On the Minimality and Global Consistency of Row-Convex Constraint Networks *

Peter van Beek [†]
Department of Computing Science
University of Alberta
Edmonton, Alberta, Canada T6G 2H1
vanbeek@cs.ualberta.ca

Rina Dechter [‡]
Department of Computer and Information Science
University of California, Irvine
dechter@ics.uci.edu.

Appears in: *Journal of the ACM,* 42:543-561, 1995.

Abstract

Constraint networks have been shown to be useful in formulating such diverse problems as scene labeling, natural language parsing, and temporal reasoning. Given a constraint network, we often wish to (i) find a solution that satisfies the constraints and (ii) find the corresponding minimal network where the constraints are as explicit as possible. Both tasks are known to be NP-complete in the general case. Task (i) is usually solved using a backtracking algorithm, and task (ii) is often solved only approximately by enforcing various levels of local consistency. In this paper, we identify a property of binary constraints called *row convexity* and show its usefulness in deciding when a form of local consistency called path consistency is sufficient to guarantee that a network is both minimal and globally consistent. Globally consistent networks have the property that a solution can be found without backtracking. We show that one can test for the row convexity property efficiently and we show, by examining

*A preliminary version of this paper appeared in *Proceedings of the Tenth National Conference on Artificial Intelligence,* pages 447-452, AAAI Press / The MIT Press, 1992.

[†] This work was supported in part by the Natural Sciences and Engineering Research Council of Canada.

[‡] This work was supported in part by NSF grant IRI-9157636, by Air Force Office of Scientific Research, AFOSR 900136, by Toshiba of America and by a Xerox grant.

applications of constraint networks discussed in the literature, that our results are useful in practice. Thus, we identify a class of binary constraint networks for which we can solve both tasks (i) and (ii) efficiently. Finally, we generalize the results for binary constraint networks to networks with non-binary constraints.

1 Introduction

Constraint networks have been shown to be useful in formulating such diverse problems as graph coloring [24], scene labeling [16, 28], natural language parsing [22], and temporal reasoning [1]. A constraint network is defined by a set of variables, a domain of values for each variable, and a set of constraints between the variables. Given a constraint network, we often wish to (i) find a solution—an instantiation of the variables that satisfies the constraints and (ii) find the corresponding minimal network where the constraints are as explicit as possible. Finding the minimal network has applications in removing redundant information from a knowledge base [23] and temporal reasoning [25]. However, both tasks are known to be NP-complete in the general case. Task (i) is usually solved using a backtracking algorithm, which is exponential in the worst case but often useful in practice, and task (ii) is often solved only approximately by enforcing various levels of local consistency.

In this paper, we begin by examining constraint networks with only binary constraints. We identify a property of binary constraints called *row convexity* and show its usefulness in deciding when a form of local consistency called path consistency is sufficient to guarantee that a network is both minimal and globally consistent. Globally consistent networks have the property that a solution can be found without backtracking. In particular, we show that if a binary constraint network is path consistent and all of the binary relations are row convex or can be made row convex, then the network is minimal and globally consistent. We also show that if there exists an ordering of the variables and of the domains of the variables such that the binary constraints can be made directionally row convex, then a solution can be found without backtracking. Testing for the row convexity property involves determining whether there exists an ordering of the domains of the variables such that all of the constraints are simultaneously row convex. We show that one can test for the row convexity property efficiently. Thus, we identify a class of binary constraint networks for which we can solve both tasks (i) and (ii) efficiently. We also show, by examining applications of constraint networks discussed in the literature, that our results are useful in practice.

Finally, we generalize the results for binary constraint networks to networks with non-binary constraints. In particular, we generalize the *row convexity* property to non-binary constraints and show its usefulness in deciding when a level of local consistency called strong $2(r-1)+1$ consistency, where r is the maximum arity of the constraints, is sufficient to ensure that the network is globally consistent. As well, we generalize the notion of path consistency for binary constraints to a local consistency condition for non-binary constraints called *relational path-consistency*, and use it to identify an interesting class of non-binary row-convex constraint networks that are globally consistent.

2 Background

We begin with some needed definitions and describe related work.

Definition 1 (binary constraint networks; Montanari [24])
A network of binary constraints $\mathcal{R}$ *is a set* X *of* n *variables* $\{x_1, \ldots, x_n\}$*, a domain* D_i *of possible values for each variable, and a set of binary constraints between variables. A* binary constraint *or relation,* R_{ij}*, between variables* x_i *and* x_j*, is a subset of the Cartesian product of their domains that specifies the allowed pairs of values for* x_i *and* x_j *(i.e.,* $R_{ij} \subseteq D_i \times D_j$*). For the networks of interest here, we require that* $(x_j, x_i) \in R_{ji}$ *if and only if* $(x_i, x_j) \in R_{ij}$*.*

Definition 2 (solution of a binary constraint network)
An instantiation *of the variables in* X *is an* n*-tuple* $(X_1, \ldots, X_n)$*, representing an assignment of* $X_i \in D_i$ *to* x_i*. A* consistent instantiation *of a network is an instantiation of the variables such that the constraints between variables are satisfied. A consistent instantiation is also called a* solution*. A network is* minimal *if each pair of values allowed by each of the constraints participates in at least one consistent instantiation (i.e, if* $(x_i, x_j) \in R_{ij}$*, then* (x_i, x_j) *is part of some consistent instantiation of the network).*

Mackworth [19, 20] defines three properties of networks that characterize local consistency of networks: *node*, *arc*, and *path consistency.*

Definition 3 (path consistent network; Mackworth [19])
A network is path consistent *if and only if, for every triple* (x_i, x_k, x_j) *of variables, we have that, for every instantiation of* x_i *and* x_j *that satisfies the direct relation,* R_{ij}*, there exists an instantiation of* x_k *such that* R_{ik} *and* R_{kj} *are also satisfied.*

Note that the definition of path consistency subsumes arc consistency if in the above definition we do not assume that variables x_i and x_j are distinct. For simplicity, unless otherwise stated, we will assume that path consistency includes arc consistency. Montanari [24] and Mackworth [19] provide algorithms for achieving path consistency that also achieve arc consistency. Freuder [12] generalizes this concept to k-consistency.

Definition 4 (strong k-consistency; Freuder [12, 13])
A network is k-consistent *if and only if given any instantiation of any* $k-1$ *variables satisfying all of the direct relations among those variables, there exists an instantiation of any* k*th variable such that the* k *values taken together satisfy all of the relations among the* k *variables. A network is* strongly k-consistent *if and only if it is* j*-consistent for all* $j \leq k$*.*

Node, arc, and path consistency correspond to strong one-, two-, and three-consistency, respectively. A strongly n-consistent network is called *globally consistent.* Globally consistent networks have the property that any consistent instantiation of a subset of the variables can be extended to a consistent instantiation of all of the variables without backtracking [6]. A strongly n-consistent network is also minimal. However, the converse is not true as it is possible for a network to be minimal but not strongly n-consistent.

