Finding k-Pareto optimals for the Constrained CP-net problem

Eisa Alanazi

Department of Computer Science, Umm Al-Qura University, Makkah, Saudi Arabia

Summary
The Constrained CP-net problem concerns finding solutions that satisfy a set of hard constraints and not preferentially dominated by other satisfiable solution in the search space. In this paper, we propose an algorithm to find k optimal solutions given a constrained CP-net structure. We identify sufficient conditions under which the proposed algorithm is guaranteed to have the anytime property where the solutions found so far will never be dominated by any future solution. Furthermore, to enhance the solving process in practice, we rely on constraint propagation techniques and maintain a heuristic search function based on the minimum number of worsening flips and hamming distance to the optimal.

Key words:
Constraints, Preferences, Optimization, Decision Making

1. Introduction

Proper handling of decision maker preferences is crucial in successfully deploying AI applications [7, 9]. Preferences and user wishes represent intuitively what the user wants to see or explore in an AI application. Hence, dealing with preferences is a cornerstone in many decision making processes.

In many scenarios, there are both preferences and constraints on what the user wishes to see and what any solution must satisfy Preferences. This is evident in different application domains such as product configurations and recommender systems [5, 13]. Thus, handling both is of great interest to many applications. Preference constrained optimization [5, 11] concerns studying such problems and efficiently finding Pareto solutions (or outcomes) that are satisfied by the set of constraints and optimal according to the given preferences. The pareto optimality of solutions is defined based on two requirements: 1) satisfiability of the constraints, i.e., the solutions need to be satisfiable with the given hard constraints, and 2) preferential dominance according to the preference model (e.g., CP-net) which asserts that no satisfiable solution is dominating the current solution.

In this paper, we propose a simple yet effective approach to find k-optimals over the problem structure while applying different constraint propagation techniques. In addition, we use a heuristic to guide the search space to the set of optimal solutions. This heuristic is based on the number of worsening flips and the hamming distance to the optimal. Finding a subset of the actual pareto optimals is recommended in many scenarios. In some applications, the set of pareto optimals can be huge, therefore, rather than waiting for the whole set, the user might be satisfied witha subset of manageable size.

One important property for any algorithm for finding k optimals is the anytime behaviour. This anytime (sometimes called online) property guarantees that the set of solutions found so far are optimal regardless of the upcoming solutions. [5] proposed an anytime algorithm for the constrained CP-net problem by utilizing the conservative semantics of the CP-nets. We are not aware of any other algorithm that preserves this property in the literature of constrained CP-nets. Furthermore, we are not aware of any research work for finding k−optimals. In this paper, we define a condition with which our heuristic will always guide the space towards the non-dominated solutions and thus preserve the anytime property. Moreover, our experiments show that even when the condition is violated, there are many cases where the heuristic successfully identifies the correct k−optimals.

Our heuristic is based on the following variable ordering: We topologically rank the variables based on their participation in the constrained network (i.e., the number of constraints that a variable X is involved in) and then fix that rank to be topologically correct with the graph in CP-net .

The paper is structured as follows. The next two sections provide the background and related work. Our algorithm is then presented in section four. Section five reports the experiments over randomly generated problem instances. Finally, concluding remarks and future directions are presented in section six.

2. Preliminaries

2.1 Conditional Preference Networks (CP-nets)

The CP-net model is a graphical model, i.e., can be represented as directed graph where vertices are the attributes of the domain and edges are interdependencies among attributes [4]. It allows us to express complex (conditional) ordinal preferences of the form: “If it is Friday night, I prefer to watch a horror movie to a comedy one”. Such information or preferences is associated with every attribute. Formally, for a set of attributes A = {X1 , . . . , Xn } and assuming every Xi is associated with a set of possible domain values D(Xi), CP-nets can be described as

Manuscript received January 5, 2020
Manuscript revised January 20, 2020

a pair (G, P) where G is a directed graph and P is a set of conditional preference table T(Xi) for every attribute Xi $\in$ A. From the preference tables, we can determine how one solution is preferred to another (a.k.a dominance relation). Figure 1 shows simple network for variables X,Y and Z. The user stated that her preference to X is that she always prefer x more than $\bar{x}$. On the other hand, her preference over the variable Z depends on the values of both attributes X and Y as stated in the graph (we say X and Y are parents of Z). The figure also show the preference relation graph on all possible solutions or alternatives. The existence of a path from s to s′ is a certificate that s′ is better to s (i.e., $s' \succ s$) whereas the absence of a path represents incomparability between s and s′ ($s \triangleleft\triangleright s'$).

