Artificial Intelligence 55 (1992) 87–107
Elsevier

From local to global consistency*

Rina Dechter**

Information and Computer Science, University of California, Irvine, CA 92717, USA

Received November 1990
Revised August 1991

Abstract

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

In reasoning tasks involving the maintenance of consistent databases (so-called QQconstraint networks/Q/Q), it is customary to enforce local consistency conditions in order to simplify the subsequent construction of a globally coherent model of the data. In this paper we present a relationship between the sizes of the variables' domains, the constraints' arity and the level of local consistency sufficient to ensure global consistency. Based on these parameters a new tractability classification of constraint networks is presented. We also show, based on this relationship, that any relation on bi-valued variables which is not representable by a network of binary constraints cannot be represented by networks with any number of hidden variables.

1. Introduction

One of the major forces shaping resource-bounded reasoning stems from the requirement of local computation, namely, from the need to consider only a few data items at any inference step and to avoid the decision where to store intermediate results. All realistic models of human reasoning invoke this notion of locality in one form or another. For example, spreading activation in conceptual memories is grounded in the notion that activity spreads locally among conceptually neighboring entities, but does not leap toward remotely designated addresses.

Correspondence to: R. Dechter, Information and Computer Science, University of California, Irvine, CA 92717, USA.

* Revised extension of the paper that won the Artificial Intelligence Journal Best Paper Award at CSCSI-90. This work was supported in part by the National Science Foundation, Grant #IRI-88215522 and by the Air Force Office of Scientific Research, Grant #AFOSR-90-0136 while the author was visiting the Cognitive Systems Lab at UCLA.

** The work was done while the author was at the Computer Science Department, Technion, Haifa, Israel.

In reasoning tasks involving the maintenance of a consistent set of information items (so-called "constraints"), the principle of locality has been embodied in techniques that enforce local consistency among groups of related variables. The rationale being that such local consistency will simplify the task of constructing a globally coherent model of the data. For example, the scene labeling scheme of Waltz [28] often leads to globally consistent objects, although each step examines only neighboring edges and vertices. Truth maintenance systems [9] often sacrifice completeness by limiting the inferential steps to those involving constraint propagation [20, 21]. Such local techniques share the computational advantages mentioned earlier: there are only a few data items participating in each inference step, these items bear meaningful conceptual relationships to one another, partial results are stored exactly where they will be useful, computational steps can be performed in any order, and there is no need to remember which part of the knowledge has been processed and which part has not.

For a collection of constraints to be *globally consistent* means that it is completely explicit, namely the set of "partial solutions" to each subset of the constraints can always be extended to a "full solution" that satisfies all the constraints. This property makes globally consistent "constraint networks" very attractive computationally. They admit greedy search algorithms which are guaranteed to generate a solution without any dead-ends (i.e., backtrack-free [12]). Our wish, therefore, is to enforce global consistency using as local a computation as possible.

So far all known conditions under which local consistency would entail global consistency involved topological properties of the network representing the interactions among the data items (so-called "constraint networks") [7, 12]. In this paper we consider additional parameters of the problem specification. We give a general relationship between the sizes of the variables' domains, their constraints' arity and the level of local consistency required for ensuring global consistency. Specifically, in any problem having constraints of arity r or less and domains of size k or less, if all subproblems whose sizes are bounded by $k(r-1)+1$ are consistent, then the whole problem is globally consistent. The main theorem and its corollaries are presented in Section 3. It results in a new classification scheme for constraint networks presented in Section 4. Section 5 modifies these results to directional consistency while Sections 6 and 7 provide examples and concluding remarks.

2. Definitions and preliminaries

A *constraint network*, R, (or CN for short) consists of a set of n *variables* $X_1, \ldots, X_n$, each associated with a finite *domain* of values, $D_1, \ldots, D_n$, and a set of *constraints* $\{C_1, \ldots, C_t\}$. A constraint C_i consists of two parts: (1) a

subset of variables, $var(C_i) = \{X_{i_1}, \ldots, X_{i_{j(i)}}\}$ called the *c-set*; and (2) a relation, $rel(C_i)$ defined on this subset:

$$rel(C_i) = rel(X_{i_1}, \ldots, X_{i_{j(i)}}) \subseteq D_{i_1} \times \cdots \times D_{i_{j(i)}} . \tag{1}$$