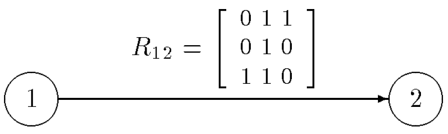
Following Montanari [24], a binary relation R_{ij} between variables x_i and x_j is represented as a (0,1)-matrix with $|D_i|$ rows and $|D_j|$ columns by imposing an ordering on the domains of the variables. A zero entry at row a column b means that the pair consisting of the ath element of D_i and the bth element of D_j is not permitted; a one entry means that the pair is permitted. Two distinguished relations are the identity relation, I, which is represented as the (0,1)-matrix consisting of ones along the diagonal and zeroes everywhere else, and the universal relation, U, which is represented as a (0,1)-matrix consisting of all ones.

A concept central to this paper is the row-convexity of constraints.

Definition 5 (row convex)
A binary relation R_{ij} represented as a (0,1)-matrix is row convex *if and only if in each row all of the ones are consecutive; that is, no two ones within a single row are separated by a zero in that same row.*

Row convex relations generalize functional and monotone relations. A binary relation R_{ij} represented as a (0,1)-matrix is *monotone* if and only if the following conditions hold: if $R_{ij,ab} = 1$ and $c \geq a$, then $R_{ij,cb} = 1$, and if $R_{ij,ab} = 1$ and $c \leq b$, then $R_{ij,ac} = 1$. A binary relation R_{ij} represented as a (0,1)-matrix is *functional* if and only if there is at most one one in each row and in each column of R_{ij}.

We use a graphical notation where vertices represent variables and directed arcs are labeled with the constraints between variables. As a graphical convention, we never show the edges (i,i), and if we show the edge (i,j), we do not show the edge (j,i). Any edge for which we have no explicit knowledge of the constraint is labeled with the universal relation, which is represented as a (0,1)-matrix consisting of all ones. By convention such edges are also not shown. For example, consider the simple constraint network with variables x_1 and x_2 and domains $D_1 = \{a,b,c\}$ and $D_2 = \{d,e,f\}$, shown below.


$$R_{12} = \begin{bmatrix} 0\,1\,1 \\ 0\,1\,0 \\ 1\,1\,0 \end{bmatrix}$$

The constraint R_{12} does not allow, for example, the pair (a,d) but does allow the pairs (a,e), (a,f). It can be seen that the constraint has the row convexity property.

2.1 Related work

Much work has been done on identifying restrictions on constraint networks such that finding a solution and finding the corresponding minimal network can be done efficiently. These restrictions fall into two classes: restricting the topology of the underlying graph of the network and restricting the type of the allowed constraints between variables.

For work that falls into the class of restricting the topology, Montanari [24] shows that if the constraint graph is a tree, path consistency is sufficient to ensure that a network is minimal. Freuder [13, 14] identifies a relationship between a property called the *width* of a constraint graph and the level of local consistency needed to ensure that a solution can be found without backtracking. As a special case, if the constraint graph is a tree, arc consistency is sufficient to ensure that a solution can be found without backtracking. Dechter and Pearl [8] provide an adaptive scheme where the level of local consistency is adjusted on a node-by-node basis. Dechter and Pearl [9] generalize the results on trees to hyper-trees which are called acyclic databases in the database community [3].

For work that falls into the class of restricting the type of the constraints (the class into which the present work falls), Dechter [6] identifies a relationship between the size of the domains of the variables and the level of local consistency needed to ensure that the network is strongly n-consistent, and thus minimal and globally consistent. Montanari [24] shows that path consistency is sufficient to guarantee that a network is both minimal and globally consistent (Montanari uses the term *decomposable*) if the relations are monotone. Van Hentenryck, Deville, and Teng [26] show that arc consistency is sufficient to test whether a network is satisfiable if the relations are from a restricted class of functional and monotone constraints. In general, arc consistency is not sufficient to test the satisfiability of networks with only functional and monotone constraints. To see this, consider the constraint network that arises from trying to color a complete graph of three vertices with two colors. The relations are functional and arc consistent but the network is unsatisfiable. Functional and monotone relations are row convex; hence, it will be seen that our results generalize Montanari's and extend Van Hentenryck et al.'s results. Importantly, in the above work, the problem of deciding whether the constraints have the desired properties is left to the user. We identify an efficient procedure for deciding whether a constraint network can be made row convex.

Finally, for work that falls into both classes, Dechter and Pearl [10] present effective procedures for determining whether a constraint network can be formulated as a *causal theory* and thus a solution can be found without backtracking. Whether a constraint network can be so formulated depends on the topology of the underlying constraint graph and the type of the constraints.

3 A Sufficient Condition for Minimality and Global Consistency

In this section we show the usefulness of row convexity in deciding when path consistency is sufficient to guarantee that a binary network is both minimal and globally consistent. Informally, we show that if a binary network is path consistent and the relations are row convex or can be made row convex, then the network is minimal and globally consistent. We show in Section 4 that a known procedure from graph theory can be used for deciding whether a constraint network can be made row convex.

More formally, the following lemma on the intersection of (0,1)-row vectors that are row convex, is needed in the proof of the result. The lemma is a discrete version of the well-known result that for a set of mutually intersecting intervals there is a point common to all.

Lemma 1 *Let F be a finite collection of (0,1)-row vectors that are row convex and of equal length such that every pair of row vectors in F have a non-zero entry in common; that is, their intersection is not the vector with all zeroes. Then all of the row vectors in F have a non-zero entry in common.*

Proof. Let $v_1, \ldots, v_n$ be row vectors of length k. Let $\text{first}(v_i)$ and $\text{last}(v_i)$ be the position of the first and last one in v_i, respectively. For example, if $v_1 = [\ 0\ 1\ 1\ 1\ 0\ ]$, $\text{first}(v_1) = 2$ and $\text{last}(v_1) = 4$. We define $\text{first}(v_i) = k+1$ and $\text{last}(v_i) = 0$, for the zero vector of length k. We want to show that if the vectors are non-zero and their pairwise intersections are all non-zero, then the intersection of all of the vectors together is a non-zero vector. The condition that a vector, v_i, is non-zero can be expressed as, $\text{first}(v_i) \leq \text{last}(v_i)$. The condition that every pair of vectors have a non-zero intersection can be expressed as, $\forall i \forall j(\text{first}(v_i) \leq \text{last}(v_j))$. It follows that, $\max(\text{first}(v_1), \ldots, \text{first}(v_n)) \leq \min(\text{last}(v_1), \ldots, \text{last}(v_n))$, which is equivalent to saying that the intersection of all of the vectors together is a non-zero vector. □

Theorem 1 *Let $\mathcal{R}$ be a path consistent binary constraint network. If there exists an ordering of the domains $D_1, \ldots, D_n$ of $\mathcal{R}$ such that the relations are row convex, the network is minimal and globally consistent.*

Proof. The theorem is proved by showing that if the network is path consistent and all of the (0,1)-matrices are row convex, then the network is k-consistent for all $k \leq n$. Hence the network is strongly n-consistent and therefore minimal.