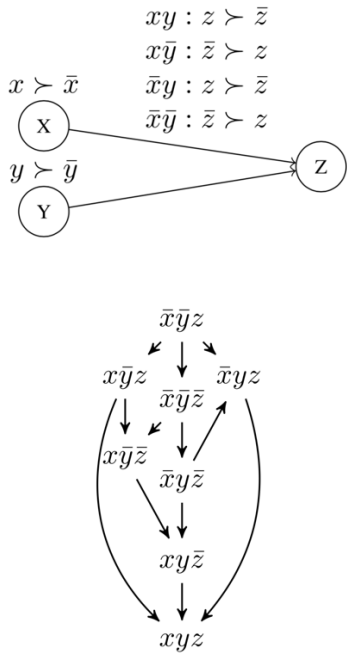

Fig. 1 A CP-net and full preference relation.

2.2 Constrained Networks

Constrained Networks (CN) [6] are undirected graph where vertices represent variables V , each has a possible domain of values D(Xi), and edges shows constraint relations between pairs of variables. A constraint relation (Xi,Xj) is a subset of the cartesian product D(Xi) × D(Xj) showing the allowed configurations. The aim is to construct an assignment (a value from each variable domain) such that all constraints are satisfied. Such assignment is known an satisfiable or consistent assignment.

A classic approach to tackle CNs is by a simple Backtracking algorithm. A more intelligent way of detecting inconsistent assignments in advance is by employing different propagation or look-aheaf techniques during the search. The main potential of using such techniques is to detect inconsistencies earlier, hence prune more branches of the search [10].

3. Related Work

Constrained CP-net problem has been tackled in many works [5, 14, 11, 8, 15, 12]. The work in [15] introduced couple of ways to prune the search space. A compilation scheme for the CP-net information into hard constraints was introduced in [12]. Then, the set of satisfiable solutions of the compiled constraints represent the pareto optimals of the constrained CP-net problem. In [8] an efficitent method to approximate the CP-net information has been proposed. The method transformed the CP-net into a soft constraint problem [3]. The work in [1] proposed a method that adopt arc consistency (AC) and edit the structure of the CP-net accordingly. The aim is to remove some domain values of the network, hence early pruning to some unpromising branches during the search.

In [5], a recursive and anytime algorithm (termed Search-CP) was introduced. Search-CP iteratively calls a smaller set of the original CP-net and strength the constraints to the current instantiated variables resulting in a simplied CN. Search-CP [5] algorithm asserts that upon the expansion of a node x, we assign the most preferred value for its current variable. This guarantees that for any node o^- after o, there is at least a variable X with a better assignment in o than o^- given their identical values of the parents. Moreover, Search-CP has a strategy to prune unpromising branches during the search process. However, the variable ordering in Search-CP is fixed in advance. The work in [2] did investigate constraint propagation for the general case of finding single solution and there exists no concrete study on variable ordering and heuristic when solving the k > 1 pareto problem.

4. Finding k Pareto Solutions

In this section, we propose a new algorithm, the k Pareto optimals, for a constrained CP-net structure. The algorithm is based on the observation that the set of (nearly) Pareto optimals are the ones that are closest to the original CP-net optimal. An outcome o is optimal with respect to CP-net N , if there is no other outcome o^- such that $N \models o^- \succ o$. In the case of acyclic CP-net, it is known that there is only one optimal outcome.