$rel(C_i)$ stands for all simultaneous value assignments to the variables of $var(C_i)$ that are restricted by C_i alone. The *scheme* of R is the set of all its c-sets, $\{var(C_1), \ldots, var(C_t)\}$. A *solution* to R is an assignment of a value to each variable such that all the constraints are satisfied. We say that the network, R, *represents* the relation, $rel(R)$, consisting of all its solutions. Formally,

$$rel(R) = \{(X_1 = x_1, \ldots, X_n = x_n) \mid \forall C_i, \Pi_{var(C_i)} rel(R) \subseteq (C_i)\} . \tag{2}$$

$\Pi_U \rho$ stands for the projection of a subset of variables $U = U_1, \ldots, U_l$ on the relation ρ, which is defined by:

$$\Pi_U(\rho) = \{\bar{x}_U = (x_{U_1}, \ldots, x_{U_l}) \mid \exists \bar{x} \in \rho, \bar{x} \text{ is an extension of } \bar{x}_U\} .$$

A relation ρ, satisfying $\rho = rel(R)$ is said to be *decomposable* by R and, alternatively, R is said to be a *network decomposition* of ρ.

Typical tasks defined in connection with networks of constraints are to determine whether or not a solution exists, to find one or all solutions, and to determine whether an instantiation of some subset of variables is a part of a global solution. These tasks are collectively called *constraint satisfaction problem* (CSPs).

A *binary constraint network* is one in which every c-set involves at most two variables. In this case the network can be associated with a constraint graph, where each node represents a variable and the arcs connect nodes whose variables are explicitly constrained. An r-ary CN involves constraints with arity r or less. Figure 1 shows the constraint graph of a binary constraint network where each node represents a variable having values $\{a, b, c\}$ and each link is associated with a strict lexicographic order (where $X_i < X_j$ iff $i < j$). (The domains and the constraints are shown explicitly on some of the links.)

Given a constraint network, R, any subset, X, of its variables determines a subnetwork R_X that contains all those constraints in R whose c-sets are contained in X. In the following paragraphs we define the notions of *local*, *relative* and *global consistency* which are central to this paper. Throughout the

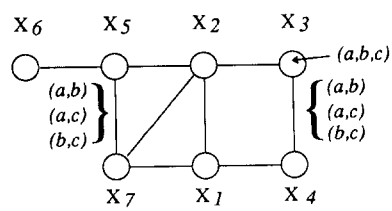

Fig. 1. An example of a binary constraint network.

paper, n stands for the number of variables, k for the number of values and r for the constraints' arity.

Definition 2.1. A partial instantiation of variables $(X_1 = x_1, \ldots, X_i = x_i)$ is *locally consistent* if it satisfies all the constraints in the subnetwork restricted to the set $\{X_1, \ldots, X_i\}$.

Definition 2.2. Given two subnetworks R_X and R_Y such that $X \subseteq Y$, subnetwork R_X is *consistent relative* to subnetwork R_Y if any partial instantiation of variables in R_X which is *locally consistent* can be extended to a consistent instantiation of R_Y.

Definition 2.3 (*Global consistency*).

(a) A *subnetwork* R_X is *globally consistent* if it is *consistent relative* to the overall network.

(b) A binary constraint network is said to be the *minimal network* [24] if *every* subnetwork of size two is globally consistent.

(c) A *globally consistent network* is one all of whose subnetworks are *globally consistent*.

Looking, for instance, at the network of Fig. 1, we see that the assignment $\alpha = (X_1 = a, X_2 = b, X_7 = c)$ is *locally consistent*. However, the network, $R_{\{X_1,X_2,X_7\}}$, is *not consistent relative* to $R_{\{X_1,X_2,X_5,X_7\}}$, since the consistent assignment α cannot be consistently extended to any value of X_5. Moreover, in this case most subnetworks are not *globally consistent* since the overall network is inconsistent, thus representing the empty relation.

We will now redefine the notion of *i-consistency* [12] in terms of *relative consistency*.

Definition 2.4.

(a) A network of constraints is said to be *i-consistent* if every subnetwork of size $i-1$ is *consistent relative to* any containing subnetwork of size i. (Equivalently, any locally consistent instantiations can be extended by any ith variable.) A network is *strong i-consistent* if it is j-consistent for every $j = 1, 2, \ldots, i$.

(b) Networks that are 2-consistent and 3-consistent are called *arc-consistent* and *path-consistent*, respectively [18, 24].

The notion of global consistency generalizes Montanari's notion of *decomposability* which was defined for the minimal network. Since a globally consistent network is completely explicit, its solutions can be generated in a backtrack-free manner in any chosen ordering [4]. Clearly, a network is i-consistent for all i (i.e., it is strong n-consistent) iff it is *globally consistent*.

