Tractable Tree Convex Constraint Networks *

Yuanlin Zhang and **Eugene C. Freuder**
Cork Constraint Computation Center
University College Cork
Cork, Republic of Ireland
{y.zhang, e.freuder}@4c.ucc.ie

Abstract

A binary constraint network is tree convex if we can construct a tree for the domain of the variables so that for any constraint, no matter what value one variable takes, all the values allowed for the other variable form a subtree of the constructed tree. It is known that a tree convex network is globally consistent if it is path consistent. However, if a tree convex network is not path consistent, enforcing path consistency on it *may not* make it globally consistent. In this paper, we identify a subclass of tree convex networks which are *locally chain convex and union closed*. This class of problems can be made globally consistent by path consistency and thus is tractable. More interestingly, we also find that some scene labeling problems can be modeled by tree convex constraints in a natural and meaningful way.

Introduction

A constraint network describes a problem as a set of variables with a finite set (called domain here) of values for each variable, and a set of constraints among the variables. In this paper, we consider only binary constraint networks where each constraint involves at most two variables. A basic task is to solve a constraint network, that is to find an assignment of a value to each variable such that all the constraints in the network are satisfied. Constraint networks, together with the effective techniques developed for them, have many applications in manufacturing, transportation, telecommunication, logistics and bio-informatics.

Since to solve general constraint networks is NP-hard, much effort has been made to identify *tractable* constraint networks which can be solved efficiently. A typical example is that when the graph of a constraint network is a tree, arc consistency is sufficient to make the network globally consistent (Freuder 1982). In this case, the topological structure among variables plays an important role. The structure or semantics of the constraints is also used to identify tractable networks, using the concept of k-consistency (Freuder 1978). Dechter (1992) shows that some local consistency in a network whose domains are of limited size ensures global consistency. For *monotone constraints*, path consistency implies global consistency (Montanari 1974). Van Beek and Dechter (1995) generalize monotone constraints to a larger class of *row convex constraints* which are further generalized to *tree convex constraints* by Zhang and Yap (2003).

A binary constraint network is tree convex if we can construct a tree for the domain of the variables so that for any constraint, no matter what value one variable takes, all the values allowed for the other variable form a subtree of the constructed tree. To construct a tree for a domain is to construct a tree such that its vertex set is exactly the domain itself. See Fig. 1(a) for an example of a tree convex constraint. It has been shown that a tree convex network is globally consistent if it is path consistent. However, if a tree convex network is not path consistent, enforcing path consistency on it *may not* make it globally consistent because some constraints may be modified during the enforcing procedure and thus may no longer be tree convex.

In this paper, we examine the tree convex constraints and characterize conditions under which the desirable tree convex property of a network is preserved when path consistency is enforced. We then identify a tractable class of restricted tree convex constraints and show an application. This result generalizes earlier work on monotone and connected row convex constraints (Deville, Barette, & Van Hentenryck 1997). Our work differs from the work by Jeavons et al. (1997) in that we study the tractability of constraint networks whereas they study that of constraint languages. The tractable class reported here may not be closed under the known algebraic operations proposed by these and other researchers.

We first generalize tree convex constraint networks by relaxing the existence of a tree on *all* values to the existence of a tree on *the domain of each variable*. The intuition is that a constraint is defined on the domains of the involved variables, rather than on all possible values in the network. Such a relaxed tree convex network is globally consistent if it is path consistent.

Next, we need to add some restrictions on a relaxed tree convex network to avoid the destruction of its tree convexity during the process enforcing path consistency. Enforcing path consistency involves the intersection and composition of constraints. It can be verified that the intersection of

*This work has received support from Science Foundation Ireland under Grant 00/PI.1/C075.

two constraints is also tree convex, but the composition may not be tree convex any more. We propose a natural restriction on tree convexity which is preserved under composition. Given a tree convex constraint c_{xy} on variables x and y with domains D_x and D_y respectively, the restriction is that for some tree on D_x, any two values that are neighbors on the tree should share a common support in D_y. The properties are then studied.