4.1 Problem Formulation

We assume an acyclic CP-net structure and adopt the hamming distance to the optimal in addition to the number of worsening flips as a measurement for how good an

assignment is. A constrained CP-net is a tuple ⟨N, con⟩ where N and con is the corresponding preferences and constraints structures respectively.
An assignment for variables X is a mapping for each variable Xi ∈ X to one of its domain values xi ∈ DXi . An assignment X is complete if it contains values to all variables in the network otherwise it is partial. The induced graph in Figure 1b represents all possible complete assignments in the network. In the following, we use both nodes and assignments interchangeably. We compute the expected cost for a node X in the search space as follow: f(X) = r(X) where r(X) is the total number of worsening flips associated with X. The number of worsening flips for a variable Y ∈ X assigned to value yi given Pa(Y), r(Y = yi|P a(Y)), is equal to the position of yi in the order associated with C P T (Y) given its parents values. For instance, r(C = c ¬|A = a, B = b) in Figure 1b is equal to one. In the following, we simply refer to the number of worsening flips for a node X as r(X) unless stating otherwise. Formally speaking, the number of worsening flips for an assignment X = [X1 = x1, X2 = x2..., Xn = xn] is computed as shown in Equation (1).

$$r(\boldsymbol{X}) = \sum_{i=1}^{n} r(X_i) \quad (1)$$

where Xi is the ith variable in X and n is the size of the assignment. We break the ties according to the hamming distance measure. The hamming distance for a node X is the number of distinct values between X and opt where opt is the unique optimal outcome for the underlying acyclic CP-net structure. Recall the network in Figure 1, it is obvious that xyz is the optimal choice and its hamming distance to xy¯z (denoted as hd(xy'z)) is equal to one where it has two worsening flips (i.e. r(xy'z) = 2). Generally, the hamming distance for an assignment X = [X1 = x1, X2 = x2,..., Xn = xn] is calculated in the following way.

$$hd(\boldsymbol{X}) = \sum_{i=1}^{n} hd(X_i) + |n - k| \quad (2)$$

where k is the cardinality of the optimal. Finally, the heuristic function is defined as:

$$f(\boldsymbol{X}) = \min(r(\boldsymbol{X})) \quad (3)$$

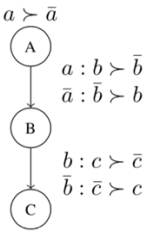

Fig. 2 f (x) fails to preserve the anytime behavior

4.2 k Pareto Algorithm

As discussed in [5], we can maintain the search space in a way that whenever a variable X is considered, we assign X to xi where xi is the preferred value of X given its ancestors. If X = xi is infeasible according to some constraints, we try X =x_{i+1} till we have a consistent value for X, otherwise we backtrack to the previous variable and change its value. This will result in an anytime behaviour from the semantics of the CP-nets. In this paper, our motivation is to find a heuristic function that can preserve the anytime behaviour to some extend while guiding the search efficiently to find k pareto optimals. We do this by posing a restriction over the dominance relation. In particular, we assume that for any two outcomes oi and oj with the same number of worsening flips (i.e. r(oi)==r(oj)), if oi ≻oj then hd(oi)<hd(oj).
Proving the anytime behaviour under this assumption is straightforward. We adapt the notations and the proof of Lemma 3 in [5]. Thus, it is sufficient to show that for any o¯ that comes after o during the search process it is the case where o¯ does not dominate o. Indeed, if o¯ comes after o then r(o¯) ≥ r(o). If r(o¯) > r(o) we are sure from the semantics of the CP-net that N entails o¯ ≻ o. Now assume r(o¯) == r(o), If o¯ ≻ o then it must be the case where hd(o¯) < hd(o) otherwise there is no way for o¯ to dominate o. However, in our algorithm we are sure if that is the case then o¯ must proceed o hence N ⊭ o¯ ≻ o.
The restriction over the hamming distance is infeasible for general CP-nets in a sense that it can be the case where oi ≻ oj and r(oi) == r(oj) but hd(oi) ≥ hd(oj). For instance, consider the CP-net in Figure 2, in this network a'b' c ≻ a'bc where both flips and hd(a'b'c) > hd(a'bc) and the induced graph over the network is total order. However, we believe that there are other classes of CP-nets where the condition is satisfied by the nature of the induced graph. The example in Figure 1 represents one structure where the anytime property is preserved while guiding the search space through the heuristic function f . Figure 3 shows the search space for a systematic anytime algorithm. Search-CP behaves in a similar fashion except it prunes some dominated branches. Figure 4 shows the set of solutions for the CP-net as points in two dimensions space. Clearly, for any two solutions xi and xj where i ≥ j while breaking the ties according to hd(x), it is the case that either xj ≻ xi or xj <> xi (i.e., they are incomparable).
Moreover, even for the general case, we argue that the resulted outcomes are nearly optimals where the decision maker is more llikely to accept them. Our algorithm is similar to the proposed approach in [5]. However, in our algorithm we go one step ahead and guide the space via the heuristic. This will result in sooner examination of different solutions in the search space that might be optimal. For instance, assume the CP-net in Figure 1 and the constraint C(X,Y) where the tuple (x,y) is not allowed. The two pareto