To show that the network is k-consistent for all $k \leq n$, we show that it is true for an arbitrary k. Suppose that variables $x_1, \ldots, x_{k-1}$ can be consistently instantiated. That is, let $X_1, \ldots, X_{k-1}$ be an instantiation such that

$$X_i \ R_{ij} \ X_j \qquad\qquad i, j = 1, \ldots, k-1$$

is satisfied. To show that the network is k-consistent, we must show that there exists at least one instantiation, X_k, of variable x_k such that

$$X_i \;\; R_{ik} \;\; X_k \qquad\qquad i = 1, \ldots, k-1 \tag{1}$$

is satisfied. We do so as follows. The $X_1, \ldots, X_{k-1}$ restrict the allowed instantiations of x_k. For each i in Equation 1, the non-zero entries in row X_i of the (0,1)-matrix R_{ik} are the allowed instantiations of x_k. The key is that all of these row vectors are row convex, i.e., the ones are consecutive. Hence, by Lemma 1 it is sufficient to show that any two row vectors have a non-zero entry in common to show that they all have a non-zero entry in common. But arc consistency guarantees that each row vector contains at least one non-zero entry and path consistency guarantees that each pair of row vectors has a non-zero entry in common. Hence, all of the constraints have a non-zero entry in common and there exists at least one instantiation of x_k that satisfies Equation 1 for all i. Because we require that $x_j R_{ji} x_i \Leftrightarrow x_i R_{ij} x_j$ we have also shown that there exists at least one instantiation, X_k, of variable x_k such that

$$X_k \;\; R_{ki} \;\; X_i \qquad\qquad i = 1, \ldots, k-1$$

is satisfied. Hence, we have shown that, for any consistent instantiation of $k-1$ variables, there exists an instantiation of any kth variable such that

$$X_i \;\; R_{ij} \;\; X_j \qquad\qquad i, j = 1, \ldots, k$$

is satisfied. Hence, the network is k-consistent. □

For simplicity, we assumed in the proof of Theorem 1 that all of the domains of the variables are of equal size. The results are easily generalized to domains of unequal size. The proof of the theorem is constructive and gives an algorithm for finding a consistent instantiation. Without loss of generality, we assume the order of instantiation of the variables is $x_1, \ldots, x_n$.

```
INSTANTIATE($\mathcal{R}, n$)
1.  choose an instantiation $X_1$ of $x_1$ that satisfies $R_{11}$
2.  for $i \leftarrow 2$ to $n$
3.      do  $r \leftarrow [1\ 1\ \cdots\ 1]$
4.          for $j \leftarrow 1$ to $i-1$
5.              do  $r \leftarrow r \cap$ (row $X_j$ of $R_{ji}$)
6.          choose an instantiation $X_i$ of $x_i$ that satisfies $r$
```

Intersecting two row vectors in Step 5 takes $O(d)$ time, hence the algorithm is $O(dn^2)$, where n is the number of variables and d is the size of the domains. The path consistency procedure is $O(d^5n^3)$ [21] which can be improved to $O(d^3n^3)$ by using a more complicated data structure [15]. So, we can find a solution and the minimal network for the class of constraint networks characterized by Theorem 1 in $O(d^3n^3)$ time.

Example 1. Scene labeling in computer vision [5, 16] can be formulated as a problem on constraint networks. We use an example to illustrate the application of Theorem 1. Figure 2 shows the variables in the constraint network and the constraints; Figure 1 shows the domains of the variables and the ordering imposed. For example, variable x_1 in Figure 2 is a fork and can be instantiated with any one of the five labelings shown in Figure 1. The constraints between variables are simply that, if two variables share an edge, then the edge must be labeled the same at both ends. Not all of the constraints are row convex. However, once the path consistency algorithm is applied, the relations become row convex. Therefore, in this example, no reordering of the domains is needed in order to satisfy the theorem. The Instantiate procedure can be used to find a solution. The four possible solutions are shown in Figure 3.

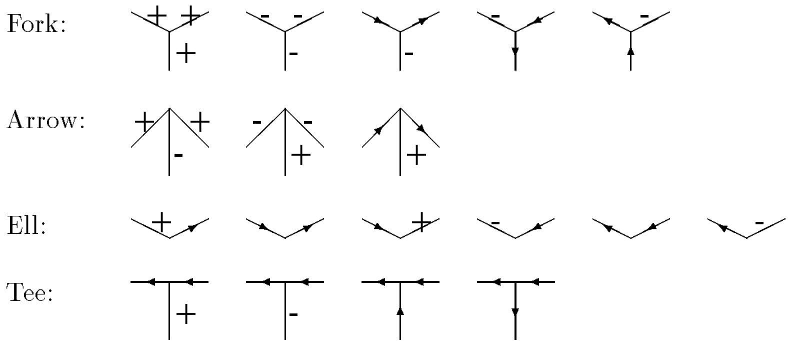
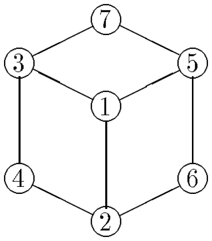


Figure 1: Huffman-Clowes junction labelings

The scene-labeling problem has been shown to be NP-complete in the general case [18]. We are attempting to prove the conjecture that constraint networks arising from orthohedral scenes are row convex once path consistency is applied.

$$R_{21} = \begin{bmatrix} 0\ 1\ 1\ 0\ 0 \\ 1\ 0\ 0\ 0\ 0 \\ 1\ 0\ 0\ 0\ 0 \end{bmatrix} \quad R_{31} = \begin{bmatrix} 0\ 1\ 0\ 1\ 0 \\ 1\ 0\ 0\ 0\ 0 \\ 1\ 0\ 0\ 0\ 0 \end{bmatrix} \quad R_{51} = \begin{bmatrix} 0\ 1\ 0\ 0\ 1 \\ 1\ 0\ 0\ 0\ 0 \\ 1\ 0\ 0\ 0\ 0 \end{bmatrix}$$



$$R_{24} = R_{37} = R_{56} = \begin{bmatrix} 0\ 0\ 1\ 0\ 0\ 0 \\ 0\ 0\ 0\ 0\ 0\ 1 \\ 0\ 0\ 0\ 1\ 1\ 0 \end{bmatrix}$$

$$R_{26} = R_{34} = R_{57} = \begin{bmatrix} 1\ 0\ 0\ 0\ 0\ 0 \\ 0\ 0\ 0\ 1\ 0\ 0 \\ 0\ 0\ 0\ 0\ 1\ 1 \end{bmatrix}$$

Figure 2: Scene labeling constraint network

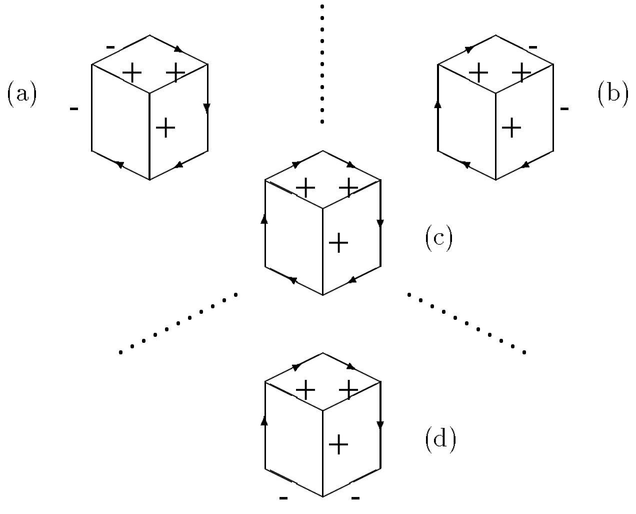


Figure 3: Solutions: (a) stuck on left wall, (b) stuck on right wall, (c) suspended in mid-air, (d) resting on floor.

