M., and Sukpan, A. 2012. Managing dynamic csps with preferences. *Applied Intelligence* DOI: 10.1007/s10489–012–0338–z.

Mouhoub, M. 2003. Arc consistency for dynamic csps. In Palade, V.; Howlett, R. J.; and Jain, L. C., eds., *KES*, volume 2773 of *Lecture Notes in Computer Science*, 393–400. Springer.

Prestwich, S.; Rossi, F.; Venable, K. B.; and Walsh, T. 2004. Constrained cpnets. In *in Proceedings of CSCLPÕ04*.

Rossi, F.; Venable, K. B.; and Walsh, T. 2008. Preferences in constraint satisfaction and optimization. *AI Magazine* 29(4):58–68.

Xu, K., and Li, W. 2000. Exact phase transitions in random constraint satisfaction problems. *Journal of Artificial Intelligence Research* 12:93–103.