When a network is not i-consistent, consistency-enforcing algorithms can be used to get it to the required consistency level [3, 10]. A brute-force algorithm for enforcing consistency of level i can be bounded by $O(n^{2i}k^{2i}i^r)$ and $\Omega((nk)^i i^r)$, as well as by $O((2nk)^{2i})$ and $\Omega((nk)^i)$ for unbounded r (see Appendix B for more details). Optimal algorithms for realizing the lower bound are available in [3]. For simplicity, we will use the brute-force upper bound complexity in our analysis.

3. Local consistency in constraint networks

3.1. Multi-valued r-ary networks

In this section we present a relationship between the maximum constraints' arity, r, the maximum domain's cardinality, k, and the level of local consistency that can ensure global consistency. For simplicity we assume that all variables have equal-sized domains. We refer to networks having at most r-ary constraints as r-ary constraint networks.

Theorem 3.1. *Any k-valued r-ary constraint network that is strong $(k(r-1)+1)$-consistent is globally consistent. In particular, any k-valued binary constraint network that is strong $(k+1)$-consistent is globally consistent.*

Proof. For simplicity we provide the proof for the special case, $r = 2$. This restricted case contains the main ideas and is much less cumbersome. The proof for the general case is given in Appendix A.

We will prove the theorem by showing that strong $(k+1)$-consistent binary networks are $(k+i+1)$-consistent for any $i \geq 1$. According to the definitions, we need to show that, if $\bar{x} = (x_1, x_2, \ldots, x_{k+i})$ is any locally consistent subtuple of the subset of variables $\{X_1, X_2, \ldots, X_{k+i}\}$, and if X_{k+i+1} is any additional variable, then there is an assignment x_{k+i+1} to X_{k+i+1} that is consistent with $\bar{x}$. We call an assignment to a single variable a *unary assignment* and we view $\bar{x}$ as a set of such unary assignments. With each value j, in the domain of X_{k+i+1} we associate a subset A_j that contains all unary assignments in $\bar{x}$ that are consistent with the assignment $X_{k+i+1} = j$. Since variable X_{k+i+1} may take on k possible values $\{1, 2, \ldots k\}$ this results in k such subsets, $A_1, \ldots, A_k$. We claim that there must be at least one set, say A_1, that contains the set $\bar{x}$. If this were not the case, each subset A_j would be missing some member, say x'_j, which means that the tuple generated by taking a missing unary assignment from each of the A_i's, i.e. $\bar{x}' = (x'_1, x'_2, \ldots, x'_k)$, whose length is k or less (there might be repetitions), could not possibly be consistent with any of X_{k+i+1}'s values. This leads to a contradiction because as a subset of $\bar{x}$, $\bar{x}'$ is locally consistent, and from the assumption of strong

$(k+1)$-consistency, this tuple should be extensible by any additional variable including X_{k+i+1}. Note that we need not assume that the $\{x'_j\}$'s are distinct unary assignments because *strong* $(k+1)$-consistency renders the argument applicable to subtuples $\bar{x}'$ of length less than k.

Assume now, without loss of generality, that A_1 spans all the $k+i$ unary assignments. This means that each assignment $X_j = x_j$, $j \in \{1, \ldots, k+i\}$, is consistent with the assignment $X_{i+k+1} = 1$, hence, we found a value (1 of X_{k+i+1}) that is consistent with $\bar{x}$. □

3.2. *Binary constraint networks*

Continuing our focus on binary constraints, we may be tempted to conclude from Theorem 3.1 that any k-valued binary network can be solved polynomially be enforcing strong $(k+1)$-consistency. However, enforcing $(k+1)$-consistency may require the addition of non-binary constraints, resulting in a $(k+1)$-consistent problem that is no longer binary. We can, however, conclude a weaker result.

Corollary 3.2. *A k-valued binary network, in which all induced constraints (generated by enforcing strong-$(k+1)$-consistency) are decomposable into binary constraints, can be solved in $O((nk)^{2(k+1)})$ steps.*

Proof. One can enforce strong $(k+1)$-consistency in $O((nk)^{2(k+1)})$ steps (see Appendix B). Then, each new recorded constraint is decomposed to its minimal network (by projecting the constraint on all subsets of two variables). Since we assumed that such a binary network represents each induced constraint precisely, the intersection of all the binary constraints results in an equivalent binary network which is strong $(k+1)$-consistent. Consequently, from Theorem 3.1, we can conclude that the resulting network is globally consistent. Since a globally consistent network can be solved in a backtrack-free manner, the cost of generating a solution is the cost of verifying that a partial instantiation of variables satisfies all the constraints. In binary networks this can be accomplished by $O(n^2)$ constraint checks. □