Finally, a tractable class of networks is presented.

A careful reader may be concerned about the usefulness of tree convexity even before our discussion of further restrictions on it. There do exist many tractable problems belonging to this class (e.g., monotone and row convex constraints), but the tree structure for a domain seems still too strange to be of any practical use. No new application is given in Zhang & Yap (2003). Interestingly, we find that tree convex constraints help to model some scene labeling problems in a natural and meaningful way, which may not be captured by a general constraint network. In the following example we show that for some problems, there exists a relationship between the constraints and a certain structure on the domain of variables. Consider the inequality on integers: $x < y$ with $x \in \{1, 2, 3\}$ and $y \in \{2, 3, 4\}$. There is a natural total ordering (denoted by $\preceq$ to be distinguished from $<$) on the values. Its graphical representation is a chain of the values {1,2,3,4}. If $x.1 < y.2$ where $x.1$ denotes integer 1 for variable x, then $x.1 < y.3$ because $y.2 \preceq y.3$. In fact, the constraint $x < y$ has to conform to the structure $\preceq$ on integers. A more complex example, where the values of each domain form a tree structure, will be presented later.

Related work is discussed and concluding remarks are given in the conclusion.

Preliminaries

In this section, we introduce the basic concepts and notations used in this paper.

Constraint Network A binary *constraint network* consists of a set of variables $V = \{x_1, x_2, \cdots, x_n\}$ with a finite domain D_i for each variable $x_i \in V$, and a set of binary constraints C. c_{xy} denotes a constraint on variables x and y which is defined as a relation over D_x and D_y. Operations on relations, e.g., *intersection ($\cap$), composition ($\circ$), and inverse*, are applicable to constraints.

We assume that, between any ordered variables (x, y), there is only one constraint. *c_{xy} and c_{yx} are considered to be two different constraints*, however, we assume the inverse of c_{xy} is equal to c_{yx}.

Image Given a constraint c_{xy} and a value $u \in D_x$, $v \in D_y$ is a *support* of u if u and v satisfy c_{xy}, that is $(u, v) \in c_{xy}$. u's *image* under c_{xy}, denoted by $I_y(u)$, is the set of all its supports in D_y. The image of a subset of D_x is the union of the images of its values.

k-consistency A constraint network is *k-consistent* if and only if any consistent instantiation of any distinct $k-1$ variables can be consistently extended to any new variable. A network is *strongly* k-consistent if and only if it is j-consistent for all $j \leq k$. A strongly n-consistent network is called *globally consistent*. 2- and 3-consistency are usually called *arc consistency* and *path consistency* respectively. Note that under this definition, we need to add a universal constraint between variables which are not explicitly constrained by the network.

Matrix Representation A constraint c_{xy} can be represented by a boolean matrix whose rows are named by values in D_x and columns by values in D_y. An entry in the matrix is 1 if its row value and column value satisfy c_{xy}, and otherwise is 0. For an example, see c_{xy} in Fig. 1(a). The composition of two constraints can be defined as the multiplication of boolean matrices. Path consistency is described by the intersection and multiplication by many authors.

More materials on these concepts can be found in Mackworth (1977), Montanari (1974), Freuder (1978).

Trees, Chains, and Sets In the following we review trees which play a fundamental role in the analysis of tree convex constraints and introduce some new notations used in this paper. A tree is a connected graph without any cycles. In the rest of the paper, *we always assume there is a root for a tree*. The path between any two nodes (or vertices) is unique and the *distance* of a node to the root is defined as the number of edges in the path between them. Given a tree, a *subtree* is defined as a connected subgraph of the tree, and its *root* is the node closest to the root of the tree.

A *tree on a set* S is a tree whose vertex set is exactly S. We also call a set I a *subtree* of a tree T if there exists a subtree of T whose vertex set is exactly I. An empty set is a *subtree* of any tree. A tree (subtree respectively) becomes a *chain (subchain)* if each node of the tree (subtree) has at most one child. The *last value* of a subchain is the farthest one away from its root. For example, the graph in Fig. 1(b) is a tree on $\{a, b, c, d\}$. $\{a, b, c\}$ is a subtree of it, and $\{a, b\}$ is a subchain whose last value is b.

