

The Polytope of Tree-Structured Binary Constraint Satisfaction Problems^{*}

Meinolf Sellmann

Brown University, Department of Computer Science
115 Waterman Street, P.O. Box 1910, Providence, RI 02912
sello@cs.brown.edu

Abstract. We correct a result that we recently published in this conference series on the polytope of Binary Constraint Problems (BCPs). We had claimed that the so-called "support formulation" would characterize the convex hull of all feasible solutions to tree-structured BCPs. We show that this claim is not accurate by providing a small counter example. We then show that the respective polytope defines a facet of the stable-set polytope of a perfect graph which allows us to perform LP inference in polynomial time.

1 Binary Constraint Satisfaction

Definition 1 (Binary Constraint Satisfaction Problem).

- A binary constraint problem (BCP) is a triplet $\langle V, D, C \rangle$, where $V = \{X_1, \dots, X_n\}$ denotes the finite set of variables, $D = \{D_1, \dots, D_n\}$ denotes a set of n finite sets of possible values for these variables (D_i is called the domain of variables X_i), and $C = \{C_1, \dots, C_m\}$ is the set of constraints, where $C_j : D_{j_1} \times D_{j_2} \rightarrow \text{Bool}$ specifies which simultaneous assignments of values to the variables X_{j_1} and X_{j_2} are allowed. The set $\{X_{j_1}, X_{j_2}\}$ is called the scope of constraint C_j .
- An assignment for a BCP $\mathcal{P} = \langle V, D, C \rangle$ is a function $\sigma : V \rightarrow \bigcup_{i \leq n} D_i$. A solution to a BCP $\mathcal{P} = \langle V, D, C \rangle$ is an assignment σ such that $\sigma(X_i) \in D_i$ for all $1 \leq i \leq n$ and such that $C_j(\sigma(X_{j_1}), \sigma(X_{j_2})) = \text{true}$ for all $1 \leq j \leq m$. The set of all solutions to a BCP \mathcal{P} is denoted by $\text{Sol}(\mathcal{P})$.

Note how, in contrast to the custom in integer programming, in CP the term "binary" is used to express that all *constraints* affect just two variables, while the size of the *domain* of each variable is not limited! The fact that the arity of the constraints is limited to two allows us to state constraints simply as sets of allowed pairs $\mathcal{R}_{j_1, j_2} = \{(k, l) \mid X_{j_1} = k, X_{j_2} = l \text{ ok}\}$, or, alternatively, as sets of forbidden pairs $\overline{\mathcal{R}}_{j_1, j_2} = \{(k, l) \mid X_{j_1} = k, X_{j_2} = l \text{ forbidden}\}$.

It is easy to see that the general BCP is NP-hard. One simple way is to reduce from graph coloring where each node is modeled as a variable that must be assigned a color such that adjacent nodes are not colored identically (i.e., the corresponding constraint on each edge $\{i, j\}$ is a not-equal constraint $\overline{\mathcal{R}}_{i, j} = \{(k, k) \mid \forall k\}$). Conversely, every binary constraint problem can be visualized

^{*} This work was supported by the National Science Foundation through the Career: Cornflower Project (award number 0644113).

as a *constraint network* where each node corresponds to a variable and an edge connects two nodes iff there exists a constraint over the corresponding variables. Of course, the exact semantic of the constraints is lost in that visualization. However, it is a well-known fact that any BCP whose corresponding constraint network is a tree can be solved in polynomial time [4, 5].