An interesting special case arises in the context of the k-colorability problem: given a set of k colors and a graph, assign a color to each node in the graph such that adjacent nodes will have different colors. This problem is clearly an instance of a k-valued binary constraint network. Moreover, since there are always enough colors (i.e., k) to consistently extend any locally consistent coloring of $k-1$ nodes (or less) by a color to one additional node, the problem is inherently *strong k-consistent*. Theorem 3.1 implies that if only we could extend the consistency level from k to $k+1$, the problem would become globally consistent and, hence, polynomially solvable. However,

Lemma 3.3 [16]. *A k-colorability problem is $(k+1)$-consistent iff each node has at most $k-1$ neighbors, namely the graph degree is bounded by k.*

Proof. Clearly, if each node's degree is less than k, the problem is $(k+1)$-consistent. Assume now that we have a $(k+1)$-consistent problem and a node X having k neighbors. Assigning a unique color to each neighbor results in a locally consistent assignment which is not extensible to X, thus yielding a contradiction. □

The colorability example strengthens the result of Theorem 3.1 since it provides an instance of a k-valued binary network that is strong k-consistent while *not* globally consistent (unless it has a degree bounded by k). We can conclude, therefore, that:

Theorem 3.4. *The level of strong local consistency required to ensure global consistency for k-valued binary networks is at least $k+1$.* □

3.3. Bi-valued binary networks

Another interesting special case occurs when the constraints are binary and all variables are bi-valued, that is, $k=2$. This is the only value of k for which the network remains binary after enforcing the required level of $(k+1)$-consistency. According to Theorem 3.1, bi-valued networks require strong 3-consistency (i.e., path-consistency) in order to be globally consistent. Since this level of consistency can be enforced by adding binary constraints only, we have the following corollary:

Corollary 3.5. *A strong 3-consistent bi-valued binary network is globally consistent and, in particular, minimal.*

Proof. Follows immediately from Theorem 3.1. □

We get that:

Theorem 3.6.

(1) *The minimal network of a bi-valued binary constraint network can be obtained in $O(n^3)$ steps.*
(2) *The consistency of a bi-valued binary constraint network can be determined in $O(n^3)$ steps.*

Proof. The minimal network is obtained by enforcing strong 3-consistency which takes $O(n^3)$ steps [19]. Since the minimal network is *globally consistent*, a solution can be generated without encountering dead-ends, therefore, requir-

ing only $O(n^2)$ additional steps. Note that finding a solution to a bi-valued binary network is equivalent to the 2-satisfiability problem [13] which is known to be linear. Thus part (2) of Theorem 3.6 does not provide the tightest bound for this task. However, we do not know of any better algorithm for the task of generating the minimal network. □

Theorem 3.6 has a surprising implication regarding the expressive power of hidden variables in bi-valued binary constraint networks. Given a bi-valued relation ρ, we can generate the minimal network of ρ by taking the projection of ρ on each pair of variables. The set of all solutions of this *minimal network* always contains ρ and it is the best binary network approximation of ρ [24]. When the minimal network represents ρ exactly, we say that ρ is *binary-network-decomposable*. In general, a relation ρ is not binary-network-decomposable, and the question is whether it can be decomposed using a larger network containing auxiliary bi-valued variables, or, equivalently, whether ρ can be a projection of some other relation *that is* binary-network-decomposable. In [5] we showed that if a bi-valued relation is not network-decomposable, adding any number of auxiliary bi-valued variables will not remedy the situation. We will provide here an alternative proof based on Theorem 3.1.

Definition 3.7. Let ρ be any n-ary bi-valued relation over a set of variables $X = \{X_1, \ldots, X_n\}$. We say that relation ρ is *h-network-decomposable* if there exist h additional bi-valued variables $Y = \{Y_1, \ldots, Y_h\}$ for which there is a binary network $R(X, Y)$ defined over $X \cup Y$ such that $\rho = \Pi_X\, rel(R(X, Y))$. The additional variables needed for decomposition will be called *hidden variables*.