The *intersection* of two trees is defined as the graph whose vertices and edges are in both trees. It has the following property:

Proposition 1 (Zhang & Yap 2003) *Let T_1, T_2 be two subtrees of some tree. The intersection of T_1 and T_2 is also a subtree of the tree. Furthermore, if the intersection is not empty, the root of the intersection is either the root of T_1 or that of T_2.*

Various convex constraints are introduced below.

Row Convex Constraints A constraint is *row convex* if and only if in each row of its matrix representation, all 1's are consecutive. It is *connected row convex* if it is row convex and all ones in any two neighboring rows are consecutive.

Tree Convex Sets (Zhang & Yap 2003) Sets $E_1, \cdots, E_k$ are *tree convex with respect to a tree T on* $\bigcup_{i \in 1..l} E_i$ if and only if every E_i is a subtree of T. For example, given the tree in Fig. 1(b), sets $\{a, b, c\}$, $\{a, b, d\}$, and $\{a, c, d\}$ are tree convex.

Tree Convex Constraints A constraint c_{xy} is *tree convex with respect to a tree T on D_y* if and only if the images of all values in D_x are tree convex with respect to T. That c_{xy} is tree convex might not mean that c_{yx} is tree convex. Consider c_{xy} in Fig. 1(a). The images of a, b, c are $\{a, b, c\}$, $\{a, c, d\}$, and $\{a, b, d\}$ respectively. They are tree convex with respect

to Fig. 1(b) and thus c_{xy} is tree convex with respect to that tree. The readers are invited to verify that there is no tree to make c'_{xy} (in Fig. 1(c)) tree convex.

Relaxed Tree Convex Constraint Networks

In Zhang & Yap (2003), a tree convex constraint network is defined as a network where all constraints are tree convex with respect to a common tree on *the union of all domains* in the network. In the following definition only the tree structures on *individual* domains matter.

Definition 1 *A constraint network is* tree convex *if and only if there exists a tree on each domain such that every constraint c_{xy} is tree convex with respect to the tree on D_y.*

In the rest of the presentation, tree convexity always refers to our new definition. This new definition is equivalent to the old one if the domains are pairwise disjoint. One advantage of the new definition is that even if the domains of two variables of a constraint c_{xy} share some values, it explicitly allows us to use different tree structures for each domain in deciding the tree convexity of c_{xy} and c_{yx}. Furthermore, it helps simplify the presentation.

Here is the consistency result on tree convex networks:

Theorem 1 *A tree convex constraint network is globally consistent if it is path consistent.*

The proof follows directly from the proof in Zhang & Yap (2003) since the new definition does not affect the essential part of that proof.

Properties of Intersection and Composition of Tree Convex Constraints

A network can be made path consistent by removing from the constraints the tuples which can not be consistently extended to a new variable. It is equivalent to the matrix computation $c_{xy} = c_{xy} \cap (c_{xz} \circ c_{zy})$, where $\circ$ means composition.

As suggested by Theorem 1, we need to study the impact of the intersection and composition operations on the tree convexity of constraints.

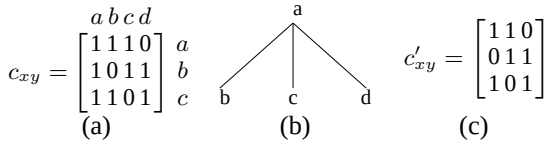
Proposition 2 *Assume constraints c^1_{xy} and c^2_{xy} are tree convex wrt a tree T on the domain D_y. Their intersection is also tree convex.*

Proof. Let $c_{xy} = c^1_{xy} \cap c^2_{xy}$. For any $v \in D_x$, its images under c^1 and c^2 are both subtrees of T. The intersection of the two images is a subtree of T, by Proposition 1. That is, the image of every $v \in D_s$ is a subtree of T. Hence c is tree convex. □

The intersection of two subtrees may be an empty set, which means that after the intersection of two tree convex constraints, the image of a value could be empty. Deleting such a value could make a constraint no longer tree convex. An example is shown is Fig. 1.