2 The Support Formulation

In [1], we devise an IP model for BCPs by using linear constraints to specify that, when a variable X_{j_1} takes value k , variable X_{j_2} must take a value that is consistent with $X_{j_1} = k$. In that way, we enforce that each variable assignment is *supported* by a correct assignment to adjacent variables (by which we mean variables that share a constraint). The IP then reads:¹

$$\begin{aligned}
S_{IP} = \max \quad & \sum p_{ik} y_{ik} \\
\text{s.t.} \quad & y_{j_1 k} - \sum_{l:(k,l) \in \mathcal{R}_{j_1, j_2}} y_{j_2 l} \leq 0 \quad \forall 1 \leq j \leq m, k \in D_{j_1} \quad (1) \\
& \sum_{k \in D_i} y_{ik} = 1 \quad \forall i \in \{1, \dots, n\} \quad (2) \\
& y_{ik} \in \{0, 1\} \quad \forall i \in \{1, \dots, n\}, k \in D_i \quad (3)
\end{aligned}$$

The above formulation dominates the traditional way of expressing constraints (1) by constraints $y_{j_1 k} + y_{j_2 l} \leq 1$ for all conflicting assignments $(k, l) \in \overline{\mathcal{R}}_{j_1, j_2}$: We show that $y \in S_{LP}$ (where S_{LP} denotes the linear continuous relaxation of the above S_{IP}) implies $y_{j_1 k} + y_{j_2 l} \leq 1$ for all $1 \leq j \leq m, (k, l) \in \overline{\mathcal{R}}_{j_1, j_2}$. Given $y \in S_{LP}$, it holds that $y \geq 0$ and $\sum_{k \in D_i} y_{ik} = 1$. Moreover, for all $1 \leq j \leq m, k \in D_{j_1}, 0 \geq y_{j_1 k} - \sum_{l:(k,l) \in \mathcal{R}_{j_1, j_2}} y_{j_2 l} = y_{j_1 k} - (1 - \sum_{l:(k,l) \in \overline{\mathcal{R}}_{j_1, j_2}} y_{j_2 l})$. Consequently, $y_{j_1 k} + \sum_{l:(k,l) \in \overline{\mathcal{R}}_{j_1, j_2}} y_{j_2 l} \leq 1$, which implies $y_{j_1 k} + y_{j_2 l} \leq 1$ for all $1 \leq j \leq m, (k, l) \in \overline{\mathcal{R}}_{j_1, j_2}$.

On the other hand, assume we are given a BCP (V, D, C) with $V = \{X_1, \dots, X_5\}$, $D = \{\{1, 2, 3, 4\}, \{1, 2\}, \{1, 3\}, \{2, 3\}, \{0, 4\}\}$, and $C = \{(X_1 = 4 \vee X_1 = X_2), (X_1 = 4 \vee X_1 = X_3), (X_1 = 4 \vee X_1 = X_4), (X_1 \neq X_5)\}$, and we are to maximize X_5 . S_{LP} returns an optimal continuous solution with value 0, which happens to be the optimal value that any integer solution can achieve. Now consider $y_{11} = y_{12} = y_{13} = 1/3, y_{14} = 0, y_{21} = y_{22} = 1/2, y_{31} = y_{33} = 1/2, y_{42} = y_{43} = 1/2, \text{ and } y_{50} = 0, y_{54} = 1$. It is easy to verify that this solution, which achieves an objective value of 4, is feasible when constraints (1) are replaced by the traditional constraints. Consequently, the support formulation is never worse but in general stronger than the traditional way of linearizing binary constraint networks.

¹ Whereby, for simplicity in constraints (1), we assume that each constraint over variables i, j induces two truth tables $\mathcal{R}_{i,j}$ and $\mathcal{R}_{j,i}$.

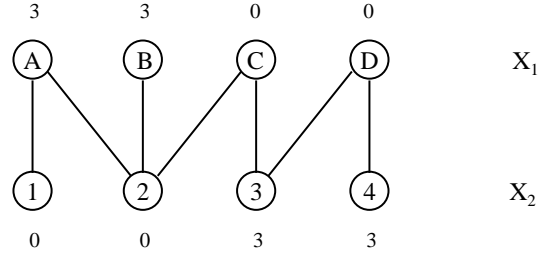


Fig. 1. A BCP with two variables X_1 and X_2 . The figure shows the domains of the respective variables and an edge between values that the variables are allowed to take simultaneously. The numbers above each value show the profit that is achieved when a variable is set to this value.