optimals are xy'z' and x'yz. Assume X, Y, Z to be the topological ordering, a typical algorithm will first try the whole branch of X = x since x is unconditionally preferred to x'. However, our algorithm will explore the branch X = x' as soon as some worsening flips happen in the branch X = x. This allows the algorithm to quickly find another optimal, if exists.

A frontier is maintained which holds nodes that are soon will be expanded. The frontier is sorted based on the minimum value of f (X) and break ties by the hamming distance of X. Once the assignment is expanded to a new variable, we assign the best value of the new variable in the light of his parents' values. Initially, we initialize an empty root node where f (X) is a large integer and rank them based on most constrained function. To simplify the problem even futher, we use the procedure in [1] to shrink the search space in advance using constraint propagation techniques. The procedure to find k optimals is outlined in Algorithm 1.

4.3 Finding k-nearly Optimals

In this section we argue that even though our heuristic does not guarantee finding k−pareto optimals for arbitrary acyclic CP-nets, it generates good quality of outcomes with respect to the preference structure. This stems from the fact that the set of solutions with the minimum number of worsening flips and closest to the original optimal have satisfied many preference statements and might be a potential pareto outcome.

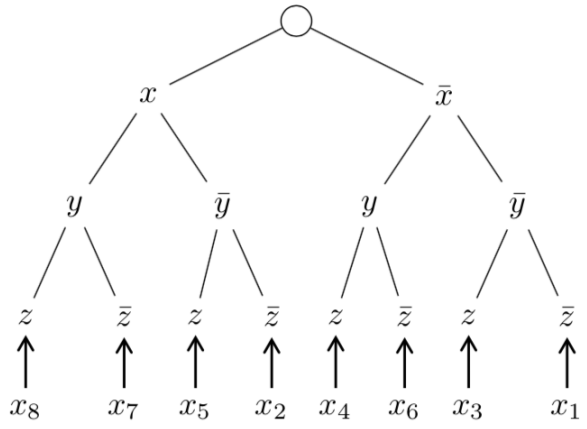

Fig. 3 All possible solutions for Example 1

input : k : number of Pareto outcomes
N : CP-net
CN : constrained network
π : CP-net order
output:
S : set of Pareto outcomes

```
1  frontier ← {root}
2  S ← {}
3  flag ← true
4  while frontier ≠ ∅ & flag do
5      n ← the top element in frontier
6      if isSatisfiable(n, CN) then
7          if isSolution(n) then
8              if ¬isPreferred(n, S) then
9                  add n to S
10                 if k == |S| then
11                     flag ← false
12                 end
13             end
14         end
15         else
16             let X ← nextVar(π)
17             let x_1 ≻ x_2 ≻ ... ≻ x_d be the order
18             for i ← 1 to d do
19                 CN ← FC(X = x_i, CN)
20                 n ← n∪ X =x_i
21                 add n to frontier
22             end
23         end
24     end
25 end
26 Return S
```

Algorithm 1: Finding k Pareto Solutions

Finding nearly optimals may have a large impact over the considered problem. For instance, consider a recommender system where the user explicitly states her preferences and constraints. The user might be interested in quick but rather nearly optimal answers instead of waiting for the exact set of optimals. Moreover, it might be the case where nearly optimal answers contain some interesting recommendations for her. In this case, nearly optimal solutions is a good candidate to apply in practice. Further extensions can be applied here. For example, maintaining q Pareto outcomes where q < k and the rest are nearly optimals that are not dominated by the q outcomes.