It is also interesting to note that a constraint c_{xy} may become tree convex after a sufficient number of values are removed from D_y.

We identify a special class of tree convex constraints which is closed under the operation of deleting values.



(a) (b) (c)

Figure 1: **(a)** Constraint c_{xy} is represented by a matrix. The column $\{a, b, c\}$ after the matrix is the domain of x and the row $\{a, b, c, d\}$ above the matrix the domain of y. **(b)** A tree constructed for the domain of y. **(c)** c'_{xy} is obtained from c_{xy} by deleting the value a from the domain of y.
c_{xy} is tree convex with respect to the tree on D_y in (b), but c'_{xy} is not tree convex with respect to any tree.

Definition 2 *A constraint c_{xy} is* locally chain convex *with respect to a tree on D_y if and only if the image of every value in D_x is a subchain of the tree. A constraint network is* locally chain convex *iff there exists a tree on each domain such that every constraint c_{xy} is locally chain convex with respect to the tree on D_y.*

For example, under the tree for D_y in Fig. 1(b), the constraint in Fig. 1(a) is not locally chain convex because the image of $a \in D_x$ is $\{a, b, c\}$ which is not a subchain of the tree on D_y. In fact, there does not exist any tree to make it chain convex.

Proposition 3 *A locally chain convex constraint network (V, D, C) is still locally chain convex after the removal of any value from any domain.*

Proof. Assume the tree on D_y is T and a value v is removed from D_y. The removal of v does not affect the property of constraint $c_{yx} \in C$ for any $x \in V$. We need to show that for all $x \in V$, constraint c_{xy} is locally chain convex. The deletion of v could make some subchain not connected. By constructing a new tree T'' on D_y, those broken subchains would be connected under T''. Let the children of v be $v_1, \cdots, v_l$. Construct a new tree T' from T by removing v and all edges incident on v. If v is the root of T, construct T'' from T' by adding an edge between v_1 and v_i for all $i (2 \leq i \leq l)$. Otherwise, let p_v be the parent of v in T, and construct T'' from T' by adding an edge between p_v and v_i for all $i (1 \leq i \leq l)$. In either case, we can verify that the claim holds under T''. □

Now let us study a property of the composition of tree convex constraints. Let us use a more intuitive way than matrix multiplication to understand the composition. See Fig. 2. After composing c_{xy} and c_{yz}, the image of a under the composition c_{xz} is $\{a, b, c, d\}$ which is exactly the union of the images of b and d in D_y under c_{yz}. We know that $\{b, d\}$ is the image of a under c_{xy}. In short, the image of a under c_{xz} is the image of $I_y(a)$ under c_{yz}.

By this understanding we can construct an example where the composition does not preserve the property of tree convexity; and we are also able to identify conditions under which tree convexity is preserved under the composition. The intuition behind the following special class of constraints is that given a constraint c_{xy}, the image of any subtree of D_x should be a connected graph (thus a subtree of

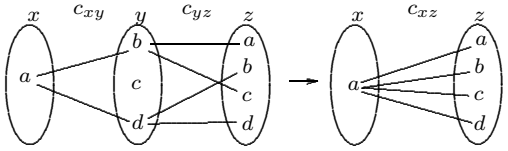


Figure 2: The composition of two constraints. In the diagrams in this paper, a value is drawn as a dot or letter, and a variable is drawn as an ellipse. The values inside an ellipse form the domain of the corresponding variable. The edges between two ellipses form the constraint between the variables.

the tree on D_y).