Now, in [6] we claimed that, when the given BCP was tree-structured, the linear continuous relaxation S_{LP} of S_{IP} provided a perfect characterization of the convex hull of all integer feasible solutions. When trying to use the same proof-technique for the linearization of a different type of constraints, we realized the following: In the proof of Theorem 1 in [6] we consider a Lagrangian relaxation of S_{IP} and show that its value is the same as that of the linear relaxation S_{LP} , while the corresponding solution is integer and obeys all relaxed constraints. However, this does *not* imply that the corresponding solution is optimal for the original problem. Ergo, the proof is not complete. The following example shows that the proof can also not be corrected:

Example 1. Consider the BCP $\langle V, D, C \rangle$, where $V = \{X_1, X_2\}$, $D = \{D_1, D_2\}$ with $D_1 = \{A, B, C, D\}$ and $D_2 = \{1, 2, 3, 4\}$, and $C = \{C_1\}$ with $C_1 : D_1 \times D_2 \rightarrow Bool$ is given by the set of allowed pairs $\mathcal{R}_{1,2} = \{(A, 1), (A, 2), (B, 2), (C, 2), (C, 3), (D, 3), (D, 4)\}$ (see Figure 1). Assume we achieve a profit of 3 when setting X_1 to A or B and 0 otherwise, and another profit of 3 when setting X_2 to 3 or 4 and 0 otherwise. Clearly, the maximum profit we can achieve is 3. However, when setting $y_{1A} = y_{1B} = y_{1D} = 1/3$, $y_{1C} = y_{21} = 0$, and $y_{22} = y_{23} = y_{24} = 1/3$, then y is feasible for S_{LP} and achieves a profit of 4 which is strictly greater than the optimal value of 3.

3 The Polytope of Tree-Structured BCPs

Theorem 1. *If the BCP that is given has a tree-structured constraint network, then the convex hull of all solutions to S_{IP} defines a facet of the stable-set polytope of a perfect graph.*

Proof. Consider the “conflict graph” that emerges from the given BCP: we have one node that corresponds to each variable assignment (similar to the nodes in Figure 1), and an edge between two nodes if and only if both corresponding assignments are incompatible (that is, one edge for each pair in $\overline{\mathcal{R}_{j_1, j_2}}$ and one for each pair of nodes that belong to the same variable domain). We claim that

this graph is perfect. We prove this claim by showing that it is Berge [2], i.e., that it and its complement graph do not contain an odd cycle of length 5 or more with no shortcuts (a so-called “odd-hole”).

Consider the conflict graph and assume that it contains a hole, i.e., a cycle with no shortcuts of length 5 or more. Wlog we may assume that this cycle is minimal and involves nodes that belong to at least three BCP variables (otherwise there exists a shortcut between nodes that belong to the same variable). Because of the tree-structure of the given BCP, this implies that there exist two non-adjacent nodes in the cycle that belong to the same variable, which implies that there exists a shortcut in the cycle. Therefore, the cycle was not minimal.

Now consider the complement of the conflict graph. It contains edges for all pairs in $\overline{\mathcal{R}_{j_1, j_2}}$ and edges between all nodes that belong to BCP variables that have no constraint linking them. The given BCP is acyclic, thus any hole cannot involve nodes that belong to more than two BCP variables (otherwise there exists a shortcut). However, any subgraph on nodes from two variables is bipartite, which means that all cycles have even length. Consequently, there also exists no odd-hole in the complement of the conflict graph. \square

According to [3], our result implies that we can characterize the polytope of tree-structured BCPs as a linear program where we the constraints enforce that the weight of each maximal clique in the conflict graph is lower or equal one. The support formulation comes close to enforcing these clique constraints, but with two shortcomings: First, in case that the support in D_{j_2} of a value $k \in D_{j_1}$ is a superset of the support in D_{j_2} of another value $h \in D_{j_1}$, the support formulation considers a clique that is not maximal (e.g., consider values A and B in Figure 1). Second, there may be other maximal clique-constraints that are not enforced.