As an immediate corollary of Theorem 1, if we know that the result of applying path consistency will be that all of the relations will be row convex, we can guarantee *a priori* that path consistency will find the corresponding minimal network and that the minimal network will be globally consistent. To use the path consistency algorithms, three operations on relations are needed: composition, intersection, and inverse[1]. Thus, if the relations in our constraint network are row convex and remain row convex under these operations, the result applies.

Corollary 1 *Let $\mathcal{L}$ be a set of (0,1)-matrices closed under composition, intersection, and transposition such that each element of $\mathcal{L}$ is row convex. Let $\mathcal{R}$ be a binary constraint network with all relations taken from $\mathcal{L}$. The path consistency algorithm will correctly determine the minimal network of $\mathcal{R}$. Furthermore, the minimal network will be globally consistent.*

Proof. This can be proved by a simple rewriting of the proof for Theorem 1. □

Example 2. Let the domains of the variables be of size two. The set of all 2×2 (0,1)-matrices is closed under composition, intersection, and transposition and each 2×2 (0,1)-matrix is row convex. Hence, the corollary applies to all binary constraint networks with domains of size two. As a specific example, the Graph 2-coloring problem can be formulated using such constraint networks. Dechter [6, p. 93] also shows, but by a different method, that a strongly 3-consistent (or path consistent) bi-valued network is minimal.

Example 3. Let the domains of the variables be finite subsets of the integers and let a binary constraint between two variables be a conjunction of linear equalities and inequalities of the form $ax_i - bx_j = c$, $ax_i - bx_j < c$, or $ax_i - bx_j \leq c$, where a, b, and c are integer constants. For example, the conjunction

$$(3x_i + 2x_j \leq 3) \wedge (-4x_i + 5x_j < 1)$$

is an allowed constraint between variables x_i and x_j. A network with constraints of this form can be formulated as an integer linear program where each constraint is on two variables and the domains of the variables are restricted to be finite subsets of the integers. However, it can be shown that each element in the closure under composition, intersection, and transposition of the resulting set of (0,1)-matrices is row convex, provided that when an element is removed from a domain by arc consistency, the associated (0,1)-matrices are "condensed." This is best illustrated through an example. Let $D_i = D_j = \{-1, 0\}$ and $D_k = \{-1, 0, 1\}$ and let R_{ij} be the matrix constructed from the constraint

[1] When the relations are represented as (0,1)-matrices, these operations correspond to binary matrix multiplication, binary matrix intersection, and transposition of the matrix, respectively. The reader may consult [19, 24] for details.

$x_i + x_j < 0$ and R_{jk} be constructed from $-2x_j + x_k = 1$.

$$R_{ij} = \begin{bmatrix} 1\ 1 \\ 1\ 0 \end{bmatrix}, \quad R_{jk} = \begin{bmatrix} 1\ 0\ 0 \\ 0\ 0\ 1 \end{bmatrix}, \quad R_{ik} = \begin{bmatrix} 1\ 0\ 1 \\ 1\ 0\ 0 \end{bmatrix}.$$

The matrix R_{ik}, the result of composing R_{ij} and R_{jk}, is not row convex. However, there is no solution with x_k assigned 0, so D_k becomes $\{-1, 1\}$ and row convexity can be restored by removing the middle column from matrices R_{ik} and R_{jk}. Hence, by Corollary 1 we can guarantee that the result of path consistency will be the minimal network and the network will be globally consistent. Two special cases are a restricted and discrete version of Dechter, Meiri, and Pearl's [7] continuous, bounded difference framework for temporal reasoning and a restricted and discrete version of Vilain and Kautz's [27] qualitative framework for temporal reasoning.

4 Identifying Row-Convex Relations

As noted in the scene-labeling example, when constructing a constraint network and the (0,1)-matrices that represent the constraints, we must impose an ordering on the domains of the variables. Sometimes a natural ordering exists, as when the domain is a finite subset of the integers, but often the ordering imposed is arbitrary and with no inherent meaning. An unlucky ordering may hide the fact that the constraint network really is row convex or, more properly, can be made row convex. How can we distinguish this case from the case where no ordering of the domains will result in row convexity? The following theorem shows that we can test for this property efficiently.

Theorem 2 (Booth and Lueker [4]) *An $m \times n$ (0,1)-matrix specified by its f nonzero entries can be tested for whether a permutation of the columns exists such that the matrix is row convex in $O(m + n + f)$ steps.*

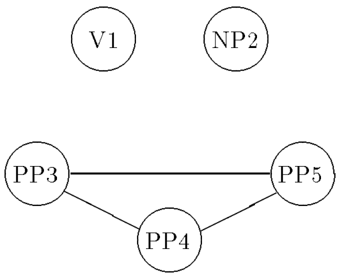
Example 4. Maruyama [22] shows that natural language parsing can be formulated as a problem on constraint networks. In this framework, intermediate parsing results are represented as a constraint network and every solution to the network corresponds to an individual parse tree. We use an example network from [22] to illustrate the application of Theorems 1 and 2. Consider the following sentence.

Put	the block	on the floor	on the table	in the room.
V1	NP2	PP3	PP4	PP5

The sentence is structurally ambiguous (there are fourteen different parses) as there are many ways to attach the prepositional phrases. Figure 4 shows the original ordering of the domains; Figure 5 shows the variables in the constraint network and the constraints. For example, the constraint between variable PP3 and variable PP4 is given by the (0,1)-matrix at row PP3 column PP4 of the table in Figure 5 (the symbol I in the figure denotes the identity matrix—the (0,1)-matrix consisting of ones along the diagonal and zeroes everywhere else). Maruyama states that a "simple backtrack search can generate the 14 parse trees of the sentence from the constraint network at any time." While the network is path consistent, it can be seen that the constraints are not all row convex given the original domain ordering used in [22]. However, using the new domain ordering shown in Figure 4, the constraints are now row convex. Hence, the INSTANTIATE procedure from the previous section can be used to find a solution in a backtrack-free manner.

Variable	Domain	
	Original ordering	New ordering
V1	{Rnil}	{Rnil}
NP2	{O1}	{O1}
PP3	{L1, P2}	{L1, P2}
PP4	{L1, P2, P3}	{P2, P3, L1}
PP5	{L1, P2, P3, P4}	{P3, P4, L1, P2}

Figure 4: Variables and domains for parsing example



	V1	NP2	PP3	PP4	PP5
V1	I	1	1 1	1 1 1	1 1 1 1
NP2	1	I	1 1	1 1 1	1 1 1 1
PP3	1 1	1 1	I	1 0 1 1 1 1	1 0 1 1 1 1 1 1
PP4	1 1 1	1 1 1	1 1 0 1 1 1	I	1 0 0 1 1 1 0 1 1 1 1 1
PP5	1 1 1 1	1 1 1 1	1 1 0 1 1 1 1 1	1 1 1 0 1 1 0 0 1 1 1 1	I

Figure 5: Left: Parsing constraint network; Right: Table representation of constraint network

Let $\mathcal{R}$ be a path consistent binary constraint network. It remains to show how Theorem 2 can be used to determine whether an ordering of the domains of the variables exists such that all of the (0,1)-matrices R_{ij}, $1 \leq i, j \leq n$, are row convex. The procedure is simple: for each variable, x_j, we take the matrix defined by stacking up R_{1j} on top of R_{2j} on top of $\cdots R_{nj}$ and test whether the matrix can be made row convex. For example, with reference to Figure 5, for variable PP4 we would test whether the columns of the matrix consisting of the 3 columns and 11 rows under the column heading PP4 can be permuted to satisfy the row convexity property. In this example such a permutation exists and corresponds to the new ordering of the domain of variable PP4 shown in Figure 4.