Definition 3 *A tree convex constraint c_{xy} is* consecutive *wrt a tree T_x on D_x if and only if every two neighboring values on T_x share a common support. A constraint network is* tree convex and consecutive *iff there exists a tree on each domain such that every constraint c_{xy} is tree convex and consecutive wrt the trees on D_y and D_x.*

Proposition 4 *The class of consecutive tree convex constraints is closed under composition.*

Proof. Let c_{xy} and c_{yz} be two consecutive tree convex constraints with trees T_x, T_y, and T_z on D_x, D_y, and D_z respectively, and c_{xz} the composition of c_{xy} and c_{yz}. Firstly, we show that c_{xz} is tree convex. Consider any $v \in D_x$. Let its image in D_y be $I_y(v)$. The image of v under c_{xz} would be $\cup_{b \in I_y(v)} I_z(b)$ where $I_z(b)$ is the image of b under c_{yz} (here we need the intuitive understanding of the composition discussed above). Since the intersection of the images of any neighboring values in $I_y(v)$ is non empty, the union of all the images of values in $I_y(v)$ is connected and thus is a subtree of T_z.

Secondly, we show that c_{xz} is consecutive. Let $u, v \in D_x$ be neighbors. Let $I_z(u)$ and $I_z(v)$ be their images under c_{xz}. Since c_{xy} is consecutive, the images of u and v share a value $w \in D_y$. Hence, the image of w in D_z will appear in both $I_z(u)$ and $I_z(v)$. □

Tractable Tree Convex Constraint Networks

We are now in a position to explore the conditions under which tree convex constraint networks are tractable. A first attempt is to combine the local chain convexity with consecutiveness. However, the composition may destroy the chain convexity, as shown by the example in Fig. 3(a).

The image of a value under the composition is the union of several subchains. This union can not be guaranteed to be a subchain by the consecutiveness of the constraints. We need a stronger restriction.

Definition 4 *A locally chain convex constraint c_{xy} (wrt a tree T_y on D_y) is* strictly union closed *wrt a tree T_x on D_x iff the image of any subchain of T_x is a subchain of T_y.*

Remark Strict union closedness implies consecutiveness for a locally chain convex constraint network; but consecutiveness might not imply union closedness as shown by the example in Fig. 3.

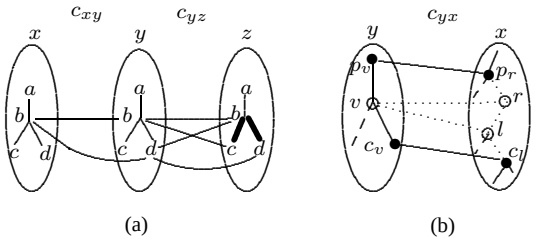


Figure 3: In this diagram, we draw the tree on a domain inside an ellipse. **(a)** Both c_{xy} and c_{yz} are locally chain convex, but their composition is not because the image of $b \in D_x$ under this composition is $\{b, c, d\}$ (the darkened shape), which is not a subchain. **(b)** t_y contains the solid lines in D_y. t_x contains (p_r, r, l, c_l). t_r contains (r, l).

Now we have the main result on a class of tractable constraint networks.

Definition 5 *A constraint network is* locally chain convex and strictly union closed *iff there exists a tree on each domain such that every constraint c_{xy} is locally chain convex (wrt the tree on D_y) and strictly union closed (wrt the tree on D_x).*

Theorem 2 *A locally chain convex and strictly union closed constraint network (V, D, C) can be transformed to an equivalent globally consistent network in polynomial time.*

Proof. We show that after enforcing arc and path consistency on the given network, it is locally chain convex. In accordance with Theorem 1, the new network is globally consistent. It is known that arc and path consistency enforcing (Zhang & Yap 2001) are of polynomial complexity.

Since arc consistency enforcing only removes values from domains, the consequent network is still locally chain convex.

