On the Minimality and Decomposability of Constraint Networks

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

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 a network is both minimal and decomposable. Decomposable networks have the property that a solution can be found without backtracking. We show that the row convexity property can be tested for efficiently and we show, by examining applications of constraint networks discussed in the literature, that our results are useful in practice. Thus, we identify a large class of constraint networks for which we can solve both tasks (i) and (ii) efficiently.

1 Introduction

Constraint networks have been shown to be useful in formulating such diverse problems as graph coloring [Montanari, 1974], scene labeling [Huffman, 1971; Waltz, 1975], natural language parsing [Maruyama, 1990], and temporal reasoning [Allen, 1983; van Beek, 1990]. 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 [Meiri *et al.*, 1990] and temporal reasoning [van Beek, 1989]. 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 is often useful in practice, 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 a network is both minimal and decomposable. Decomposable networks have the property that a solution can be found without backtracking. In particular, we first show that, if all of the binary constraints are taken from a language that is closed under composition, intersection, and transposition and for which the binary constraints are all row convex, then we can guarantee *a priori* that the result of path consistency will be the minimal network and that the network will be decomposable. Second, we show that a large class of networks can be shown to have the row convexity property after path consistency. If such is the case, then the network is minimal and decomposable. Third, we 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 row convex, then a solution can be found without backtracking. We also show that the row convexity property can be tested for efficiently and we show, by examining applications of constraint networks discussed in the literature, that our results are useful in practice. Thus, we identify a large class of constraint networks for which we can solve both tasks (i) and (ii) efficiently.

2 Background

We begin with some needed definitions and describe related work.

A *network of binary constraints* [Montanari, 1974] is defined as a set X of n variables $\{x_1, x_2, ..., x_n\}$, a domain D_i of possible values for each variable, and 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 $(x_j, x_i) \in R_{ji} \Leftrightarrow (x_i, x_j) \in R_{ij}$.

An *instantiation* of the variables in X is an n-tuple $(X_1, X_2, ..., X_n)$, 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 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).

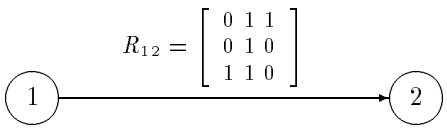
Mackworth [1977; 1987] defines three properties of networks that characterize local consistency of networks: *node*, *arc*, and *path consistency*. A network is *path consistent* if and only if, for every triple (x_i, x_k, x_j) of variables, we have that, for every instantiation of x_i and x_j that satisfies the direct relation, R_{ij}, there exists an instantiation of x_k such that R_{ik} and R_{kj} are also satisfied. Montanari [1974] and Mackworth [1977] provide algorithms for achieving path consistency. Freuder [1978] generalizes this to k-consistency. A network is *k-consistent* if and only if given any instantiation of any $k-1$ variables satisfying all the direct relations among those variables, there exists an instantiation of any kth variable such that the k values taken together satisfy all the relations among the k variables. Freuder [1982] defines *strongly k-consistent* as 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 *decomposable*. Decomposable 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 [Dechter, 1990]. 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 [1974], 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 the pair is permitted.

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. A binary relation R represented as a (0,1)-matrix is *monotone* if and only if the following conditions hold: if $R_{ij} = 1$ and $k \geq i$, then $R_{kj} = 1$, and if $R_{ij} = 1$ and $k \leq j$, then $R_{ik} = 1$. A binary relation R 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.

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 (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.



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 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 [1974] shows that if the constraint graph is a tree, path consistency is sufficient to ensure a network is minimal. Freuder [1982; 1985] identifies a relationship between a property called the *width* of a constraint graph and the level of local consistency needed to ensure a solution can be found without backtracking. As a special case, if the constraint graph is a tree, arc consistency is sufficient to ensure a solution can be found without backtracking. Dechter and Pearl [1988] provide an adaptive scheme where the level of local consistency is adjusted on a node-by-node basis. Freuder [1990] generalizes the results on trees to k-trees.

For work that falls into the class of restricting the type of the constraints (the class into which the present work falls), Dechter [1990] identifies a relationship between the size of the domains of the variables and the level of local consistency needed to ensure the network is strongly n-consistent, and thus minimal and decomposable. Montanari [1974] shows that path consistency is sufficient to guarantee a network is both minimal and decomposable if the relations are monotone, and Deville and Van Hentenryck [1991] show that arc consistency is sufficient to test the satisfiability of networks with only functional and monotone constraints. Functional and monotone relations are row convex; hence, it will be seen that our results generalize Montanari's and extend Deville and Van Hentenryck'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 [1991] 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 PreCondition

Informally, the basic result of this section is that 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 minimal network and the minimal network will be decomposable. 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.

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.