It is, of course, not true that for every path consistent network there exists an ordering of the domains such that all of the constraints are simultaneously row convex. However, in those cases where such an ordering does not exist, a weaker property may hold, where the network is *directionally* row convex relative to a particular ordering of the variables.

Definition 6 (directionally row convex)
Given an ordering of the variables $x_1, \ldots, x_n$, a binary constraint network $\mathcal{R}$ is directionally row-convex *if each of the (0,1)-matrices R_{ij}, where variable x_i occurs before variable x_j in the ordering, is row convex.*

Theorem 3 *Let $\mathcal{R}$ be a path consistent binary constraint network. If there exists an ordering of the variables $x_1, \ldots, x_n$ and of the domains $D_1, \ldots, D_n$ of $\mathcal{R}$ such that $\mathcal{R}$ is directionally row convex, then a solution can be found without backtracking.*

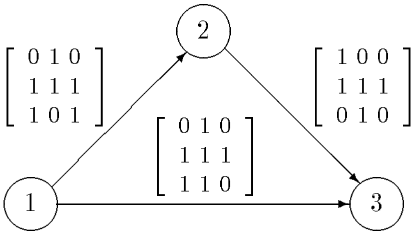
Proof. This can be proved by a simple rewriting of the proof for Theorem 1. □

Example 5. Consider the constraint network with three variables and domains $D_1 = D_2 = D_3 = \{$a,b,c$\}$ as shown in Figure 6. While this example is path consistent, no ordering of the domain of x_2 exists that will simultaneously make the (0,1)-matrices R_{12} and R_{32} row convex. However, order $D_2 = \{$a,c,b$\}$ satisfies the condition of Theorem 3 and the variables can be instantiated in the order x_1, x_2, x_3 using the INSTANTIATE procedure, and it can be guaranteed that no backtracking is necessary[2].

An algorithm for finding a directionally row convex ordering of the variables and of their respective domains, if such an ordering exists, is given below.

```
FINDORDER($\mathcal{R}$, $n$)
$L \leftarrow \{1, 2, \ldots, n\}$
for $m \leftarrow n$ downto 1
    do  find a $j \in L$ for which there exists an ordering of
        domain $D_j$ such that $\forall i \in L$, $R_{ij}$ is row convex
        (if no such $j$ exists, then report failure and halt)
        put variable $x_j$ at position $m$ in the ordering
        $L \leftarrow L - \{j\}$
```

[2] The example was chosen to illustrate the application of the theorem as simply as possible; in actuality, path consistency is sufficient for guaranteeing the minimality and global consistency of any three node network.



	1	2	3
1	*I*	0 1 0 1 1 1 1 0 1	0 1 0 1 1 1 1 1 0
2	0 1 1 1 1 0 0 1 1	*I*	1 0 0 1 1 1 0 1 0
3	0 1 1 1 1 1 0 1 0	1 1 0 0 1 1 0 1 0	*I*

Figure 6: Left: A path consistent network that can be made directionally row convex; Right: Table representation of constraint network

Lemma 2 *Algorithm* FindOrder *correctly finds a directionally row convex ordering of the variables and of their respective domains, if such an ordering exists, in $O(d^2n^3)$ time.*

Proof. In the worst-case, Step 3 of the algorithm involves testing, for each of the n elements of L, whether an $nd \times d$ (0,1)-matrix can be made row convex, where n is the number of variables and d is the size of the domains. By Theorem 2, each test can be done in $O(d^2n)$ time, hence the algorithm is $O(d^2n^3)$.

For the proof of correctness, it is sufficient to note that at each stage of the algorithm, there may be more than one $j \in L$ that satisfies the condition (Step 3), but that any j that satisfies the condition for position m in the constructed ordering will still satisfy the condition for all positions $m' < m$ (recall that we are constructing the ordering in reverse). That is, choosing a j and removing j from L only makes the problem smaller and simpler, introducing new choices but never blocking any previously available choices of j. □

Once (and if) a backtrack-free ordering has been found using the FindOrder procedure, a solution can be found using the Instantiate procedure. Recall that a precondition of the algorithms is that the network be path consistent. Ensuring path consistency once again dominates the overall computation since the complexity of the path consistency procedure is $O(d^3n^3)$ [15]. So, we can find a solution for the class of constraint networks characterized by Theorem 3 in $O(d^3n^3)$ time.

5 Non-Binary Row-Convex Constraints

In this section we generalize the results for binary row-convex constraint networks to non-binary row-convex networks. To do this, we generalize row-convexity from binary relations to r-ary relations, namely relations having r variables. We first need the following notational conventions and definitions regarding general networks.

Definition 7 (relations)
Given a set of variables $X = \{x_1, \ldots, x_r\}$, each associated with a domain of discrete values $D_1, \ldots, D_r$, respectively, a relation *(or, alternatively, a* constraint*) R over X is any subset*

$$R \subseteq D_1 \times \cdots \times D_r.$$

Given a relation R on a set X of variables and a subset $Y \subseteq X$, we denote by $Y = y$, or by y, an instantiation of the variables in Y, called a subtuple *and by $\sigma_{Y=y}(R)$ the selection of those tuples in R that agree with $Y = y$. We denote by $\Pi_Y(R)$ the projection of the relation R on the subset Y; that is, a tuple over Y appears in $\Pi_Y(R)$ if and only if it can be extended to a full tuple in R. The operator $\bowtie$ is the join operator of the relational database model.*

Definition 8 (constraint networks)
A constraint network *$\mathcal{R}$ over a set X of variables $\{x_1, \ldots, x_n\}$, is a set of relations denoted $R_1, \ldots, R_t$, each defined on a subset of variables $S_1, \ldots, S_t$ respectively. A relation in $\mathcal{R}$ specified over $Y \subseteq X$ is also denoted R_Y. The set of subsets $S = \{S_1, \ldots, S_t\}$ on which constraints are specified is called the* scheme *of $\mathcal{R}$. The network $\mathcal{R}$ represents its set of all consistent solutions over X, denoted $\rho(\mathcal{R})$ or $\rho(X)$, namely,*

$$\rho(\mathcal{R}) = \{x = (X_1, \ldots, X_n) \mid \forall S_i \in S, \Pi_{S_i}(x) \in R_i\}.$$

For non-binary networks the notion of *consistency of a subtuple* can be defined in several ways. We will use the following definition. A subtuple over Y is consistent if it satisfies all of the constraints defined over Y or any subsets of Y.