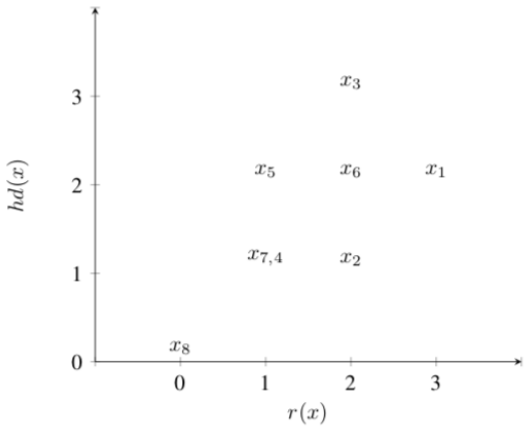

Fig. 4 f (x) when preserving the anytime property

5. Experimentation

In this section we evaluate three methods for finding k optimals. The first method is similar to the method proposed in [5] except that it is not a recursive and with no variable ordering heuristic and constraint propagation (called Depth First Search (DFS) henceforth). The second method is our proposed k algorithm (that we call f(x)). The third method is our proposed algorithm in addition to the most constrained heuristic and Forward Checking (FC) constraint propagation. All experiments were conducted over the ACCPnet structure [1], which is the resulting CP-net after removing inconsistent domain values following AC propagation technique.

The computer specifications for the experiments are as follow: the computer is Mac Book Pro with processor Intel Core i5 with four gigabyte RAM. For all experiments,n, which is the number of variables, is randomly selected between 3 and 20. After fixing n, we generate a random structure of CP-net where every attribute has at most two parents.

The constrained network structure is randomly generated where the number of variables ncap is equal to n/1.5 and constraints density and tightness both set to 50%.

We conduct three experimental tests. First, we report the time needed (in milliseconds) to find the k optimal for all the three methods. Second, we report the number of generated nodes for each method. In the third experiment, we report a comparison between the DFS actual k−pareto outcomes and the proposed method. More precisely, we run DFS first, if the result is the original optimal, we ignore it and regenerate random structures again. Otherwise, we run the other two algorithms over the same structure. For any n, we compute time needed to find the optimals for five times and then take the average. We abort any iteration that takes more than 30 seconds. We also take the average number of the visited nodes. Lastly, we compare the f(x) and DFS results. For any n, we compare the DFS generated solutions with the f(x) results. We calculate the change percentage as follows. Each solution s has a weight equal to w(s) = 1/k. For each solution, we compute the number of different values diff and take their percentage (i.e. diff/n). We sum these numbers and multiply them by w(s). For instance, assume k = 3 and f(x) misses two solutions in a particular iteration; one solution with 4 different values and another with 6 out of n = 20. The percentage is ((4/20) · 100)/3 = 6.6%. Similarly, the second solution has 10%, Therefore, this iteration has 16.6% change.

Figures 5 and 6 show the time and number of nodes generated respectively where the x-axis is the number of variables. DFS goes blindly in the search space looking for an optimal. It examines the whole branch before considering the other branches. Generally, this takes more time than the heuristic method. On the other hand, f(x) finds the optimals sooner in many instances. This is specially true if the optimals are distributed along the search space. While the performance of FC is not superior, it removes many inconsistent nodes from the space and it is unclear yet whether it is favourable to apply it in the long run with the most constrained variable heuristic.