The first shortcoming is easily addressed: Denote with $H_j(k) := \{l \in D_{j_2} \mid (k, l) \in \mathcal{R}_{j_1, j_2}\}$ the support of value $k \in D_{j_1}$ in D_{j_2} . Problematic are those $h \in D_{j_1}$ for which $H_j(h) \subseteq H_j(k)$. Then, $y_{j_1 h} \leq \sum_{l \in H_j(h)} y_{j_2 l} \leq \sum_{l \in H_j(k)} y_{j_2 l}$. And since X_{j_1} can only take either value k or h , with $y_{j_1 k} \leq \sum_{l \in H_j(k)} y_{j_2 l}$, we can enforce $y_{j_1 k} + y_{j_1 h} \leq \sum_{l \in H_j(k)} y_{j_2 l}$. We achieve a strengthened formulation:

$$SS_{IP} = \max \sum p_{ik} y_{ik}$$

$$s.t. \quad y_{j_1 k} + \sum_{h: H_j(h) \subseteq H_j(k)} y_{j_1 h} - \sum_{l \in H_j(k)} y_{j_2 l} \leq 0 \quad \forall 1 \leq j \leq m, k \in D_{j_1} \quad (4)$$

$$\sum_{k \in D_i} y_{ik} = 1 \quad \forall i \in \{1, \dots, n\} \quad (5)$$

$$y_{ik} \in \{0, 1\} \quad \forall i \in \{1, \dots, n\}, k \in D_i \quad (6)$$

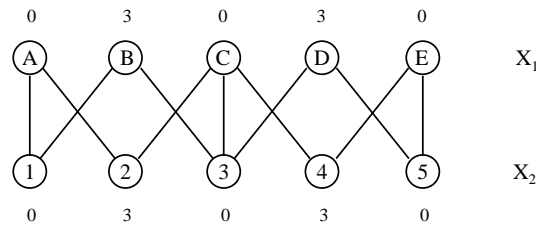


Fig. 2. A BCP with two variables X_1 and X_2 . The figure shows the domains of the respective variables and an edge between values that the variables are allowed to take simultaneously. The numbers above each value show the profit that is achieved when a variable is set to this value.

To address the second shortcoming, since there may be many other maximal cliques in the perfect conflict graph, it is not feasible to add them all to our formulation. We show another example in Figure 2. Note that there is a clique on nodes $y_{1B}, y_{1D}, y_{22}, y_{24}$ in the conflict graph whose corresponding constraint is not enforced in SS_{IP} : The solution $y_{1B} = y_{1C} = y_{1D} = y_{22} = y_{23} = y_{24} = 1/3, y_{1A} = y_{1E} = y_{21} = y_{25} = 0$ assigns a weight of $4/3$ to the nodes in this clique but is feasible for SS_{IP} . The solution achieves a profit of 4 while the optimal integer solution has value 3.

In theory, for each problem it is possible to generate the relevant missing clique constraints in a lazy fashion [3]. To obtain polynomial guarantees on the number of additional constraints that need to be generated, we would have to use the Ellipsoid algorithm to solve our LPs, though. In practice, we may prefer to use a practically efficient LP solver instead and terminate the generation of cuts early, knowing that the support formulation without any additional cuts is already strictly stronger than the traditional formulation.

References

1. I.D. Aron, D.H. Leventhal, M. Sellmann. A Totally Unimodular Description of the Consistent Value Polytope for Binary Constraint Programming. *CPAIOR*, LNCS 3990:16–28, 2006.
2. M. Chudnovsky, N. Robertson, P.D. Seymour, R. Thomas. Progress on Perfect Graphs. *Mathematical Programming*, 97:405–422, 2003.
3. V. Chvatal. On certain polytopes associated with graphs. *Combinatorial Theory B*, 18:138–154, 1975.
4. R. Dechter and J. Pearl. Tree clustering for constraint networks. *Artificial Intelligence*, 38:353–366, 1989.
5. E.C. Freuder. Complexity of k-tree structured constraint satisfaction problems. *AAAI*, pp. 4–9, 1990.
6. M. Sellmann, L. Mercier, D.H. Leventhal. The Linear Programming Polytope of Binary Constraint Problems with Bounded Tree-Width. *CPAIOR*, 275–287, 2007.