Definition 9 (consistency of a subtuple)
A subtuple $Y = y$ is consistent relative to $\mathcal{R}$ iff for all $S_i \in S$, such that $S_i \subseteq Y$,

$$\Pi_{S_i}(y) \in R_i,$$

where a projection of a tuple y over a set of variables S is the subset of y restricted to values assigned to S. $\rho(Y)$ is the set of all consistent instantiations of the variables in Y. One can view $\rho(Y)$ as the set of all solutions of the subnetwork defined by Y.

Informally, an r-ary relation is row convex if, in the multidimensional matrix representing the constraint, each vector that is parallel to one of the axes has the consecutive 1's property.

Definition 10 *An r-ary relation R on a set X of variables $\{x_1, \ldots, x_r\}$ is* row convex *if for any subset of $r-2$ variables $Z \subseteq X$ and for every instantiation, z, of the variables in Z, the binary relation $\Pi_{(X-Z)}(\sigma_z(R))$ is row convex.*

For binary constraint networks, we identified an efficient procedure for determining whether a domain ordering exists such that the relations are all row convex (see Section 4). It is an open question whether such an efficient procedure exists for r-ary constraint networks. However, there are practical examples, such as bi-valued relations and implicational constraints, where we can assert that the r-ary relations of a network are all row convex.

Example 6. Any bi-valued relation is row convex since the operations of *selection* σ, and *projection* Π, generate a binary bi-valued relation which is always row convex. In particular, the set of models of any propositional formula is row convex. Therefore, formulas given in conjunctive normal form *(CNF)*, namely as conjunctions of clauses, are row convex constraint networks.

In the following theorem we generalize the results obtained earlier for binary row-convex constraint networks (Theorem 1) to r-ary row-convex networks.

Theorem 4 *Let $\mathcal{R}$ be a network of relations whose arity is r or less that is strongly $2(r-1)+1$ consistent. If there exists an ordering of the domains $D_1, \ldots, D_n$ of $\mathcal{R}$ such that the relations are row convex, the network is globally consistent.*

Proof. The proof is a simple extension of the ideas in the proof for the binary case. Assume that the network is strongly $2(r-1)+1$ consistent. We show that for any $i \geq 2(r-1)+2$, the network is also i-consistent. Let $X' = (X_1, X_2, \ldots, X_{i-1})$ be a consistent instantiation of $i-1$ variables and let x_i be an arbitrary new variable. We show that there exists a value X_i of x_i such that the extended tuple $(X_1, X_2, \ldots, X_{i-1}, X_i)$ is consistent. This means that any relation $R_t \in \mathcal{R}$ involving variable x_i and a subset of the variables from $\{x_1, \ldots, x_{i-1}\}$ of size $r-1$ or less should be satisfied by such an extension. Since all of the constraints are row convex, all of the values of x_i that agree with the values in X', and that are allowed by R_t, can be listed as a row convex vector relative to the order of x_i's domain.

We claim that any two such constraints have a common consistent value in x_i. Consider two arbitrary constraints, R_1 and R_2. Let $Y_1 \subseteq X'$ and $Y_2 \subseteq X'$ be all of the values on which R_1 and R_2 are defined respectively. Since a constraint's arity is at most r, $Y_1 \cup Y_2$ is a consistent subset of values of size at most $2(r-1)$. Since the problem is strongly $2(r-1)+1$ consistent there must exist a value of x_i that satisfies both R_1 and R_2.

We know, therefore, that any two row vectors corresponding to x_i of any two relevant constraints, intersect. Since they are all row convex they all have a common value in their intersection. Thus x_i can be consistently extended and the claim is proved. □

We may be tempted to translate the above condition into a solution procedure of r-ary row convex network as follows. First enforce strong $2(r-1)+1$ consistency, and then solve the problem in a backtrack-free manner. This, however will not work in general, since while enforcing $2(r-1)+1$ consistency we may need to record constraints whose arity is greater then r. Consequently, the resulting network may not be globally consistent.

We now show that the above theorem can be rephrased in terms of a local consistency condition between *constraints* that resembles path consistency.

Definition 11 (relational arc and path-consistency)
Let $\mathcal{R}$ be a network of relations over a set of variables X, and let R_S and R_T be two relations in $\mathcal{R}$, where $S, T \subseteq X$. A network is relationally arc-consistent if for any $R_S \in \mathcal{R}$ and for every variable $x \in S$,

$$\rho(S - \{x\}) \subseteq \Pi_{S-\{x\}}(R_S). \tag{2}$$

We say that R_S and R_T are relationally path-consistent relative to variable *x iff any consistent instantiation of the variables in $(S \cup T) - \{x\}$, has an extension to x that satisfies R_S and R_T simultaneously; that is, iff*

$$\rho(A) \subseteq \Pi_A(R_S \bowtie R_T), \tag{3}$$

where $A = (S \cup T) - \{x\}$. (Recall that $\rho(A)$ is the set of all consistent instantiations of the variables in A). A pair of relations R_S and R_T is relationally path-consistent *iff it is relationally path-consistent relative to each variable in $S \cap T$. A network of relations is relationally path-consistent iff every pair of relations is relationally path-consistent.*

Note that the definition of relational path-consistency subsumes relational arc-consistency if in (3) we do not assume distinct pairs of relations. For simplicity, unless otherwise stated, we will assume that relational path-consistency includes relational arc-consistency.

Example 7. Consider the following *CNF* formula:

$$\varphi = \{(f \vee x \vee y \vee \neg z) \wedge (f \vee z) \wedge (x \vee y \vee f)\}.$$

Formula φ can be viewed as a constraint network where each clause corresponds to a constraint defined by its models. It is easy to see that the clauses are always relationally arc-consistent since their projection on any subset of propositional symbols results in the universal relations allowing everything. The first two clauses are also relationally path-consistent relative to f since any truth assignment to x, y, z can be extended by assigning "true" to f, thereby satisfying both clauses. Similarly, the two clauses are relationally path-consistent relative to z since any consistent assignment to x, y, f has to satisfy the third clause of φ and therefore by assigning $z = true$ the first two clauses are satisfied. The reader is invited to check that the entire set of clauses is relationally path-consistent.

Example 8. Consider the following linear constraints over the non-negative integers:

$$(1)\ f + x + y + z \leq 1, \qquad (2)\ f - z \leq -1$$

The constraints are not relationally arc-consistent. For instance, $f = 1, z = 2$ satisfies (2), and thus is in $\rho(f, z)$, but it cannot be extended to (1). In order to make constraint (1) relationally arc-consistent we have to add all of its projections having the form: for any subset $T \subset \{f, x, y, z\}$ $\sum_{x_i \in T} x_i \leq 1$. To make (2) relationally arc-consistent we have to add only $z \geq 1$. Once these sets of linear inequalities are added to (1) and (2) we have relational arc-consistency. Still we do not have relational path-consistency. For instance, the instantiation $f = 0, x = 1, y = 0$ satisfies all of the constraints, but it cannot be consistently extended to a value satisfying (1) and (2). If we add the constraint (3) $2f + x + y \leq 0$, constraints (1) and (2) will become relationally path-consistent relative to z since constraint (3) will disallow the partial assignments $f = 0, x = 1, y = 0$. Constraints (1) and (2) are also relationally path-consistent relative to f since any consistent instantiations of x, y, and z will have to satisfy the two constraints $x + y + z \leq 1$ and $-z \leq -1$ that were added to make the network relationally arc-consistent. Once these constraints are obeyed there is an extension to $f = 0$ that satisfies (1) and (2) simultaneously.