Next we show that the removal of any value $v \in D_y$ preserves strict union closedness. For all $x \in V$, c_{xy} is still strictly union closed (similar to the proof of Proposition 3). Consider a constraint c_{yx} for any variable $x \in V$. If it is still strictly union closed, the claim is true. Otherwise, there exists a subchain, denoted by t_y, of D_y which contains exactly v and its parent and child, such that its image is no longer a connected graph due to the removal of v. See Fig. 3(b). Let t_x be the image of t_y before removing v. After the removal of v, t_x is broken into two chains. Let the gap (removed subchain) in t_x be t_r. Note t_r might not be equal to the image of v due to possible overlapping of the image of v and that of its parent and/or child. Let r be the root and l the last node of t_r. Let p_v and p_r be the parents of v and r respectively, and c_v and c_l the children of v and l respectively. Consider any node $u \in t_r$. We know that u is supported by v, but not by p_v nor by c_v in t_y. Further, since c_{xy} is strictly union closed, the image of t_x must be a subchain containing (p_v, v, c_v). It implies that the image of u must be on or contain the subchain (p_v, v, c_v) . Hence, v is the only support of u. After it is gone, u should also be removed. After the removal of t_r, the image of t_y is now connected and thus a subchain.

Next we show that path consistency enforcing preserves the local chain convexity and strict union closedness. For any constraint c_{xz}, path consistency is usually done by first composing c_{xy} and c_{yz}, and then setting the new constraint between x and z to be the intersection of the composition and c_{xz}.

Firstly, we show that the composition of c_{xy} and c_{yz} is locally chain convex and strictly union closed. 1) For any $v \in D_x$, its image t_y is a subchain. Since c_{yz} is strictly union closed, the image of t_y in D_z is also a subchain. We know that the image of v under the composition is the image of t_y in D_z. 2) For any subchain $t_x \in D_x$, its image t'_y under c_{xy} is also a subchain due to the strict union closedness of c_{xy}. Thanks to the strict union closedness of c_{yz} again, the image of t'_y is a subchain of D_z: that is, the image of t_x under the composition is a subchain.

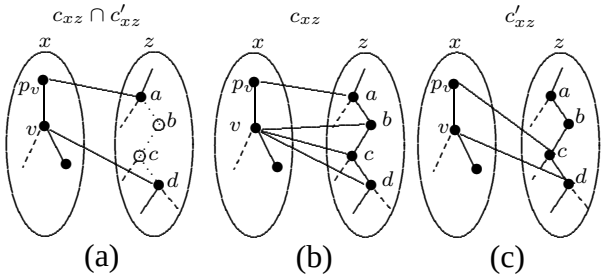


Figure 4: **(a)** $c_{xz} \cap c'_{xz}$. In the intersection, assume b and c are not shared by the images of v under c_{xz} and under c'_{xz}. The constraints c_{xz} and c'_{xz} should have a form as shown in **(b)** and **(c)**.

Secondly, we show that the intersection c''_{xz} of c_{xz} and c'_{xz} ($= c_{xy} \circ c_{yz}$) is locally chain convex and strictly union closed. Local chain convexity is preserved under intersection in accordance with Proposition 3.

We now prove the strict union closedness of the constraint c''_{xz}. In this paragraph, when we refer to *image*, it is under c''_{xz}. Assume there is a subchain t_x in D_x whose image t''_z is not a subchain. Since the intersection does not form a cycle, t''_z must not be connected. Starting from the root of t_x, we find the first value $v \in t_x$ whose image is disjoint from the image of its parent p_v. Assume the image of v is below the image of p_v (the opposite can be proved similarly). Let a be the last value of p_v's image. Let d be the root of v's image. See Fig. 4(a). Let u be any value between (but not including) a and d in D_z. We next prove that there is no support for u. Hence it should be removed and thus the image of t_x is a chain after the deletion.