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 the row vectors in F have a non-zero entry in common.*

Theorem 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 R be a binary constraint network with all relations taken from $\mathcal{L}$. The path consistency algorithm will correctly determine the minimal network of R. Further, the minimal network will be decomposable.*

Proof. The theorem is proved by showing that if all (0,1)-matrices are from $\mathcal{L}$ and the network is made path consistent, then the network is k-consistent for all $k \leq n$. Hence, the network is strongly n-consistent and therefore the network is 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

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

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

$$X_i \; R_{ik} \; x_k \qquad i = 1, \ldots, k-1 \qquad (1)$$

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 Eqn. 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 this follows directly from the definition of path consistency. Hence, all the constraints have a non-zero entry in common and there exists at least one instantiation of x_k that satisfies Eqn. 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 of variable x_k such that

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

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

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

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

The proof of the Theorem 1 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(R)

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** $r \leftarrow r \cap (\text{row } X_j \text{ of } R_{ji})$
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 $O(n^3)$ [Mackworth and Freuder, 1985]. So, we can find a solution and the minimal network for the class of constraint networks characterized by Theorem 1 in $O(\max(dn^2, n^3))$ time.

Example 1. 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 theorem applies to all constraint networks with domains of size two. As a specific example, the Graph 2-coloring problem can be formulated using such constraint networks. Dechter [1990, p.233] also shows, but by a different method, that a strong 3-consistent (or path consistent) bi-valued network is minimal.

Example 2. 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 inequalities of the form $ax_i + bx_j < c$ or $ax_i + bx_j \leq c$, where a, b, and c are rational 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 viewed as an integer linear program where each constraint is in two variables and the domains of the variables are

[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 [Montanari, 1974; Mackworth, 1977] for the details.

restricted to be finite subsets of the integers. 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. Hence, by Theorem 1 we can guarantee that the result of path consistency will be the minimal network and the network will be decomposable. Two special cases are a restricted and discrete version of Dechter, Meiri, and Pearl's [1991] continuous, bounded difference framework for temporal reasoning and a restricted and discrete version of Vilain and Kautz's [1989] qualitative framework for temporal reasoning.

4 Sufficient PostConditions

Informally, the basic result of this section is that if we have applied path consistency to a network and the relations are row convex or can be made row convex, then the network is minimal and decomposable. We also show that a known procedure from graph theory can be used for deciding whether a constraint network can be made row convex.

Theorem 2 *Let R be a path consistent binary constraint network. If there exists an ordering of the domains $D_1, \ldots, D_n$ of R such that the (0,1)-matrices are row convex, the network is minimal and decomposable.*

Example 3. Scene labeling in computer vision [Huffman, 1971; Clowes, 1971] can be formulated as a problem on constraint networks. We use an example to illustrate the application of Theorem 2. Fig. 2 shows the variables in the constraint network and the constraints; Fig. 1 shows the domains of the variables and the ordering imposed. For example, variable x_1 in Fig. 2 is a fork and can be instantiated with any one of the five labelings shown in Fig. 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. Procedure Instantiate from the previous section can be used to find a solution. The four possible solutions are shown in Fig. 3.

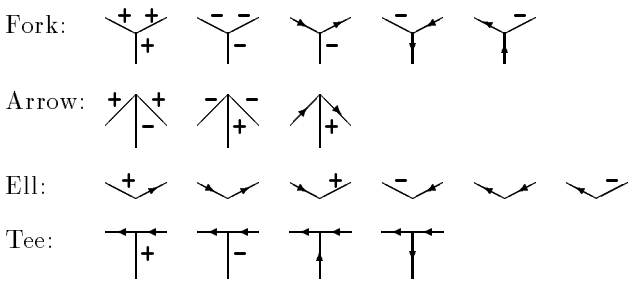
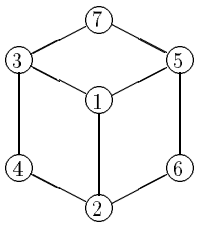


Figure 1: Huffman-Clowes junction labelings

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

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



$$R_{21} = \begin{bmatrix} 0&1&1&0&0 \\ 1&0&0&0&0 \\ 1&0&0&0&0 \end{bmatrix}$$

$$R_{31} = \begin{bmatrix} 0&1&0&1&0 \\ 1&0&0&0&0 \\ 1&0&0&0&0 \end{bmatrix}$$

$$R_{51} = \begin{bmatrix} 0&1&0&0&1 \\ 1&0&0&0&0 \\ 1&0&0&0&0 \end{bmatrix}$$

Figure 2: Scene labeling constraint network

The scene-labeling problem has been shown to be NP-complete in the general case [Kirousis and Papadimitriou, 1985]. We are attempting to prove the conjecture that constraint networks arising from orthohedral scenes are row convex once path consistency is applied.

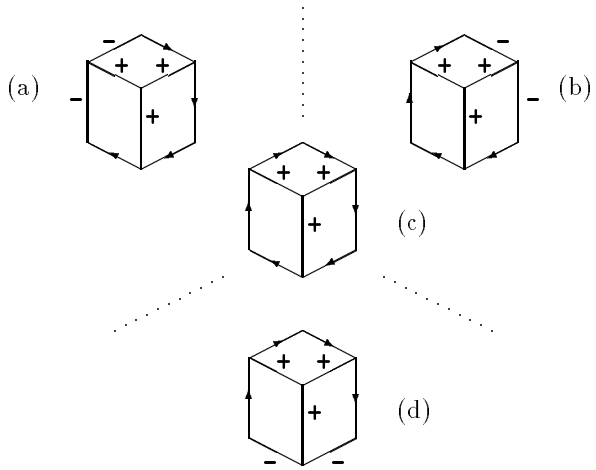


Figure 3: Solutions: (a) stuck on left wall, (b) stuck on right wall, (c) suspended in mid-air, (d) resting on floor.