We now show that relational path-consistency is sufficient to ensure global consistency when the relations are row convex.

Theorem 5 *Let $\mathcal{R}$ be a network of relations that is relationally path-consistent. If there exists an ordering of the domains $D_1, \ldots, D_n$ of $\mathcal{R}$ such that the relations are row convex, the network is globally consistent.*

Proof. Assume that the network is relationally path-consistent. Let $X' = (X_1, X_2, \ldots, X_{i-1})$ be a consistent instantiation of $i - 1$ variables. We show that for any x_i, there exist a value X_i of x_i such that the extended tuple $(X_1, X_2, \ldots, X_{i-1}, X_i)$ is consistent. This means that any given applicable relation $R_Y \in \mathcal{R}$ that is defined over $\{x_1, \ldots, x_i\}$ should be satisfied. Since all of the constraints are relationally arc-consistent, they must be consistent with X'. Since they are also row convex, all of the values of x_i that are allowed by R_Y and that are restricted to the values in X' can be listed as a row convex vector.

Relational path-consistency implies that any two rows corresponding to any two constraints, must have a common value in their intersection. Since they are all row convex they all have a common value in their intersection and the claim is proved. □

Clearly, when all constraints are binary, relational path-consistency is identical to path consistency in binary networks. As demonstrated above, relational path-consistency can be enforced on a network that does not possess this level of consistency. Below we present algorithm Relational-PC, a brute-force algorithm for enforcing relational arc- and path-consistency on a network $\mathcal{R}$.

```
Relational-PC($\mathcal{R}$)
1. repeat
2.        $Q \leftarrow \mathcal{R}$
3.        for every two relations $R_S$, $R_T \in Q$ (not necessarily distinct)
          and for every $x \in S \cap T$
4.            do  $A \leftarrow (S \cup T) - \{x\}$
5.                $R_A \leftarrow R_A \cap \Pi_A(R_S \bowtie R_T)$
6. until $Q = \mathcal{R}$
```

The algorithm takes any pair of relations that may or may not be relationally path-consistent and assures their relational path-consistency by enforcing a relation (i.e., a constraint) on a subset of their variables. We call the operation in Step 5 of the algorithm *extended composition*, since it generalizes the composition operation defined on binary relations. The Relational-PC algorithm computes the closure of $\mathcal{R}$ with respect to extended composition. If an empty relation is generated the network is inconsistent. We can conclude that:

Theorem 6 *For any network $\mathcal{R}$, whose closure under extended composition is row convex,* Relational-PC *will either decide that the network is inconsistent or else compute an equivalent globally-consistent network of $\mathcal{R}$.*

Proof. Follows immediately from Theorem 5 and from the fact that algorithm Relational-PC generates a relationally path-consistent network. □.

While enforcing *variable-based* path-consistency can be done in polynomial time, it is unlikely that relational path-consistency can be achieved tractably, since, as we will shortly see, it solves the NP-complete problem of propositional satisfiability. A more direct argument suggesting an increase in time and space complexity is the fact that the algorithm may need to record relations of arbitrary arity. Note that relational arc-consistency can be enforced in time polynomial in the arity of its constraints.

Example 9. Implicational constraints [17] can be shown to characterize certain types of scene constraints in vision. These constraints are defined as follows: A binary constraint R_{ij}, between variables x_i and x_j, is implicational if any value of x_i either implies the value of x_j or it does not constrain it at all, and vice-versa. A higher order relation is implicational if all of the binary constraints generated from all projections on all pairs are implicational. It is easy to see that implicational constraints are necessarily row convex. They generalize the notions of monotone and functional binary constraints. Consequently, whenever implicational constraints are closed under extended composition their consistency can be determined by Relational-PC. Note that the consistency of implicational constraints can be decided in polynomial time [17].

Example 10. Consider a set of r-ary linear inequalities, where the domains of the variables are finite subsets of integers and the r-ary constraints over a subset of variables $x_1, \ldots, x_r$ are of the form $a_1 x_1 + \cdots + a_r x_r \leq c$, where the a_i's are rational constants. This is a general integer linear program. The r-ary inequalities define corresponding r-ary relations that are row convex. Therefore, whenever the extended composition of any two of these r-ary relations can be guaranteed to be row-convex their consistency can be determined by Relational-PC.

Example 11. Bi-valued relations are row convex and closed under composition. Consequently, from Theorem 6, bi-valued networks can be solved by Relational-PC. In particular, the satisfiability of propositional *CNFs* can be decided by Relational-PC. In this case, the composition operation (Step 5 of the Relational-PC algorithm) takes the form of pair-wise resolution. For more details see [11].

As with variable-based local consistency, we can improve the efficiency of enforcing relational consistency by enforcing only *directional* consistency. Below we present algorithm Directional-Relational-PC, which enforces relational path-consistency on a network $\mathcal{R}$, relative to a given ordering d of the variables $x_1, \ldots, x_n$. We will call the network generated by the algorithm the *directional closure of* $\mathcal{R}$.

```
Directional-Relational-PC($\mathcal{R}, d, n$)
for $i \leftarrow n$ downto 1
    do for every pair of relations $R_S$ and $R_T$ (not necessarily distinct)
       involving variable $x_i$ and any variable $x_j$, $j < i$ in the ordering $d$
           do $A \leftarrow (S \cup T) - \{x_i\}$
              $R_A \leftarrow R_A \cap \Pi_A(R_S \bowtie R_T)$
              if $R_A$ is the empty relation
                  then exit and return the empty network
```

While the algorithm is incomplete for deciding consistency in general, it is complete for row convex relations that are closed under extended composition. In addition, like similar algorithms for imposing directional consistency, Directional-Relational-PC's worst-case complexity can be bounded as a function of the topological structure of the problem via parameters like the *induced width* of the graph [8]. We elaborate on these issues below.

Theorem 7 (Completeness) *For any network $\mathcal{R}$, whose directional closure is not empty and row convex, the* Directional-Relational-PC *algorithm computes an equivalent network of $\mathcal{R}$ that is backtrack-free along the ordering d.*

Proof. Clearly the directional closure of $\mathcal{R}$ is equivalent to $\mathcal{R}$. What needs to be shown is that the directional closure relative to d, is backtrack-free along the ordering d. We will prove this by induction on d. For the first variable x_1 (remember that x_1 is processed last) there must be a value in the domain of x_1 that is allowed since otherwise the domain will be empty and the directional closure will be empty. Assume now that we have already consistently instantiated the first $i-1$ variables as $X' = (X_1, \ldots, X_{i-1})$. Lets x_i be the next variable. We claim that (i) every *applicable* constraint in the directional closure, defined on x_i and a subset of $\{x_1, \ldots, x_{i-1}\}$ must be consistent with X'; and (ii) any two such constraints must have a common extension in x_i. Otherwise, if (i) is violated, there is an applicable constraint $R' = R_{S \cup \{x_i\}}$, that is not consistent with X'. However, the operation in Step 4 of the algorithm in its ith iteration, generates, R'_S, a constraint applicable to X' that is *not* satisfied by X', in contradiction to the assumed consistency of X'. If (ii) is violated, then there are two relations $R_{S \cup \{x_i\}}$ and $R_{T \cup \{x_i\}}$ that are individually consistent with X' but have no common extension in x_i. However the extended composition over these two relation, when x_i was processed generated a relation over $S \cup T$ that should have been consulted when testing X''s consistency and should have disallowed X', again, yielding a contradiction.