Let p_v's images under c_{xz} and c'_{xz} be $I(p_v)$ and $I'(p_v)$ respectively. The intersection of $I(p_v)$ and $I'(p_v)$ is a subchain of D_z. Since both $I(p_v)$ and $I'(p_v)$ are subchains, a must be the last value of either $I(p_v)$ or $I'(p_v)$. Assume it is the last value of $I(p_v)$. See Fig. 4(b). This implies p_v is not in u's image $I(u)$ under c_{zx}, since u is between a and d. $I(u)$ has to be below p_v (not including it) because $I(u)$ is a chain. Let $I(v)$ and $I'(v)$ be the images of v under c_{xz} and c'_{xz} respectively. $I(v)$ should include at least d and all values between a and d in the tree D_z because the chain (p_v, v) is strictly union closed. For d is the root of $I(v) \cap I'(v)$, $I'(v)$ includes d but does not include values above d (see Fig. 4(c)). Hence, v is not a support of u, implying that $I'(u)$ has to be above v (not including it). Therefore, the image of u under c''_{xz} is empty because it is the intersection of $I(u)$ and $I'(u)$. In other words, u has no support in the intersection of c_{xz} and c'_{xz}.

Finally, path consistency also involves universal constraints. It can be verified that strict union closedness and local chain convexity of a constraint are preserved when it is composed and intersected with a universal constraint.

There may be universal constraints in the final path consistent network, but the tree convexity of all constraints is sufficient to ensure the global consistency of the network. □

In fact the strict union closedness can be further relaxed in the following way.

Definition 6 *A locally chain convex constraint c_{xy} is* union closed *if and only if for any subchain of D_x, its image is either a subchain of D_y or the whole tree of D_y.*

Locally chain convex constraint networks with relaxed union closedness are still tractable.

Theorem 3 *A locally chain convex and union closed constraint network (V, D, C) can be transformed to an equivalent globally consistent network in polynomial time.*

This can be proved similarly to Theorem 2.

Application of Tree Convex Networks

Scene labeling problems are NP-hard (Kirousis & Papadimitriou 1988). In the following, we show that some scene labeling instances can be naturally modeled by tree convex constraints.

Consider the labeling of the line drawing in Fig. 5 taken from van Beek & Dechter (1995). Traditionally, it is modeled as a constraint network in the following way. Each vertex i is a variable x_i. A value for a variable is the edges incident on this vertex and a label (of `+`, `-` or `>`) on each edge. For example, the edges on vertex 1 form a fork. There are six possible ways to label the fork (Fig. 5), and thus x_1 has six values. Similarly, we have two other domains of values: Arrow and Ell as shown in Fig. 5. The constraint is that any two variables should share the same labeling on their shared edge (see Fig. 6).

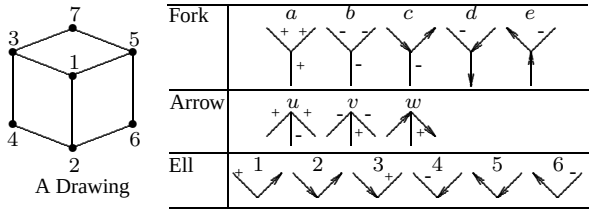


Figure 5: On the left is a line drawing and on the left is a table of the values necessary to model the labeling of this drawing

A distinctive feature of this model is that the values of a variable have complex structures and there is some natural relationship among them. Consider the fork values in Fig. 5. Values c, d, and e have an edge labeled as `-`, and all three edges of b are labeled as `-`. We can let b be the parent of c, d and e, resulting in the subtree $\{b, c, d, e\}$ in Fig. 6(a). Since value a has nothing to do with the rest, we simply connect it with b to make a tree for all fork values. Similarly we have tree structures for Arrow values in Fig. 6(b) and Ell values in Fig. 6(c). The tree on Ell values is not constructed by intuition only, but also by the constraints on the Ell values. Under these trees, the constraints are both locally chain convex and union closed. Note that for constraint c_{12} (and some other constraints), the image of the subchain (a, b, c) of the tree on the domain of x_1 (Fig. 6(a)) is the whole tree of D_2 (the Arrow tree in Fig. 6(b)). Note also that an empty set is taken as a (trivial) subchain of any tree.

This network is globally consistent after enforcing arc and path consistency on it.

Constraints c_{21}, c_{31}, and c_{51} are not row convex. They are "connected row convex" as defined by Deville, Barette, & Van Hentenryck (1997) in the sense that their reduced forms (by deleting the values without any support) are connected row convex (as defined in the preliminary section of this paper).