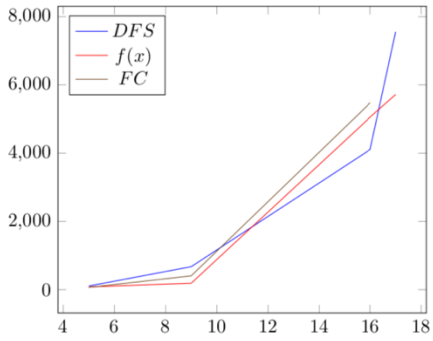

Fig. 5 Time needed for different methods when k = 3

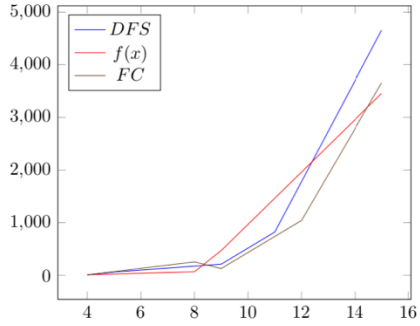

Fig. 6 Number of Generated Nodes

Table 1: Percentage of missing some actual k−optimal values

Variables	18	19	20	26
Percentage	33.5%	10%	8.25%	6.6%

5. Conclusion

This paper discusses the problem of finding k-Pareto optimals for the constrained CP-net problem. The proposed algorithm uses the number of worsening flips and the hamming distance to the optimal as a heuristic. We adopted the most constrained variable heuristic to rearrange variables in the CP-net and used Forward Checking (FC) during the search to remove some inconsistent values from the search space. We identified a condition under which the algorithm has an anytime property where the set of pareto optimals found so far will never shrinks. We briefly discussed the applicability of finding nearly optimal solutions.

In the near future, we are planning to find an enhancement over the proposed heuristic. Another important line of work is to have theoretical justification for choosing the number k. Different k values have a clear and direct impact on the performance of the algorithm. So far, we have considered k to be user-centric and ignored the underlying problem structure. That is, the user inputs it and then the algorithm looks for k Pareto solutions. An enhancement over this is to relate k to the underlying structure and find an optimal value for it. While the notion of optimality in this context is debatable, we think a value that will most likely satisfy the user. This relation might be established based on the maximal anti-chain resulted from the induced graph.

References

[1] E. Alanazi and M. Mouhoub. Arc consistency for cp-nets under constraints. In FLAIRS Conference, 2012.
[2] E. Alanazi and M. Mouhoub. Variable ordering and constraint propagation for constrained cp-nets. Applied Intelligence, 44(2):437–448, 2016.
[3] S. Bistarelli, U. Montanari, and F. Rossi. Semiring-based con- straint satisfaction and optimization. JOURNAL OF THE ACM, 44(2):201–236, 1997.
[4] C. Boutilier, R. I. Brafman, C. Domshlak, H. H. Hoos, and D. Poole. Cp-nets: A tool for representing and reasoning with conditional ceteris paribus preference statements. J. Artif. Intell. Res. (JAIR), 21:135–191, 2004.
[5] C. Boutilier, R. I. Brafman,H..Hoos,and D. Poole. Preference- based constrained optimization with cp-nets. Computational Intelligence, 20:137–157, 2001.
[6] R. Dechter. Constraint processing. Elsevier Morgan Kaufmann, 2003.
[7] C. Domshlak, E.Hullermeier,S. Kaci,andH. Prade. Preferences in ai: An overview. Artif. Intell., 175(7-8):1037–1052, 2011.
[8] C. Domshlak, F. Rossi, K. B. Venable, and T. Walsh. Reasoning about soft constraints and conditional preferences: complexity results and approximation techniques. CoRR, abs/0905.3766, 2009.
[9] J. Goldsmith and U. Junker. Preference handling for artificial intelligence. AI Magazine, 29(4):9–12, 2008.
[10] A. K. Mackworth. Consistency in networks of relations. Artificial Intelligence, 8(1):99 – 118, 1977.
[11] S. Prestwich, F. Rossi, K. B. Venable, and T. Walsh. Constrained cpnets. In in Proceedings of CSCLP04, 2004.
[12] S. D. Prestwich, F. Rossi, K. B. Venable, and T. Walsh. Constraint-based preferential optimization. In AAAI, pages 461–466, 2005.
[13] F. Rossi, K. B. Venable, and T. Walsh. Preferences in constraint satisfaction and optimization. AI Magazine, 29(4):58–68, 2008.
[14] N. Wilson. Consistency and constrained optimisation for conditional preferences. In ECAI, pages 888–894, 2004.
[15] N. Wilson and W. Trabelsi. P. rules for constrained optimisation for conditional preferences. In CP, pages 804–818, 2011.

Eisa Alanazi received his B.Sc. degree in Information Systems from King Saud University, in 2007, and the MSc and PhD degree from the University of Regina, Canada in 2011 and 2017 respectively. He is currently an Assistant Professor at the Department of Computer Science, College of Computers and Information Systems in Umm Al-Qura University in Saudi Arabia. His research interests include preference learning and reasoning.