Now that we have established that all pairs of applicable constraints are both consistent with X' and since each pair of applicable relations has a common extension to x_i, the row-convexity property guarantees that they all have a common extension to x_i. $\Box$.

It is important to note that for any ordering d, the complexity of the algorithm can be bounded as a function of its induced width $W*(d)$. A network of constraints $\mathcal{R}$ can be associated with a constraint graph, where each node is a variable and two variables that appear in one constraint are connected. A general graph can be embedded in a *clique-tree* namely, in a graph whose cliques form a tree-structure. The induced width, $W*$, of such an embedding is its maximal clique size and the induced width $W*$ of an arbitrary graph is the minimum induced width over all of its tree-embeddings. It is well known that finding the minimal width embedding is NP-hard [2], nevertheless every ordering of the variables d, yields a simple to compute upper bound denoted $W^*(d)$ (see [9]). The complexity of Directional-Relational-PC along d can be bounded as a function of $W*(d)$ of its constraint graph. Specifically, it was shown that the time complexity and size of the network generated by Directional-Relational-PC along d is $O(\exp(W*(d)+1))$.

6 Conclusions

Constraint networks have been shown to have many applications. However, two common reasoning tasks: (i) find a solution that satisfies the constraints and (ii) find the corresponding minimal network are known to be NP-complete in the general case. In this paper, we have identified sufficient conditions based on the row-convexity of the constraints and the level of local consistency that guarantee that a solution can be found in a backtrack-free manner. For binary networks, we showed that we can efficiently test whether a network satisfies the conditions, and when it does, we gave efficient algorithms for solving both tasks (i) and (ii). We argued, by examining applications of constraint networks in the literature, that we have identified an interesting and useful special class of constraint networks.

Acknowledgements

The authors wish to thank the referees for their careful reading of the paper and their helpful comments.

References

[1] J. F. Allen. Maintaining knowledge about temporal intervals. *Comm. ACM*, 26:832–843, 1983.

[2] S. Arnborg, D. G. Corneil, and A. Proskurowski. Complexity of finding an embedding in k-trees. *SIAM Journal of Algebraic Discrete Methods*, 8:177–184, 1987.

[3] C. Beeri, R. Fagin, D. Maier, and M. Yannakakis. On the desirability of acyclic database schemes. *J. ACM*, 30:479–513, 1983.

[4] K. S. Booth and G. S. Lueker. Testing for the consecutive ones property, interval graphs, and graph planarity using pq-tree algorithms. *J. Comput. Syst. Sci.*, 13:335–379, 1976.

[5] M. B. Clowes. On seeing things. *Artificial Intelligence*, 2:79–116, 1971.

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

[7] R. Dechter, I. Meiri, and J. Pearl. Temporal constraint networks. *Artificial Intelligence*, 49:61–95, 1991.

[8] R. Dechter and J. Pearl. Network-based heuristics for constraint satisfaction problems. *Artificial Intelligence*, 34:1–38, 1988.

[9] R. Dechter and J. Pearl. Tree clustering for constraint networks. *Artificial Intelligence*, 38:353–366, 1989.

[10] R. Dechter and J. Pearl. Directed constraint networks: A relational framework for causal modeling. In *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence*, pages 1164–1170, Sydney, Australia, 1991.

[11] R. Dechter and I. Rish. Directional resolution: The Davis-Putnam procedure, revisited. In *Proceedings of the Fourth International Conference on Principles of Knowledge Representation and Reasoning*, Bonn, Germany, 1994.

[12] E. C. Freuder. Synthesizing constraint expressions. *Comm. ACM*, 21:958–966, 1978.

[13] E. C. Freuder. A sufficient condition for backtrack-free search. *J. ACM*, 29:24–32, 1982.

[14] E. C. Freuder. A sufficient condition for backtrack-bounded search. *J. ACM*, 32:755–761, 1985.

[15] C.-C. Han and C.-H. Lee. Comments on Mohr and Henderson's path consistency algorithm. *Artificial Intelligence*, 36:125–130, 1988.

[16] D. A. Huffman. Impossible objects as nonsense sentences. In B. Meltzer and D. Michie, editors, *Machine Intelligence 6*, pages 295–323. Edinburgh Univ. Press, 1971.

[17] L. M. Kirousis. Fast parallel constraint satisfaction. *Artificial Intelligence*, 64:147–160, 1993.

[18] L. M. Kirousis and C. H. Papadimitriou. The complexity of recognizing polyhedral scenes. In *Proceedings of the 26th Symposium on Foundations of Computer Science*, pages 175–185, Portland, Oregon, 1985.

[19] A. K. Mackworth. Consistency in networks of relations. *Artificial Intelligence*, 8:99–118, 1977.

[20] A. K. Mackworth. Constraint satisfaction. In S. C. Shapiro, editor, *Encyclopedia of Artificial Intelligence, 2nd Edition*, pages 285–293. John Wiley & Sons, 1992.

[21] A. K. Mackworth and E. C. Freuder. The complexity of some polynomial network consistency algorithms for constraint satisfaction problems. *Artificial Intelligence*, 25:65–74, 1985.

[22] H. Maruyama. Structural disambiguation with constraint propagation. In *Proceedings of the 28th Conference of the Association for Computational Linguistics*, pages 31–38, Pittsburgh, Pennsylvania, 1990.

[23] I. Meiri, R. Dechter, and J. Pearl. Tree decomposition with applications to constraint processing. In *Proceedings of the Eighth National Conference on Artificial Intelligence*, pages 10–16, Boston, Mass., 1990.

[24] U. Montanari. Networks of constraints: Fundamental properties and applications to picture processing. *Inform. Sci.*, 7:95–132, 1974.

[25] P. van Beek. Reasoning about qualitative temporal information. *Artificial Intelligence*, 58:297–326, 1992.

[26] P. Van Hentenryck, Y. Deville, and C.-M. Teng. A generic arc consistency algorithm and its specializations. *Artificial Intelligence*, 57:291–321, 1992.

[27] M. Vilain and H. Kautz. Constraint propagation algorithms for temporal reasoning. In *Proceedings of the Fifth National Conference on Artificial Intelligence*, pages 377–382, Philadelphia, Pa., 1986.

[28] D. Waltz. Understanding line drawings of scenes with shadows. In P. H. Winston, editor, *The Psychology of Computer Vision*, pages 19–91. McGraw-Hill, 1975.