As noted in the previous 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 3 (Booth and Lueker [1976]) *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 [1990] 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 [Maruyama, 1990] to illustrate the application of Theorems 2 and 3. Consider the following sentence.

Put the block on the floor on the table in the room.
V_1 NP_2 PP_3 PP_4 PP_5

The sentence is structurally ambiguous (there are fourteen different parses) as there are many ways to attach the prepositional phrases. Fig. 4 shows the variables and their domains; Fig. 5 shows the constraint network (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." The network is path consistent but it can be seen that the constraints are not all row convex given the original domain ordering used in [Maruyama, 1990]. However, using the new domain ordering shown in Fig. 4, the constraints are now row convex. Hence, procedure Instantiate from the previous section can be used to find a solution in a backtrack-free manner.

Variable	Domain	
	Original ordering	New ordering
V_1	{Rnil}	{Rnil}
NP_2	{O1}	{O1}
PP_3	{L1, P2}	{L1, P2}
PP_4	{L1, P2, P3}	{P2, P3, L1}
PP_5	{L1, P2, P3, P4}	{P3, P4, L1, P2}

Figure 4: Variables and domains for parsing example

Let R be a path consistent binary constraint network. It remains to show how Theorem 3 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 $\ldots R_{nj}$ and test whether the matrix can be made row convex. For example, with reference to Fig. 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.

	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: Matrix representation of constraint network for parsing example [Maruyama, 1990]

In this example such a permutation exists and corresponds to the new ordering of the domain of variable PP4 shown in Fig. 4.

It is, of course, not true that for every path consistent network there exists an ordering of the domains such that the constraints are row convex. However, in those cases where there does not, sometimes a weaker result still applies.

Theorem 4 *Let 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 R such that the (0,1)-matrices R_{ij}, $1 \leq i < j \leq n$, are row convex, then a consistent instantiation can be found without backtracking.*

Example 5. Consider the constraint network with three variables and domains $D_1 = D_2 = D_3 = \{a,b,c\}$ shown in Fig. 6. The example is path consistent and 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 4 and the variables can be instantiated in the order x_1, x_2, x_3 using procedure Instantiate, and it can be guaranteed that no backtracking is necessary (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 decomposability 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: Matrix representation of constraint network that can be made "sufficiently" row convex

5 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 a large, and we believe interesting and useful, class of constraint networks for which we can solve both tasks (i) and (ii) efficiently.

Acknowledgements. I would like to thank Rina Dechter for fruitful discussions on this topic. Financial assistance was received from the Central Research Fund of the University of Alberta and the Natural Sciences and Engineering Research Council of Canada.

References

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

[Booth and Lueker, 1976] 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.

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

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

[Dechter and Pearl, 1991] 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.

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

[Dechter, 1990] R. Dechter. From local to global consistency. In *Proceedings of the Eighth Canadian Conference on Artificial Intelligence*, pages 231–237, Ottawa, Ont., 1990.

[Deville and Van Hentenryck, 1991] Y. Deville and P. Van Hentenryck. An efficient arc consistency algorithm for a class of CSP problems. In *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence*, pages 325–330, Sydney, Australia, 1991.

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

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

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

[Freuder, 1990] E. C. Freuder. Complexity of k-tree structured constraint satisfaction problems. In *Proceedings of the Eighth National Conference on Artificial Intelligence*, pages 4–9, Boston, Mass., 1990.

[Huffman, 1971] 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.

[Kirousis and Papadimitriou, 1985] 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, October 1985.

[Mackworth and Freuder, 1985] 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.

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

[Mackworth, 1987] A. K. Mackworth. Constraint satisfaction. In S. C. Shapiro, editor, *Encyclopedia of Artificial Intelligence*. John Wiley & Sons, 1987.

[Maruyama, 1990] 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.

[Meiri *et al.*, 1990] 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.

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

[van Beek, 1989] P. van Beek. Approximation algorithms for temporal reasoning. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, pages 1291–1296, Detroit, Mich., 1989.

[van Beek, 1990] P. van Beek. Reasoning about qualitative temporal information. In *Proceedings of the Eighth National Conference on Artificial Intelligence*, pages 728–734, Boston, Mass., 1990.

[Vilain *et al.*, 1989] M. Vilain, H. Kautz, and P. van Beek. Constraint propagation algorithms for temporal reasoning: A revised report. In D. S. Weld and J. de Kleer, editors, *Readings in Qualitative Reasoning about Physical Systems*, pages 373–381. Morgan-Kaufman, 1989.

[Waltz, 1975] 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.