To see the difference between connected row convexity, and locally chain convexity and union closedness, we modify c_{21}, c_{31}, and c_{51} by allowing v to be compatible with every value in the domains of x_2, x_3 and x_5 respectively. Now the new constraints are no longer connected row convex (as defined by Deville, Barette, & Van Hentenryck (1997)), but are still locally chain convex and union closed.

In this example, we can identify the tree structures for some domains in an intuitive way. However, there also exists some domain to construct a tree for which we need more knowledge about the constraints on it. It is both interesting and challenging to explore the principles of the construction of trees for domain values appearing in the scene labeling problems.

A more general lesson is that by studying the semantics of domain values, we could discover more efficient constraint solving techniques.

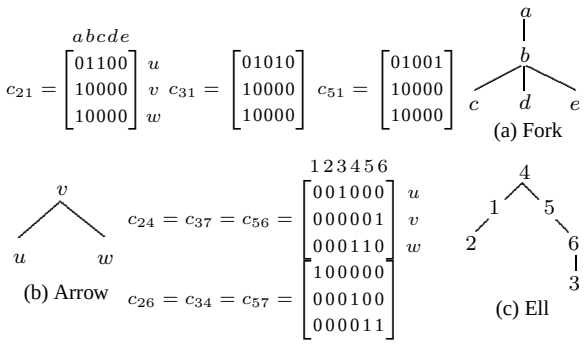


Figure 6: The constraints for labeling the drawing in Fig. 5

Conclusion

We have identified a new tractable class of networks: locally chain convex and union closed networks. This result not only generalizes the existing work, but also shows an direct interaction between the semantics of constraints and the semantics of domain values in deciding a tractable class of problems. This interaction is reflected in the properties of intersection and composition of tree convex constraints. An application of the new tractable class of networks is also presented, demonstrating that tree convexity is a useful way to characterize the semantics of domain values, in addition to the traditional ones like total ordering.

Compared with the work on connected row convexity by Deville et al. (1997), our work reveals more fundamental properties – local chain convexity and union closedness – that determine the tractability of a class of convex constraints. The new tractable class covers more networks.

Acknowledgement

We thank Angela Glover for her help with the writing of this paper.

References

Dechter, R. 1992. From local to global consistency. *Artificial Intelligence* 55:87–107.

Deville, Y.; Barette, O.; and Van Hentenryck, P. 1997. Constraint satisfaction over connected row convex constraints. In *Proceedings of IJCAI-97*, volume 1, 405–411. Nagoya, Japan: IJCAI Inc. (See also Artificial Intelligence 109(1999): 243–271).

Freuder, E. 1978. Synthesizing constraint expressions. *Communications of ACM* 21(11):958–966.

Freuder, E. 1982. A sufficient condition for backtrack-free search. *Journal of The ACM* 29(1):24–32.

Jeavons, P. G.; Cohen, D. A.; and Gyssens, M. 1997. Closure properties of constraints. *Journal of The ACM* 44(4):527–548.

Kirousis, L. M., and Papadimitriou, C. H. 1988. The complexity of recognizing polyhedral scenes. In *Journal of Computer and System Sciences*, volume 37, 14–38.

Mackworth, A. K. 1977. Consistency in networks of relations. *Artificial Intelligence* 8(1):118–126.

Montanari, U. 1974. Networks of constraints: Fundamental properties and applications. *Information Science* 7(2):95–132.

van Beek, P., and Dechter, R. 1995. On the minimality and global consistency of row-convex constraint networks. *Journal of The ACM* 42(3):543–561.

Zhang, Y., and Yap, R. H. C. 2001. Making AC-3 an optimal algorithm. In *Proceedings of IJCAI-01*, 316–321. Seattle: IJCAI Inc.

Zhang, Y., and Yap, R. H. C. 2003. Consistency and set intersection. In *Proceedings of IJCAI-03*, 263–268. Acapulco, Mexico: IJCAI Inc.