Theorem 3.8. *A bi-valued relation that is not network-decomposable is also not h-network-decomposable, for any h.*

Proof. Suppose the contrary, that ρ is a bi-valued relation that is not network-decomposable over variables $X = \{X_1, \ldots, X_n\}$, and let $Y = \{Y_1, \ldots, Y_h\}$ be a set of hidden variables such that there is a relation ρ' over $X \cup Y$ satisfying $\rho = \Pi_X \rho'$ and ρ' is network-decomposable. Since ρ' is network-decomposable, its minimal network, M, must be a binary-network decomposition of ρ' and since it is minimal, it is also strong 3-consistent. Let M_X be the subnetwork restricted to the set X. According to Theorem 3.1, since M is a bi-valued strong 3-consistent binary network, any tuple which is locally consistent is also globally consistent. In particular, locally consistent solutions of M_X are globally consistent and, hence, $rel(M_X)$ is identical to the projection of $rel(M)$ on the set X, namely:

$$rel(M_X) = \Pi_X\, rel(M)\,, \tag{3}$$

or, substituting, $\rho' = rel(M)$,

$$rel(M_X) = \Pi_X \rho' . \tag{4}$$

The assumption $\rho = \Pi_X \rho'$ yields,

$$\rho = rel(M_X) , \tag{5}$$

which means that M_X is an exact network decomposition of ρ, thus contradicting our initial supposition. □

4. Classes of tractability

Theorem 3.1 supplies us with the ability to pose useful conjectures for certain families of constraint networks. When the maximum number of values is k, and the maximum constraint arity is r, we may try to prove that strong $(k(r-1)+1)$-consistency can be achieved using constraints with arity not exceeding r. If this is proved, Theorem 3.1 ensures that the resulting network is globally consistent, hence tractable.

The "level of local consistency" which is sufficient for global consistency can serve as a parameter for classifying problem instances into different classes of tractability. If we have a k-valued binary network, for instance, we can apply strong $(k+1)$-consistency to it, and if this is achievable using binary constraints only, we know that the problem is tractable and we say that it belongs to *class 1*. However, if the newly induced constraints are not "binary-decomposable", they may be of arity k, at worse, thus resulting in a k-valued k-ary constraint network. To make this new problem globally consistent, we are advised by Theorem 3.1 to enforce strong $(k(k-1)+1)$-consistency. If this is achievable with k-ary constraints only, we know the problem is globally consistent and we say it is in *class 2*. Else, the resulting problem may have $k(k-1)$-ary constraints and, accordingly, we need to enforce now strong $(k(k(k-1)-1)+1)$-consistency and so on. Continuing in this fashion, we can define higher classes of tractability, depending on the number of times we have to increase the consistency level until achieving global consistency.

In summary, we can divide k-valued binary constraint problems having n variables into approximately $\log_k n + 1$ complexity classes. Class 1 contains those problems that can be made $(k+1)$-consistent with only binary constraints, class 2 contains problems that can be made $(k(k-1)+1)$-consistent using constraints of arity k or less, and, in general:

Definition 4.1. Let K_i be defined by

$$\begin{aligned} &K_0 = r , \\ &\forall i > 0 , \quad K_i = k^i(r-1) - k^{i-1} - k^{i-2} - \cdots - k . \end{aligned} \tag{6}$$

A k-valued r-ary constraint network is in class i if it "can be made" strong $(K_i + 1)$-consistent via constraints of arity K_{i-1} or less. We say that a problem "can be made" strong consistent via constraints of size $K_i - 1$ if, when applying BFC(j) (see Appendix B) for increasing values of j, each new level-j constraint ($j = K_{i-1} + 1, \ldots, K_i + 1$) can be decomposed into constraints of size K_{i-1}.

Theorem 4.2. *Any constraint network in class i can be solved in* $O((2nk)^{2(K_i+1)})$ *steps.*

Proof. If a problem is in class i, we can make it globally consistent by enforcing strong $(K_i + 1)$-consistency which is bounded by $O((2nk)^{2(K_i+1)})$ steps (see Appendix B). Since the problem is in class i, we know that it can be made strong $(K_i + 1)$-consistent with constraints of arity K_{i-1} or less. From Theorem 3.1 it follows that, since the resulting problem (after enforcing $(K_i + 1)$-consistency) has k values and at most K_{i-1}-ary constraints, it needs be only $(k(K_{i-1} - 1) + 1)$-consistent to ensure global consistency. By substituting the expression for K_{i-1} (equation (6)), we get:

$$\begin{aligned} k(K_{i-1} - 1) + 1 &= k(k^{i-1}(r-1) - k^{i-2} - k^{i-3} - \cdots - k - 1) + 1 \\ &= k^i(r-1) - k^{i-1} - k^{i-2} - \cdots - k + 1 \\ &= K_i + 1 , \end{aligned} \tag{7}$$

which has already been enforced on the network. We can conclude, therefore, that the problem is globally consistent. As a result, for bounded k and r, problems in class i are tractable. □

We suspect that determining the lowest class to which a given problem belongs is NP-hard. However, the membership in a specific class can be decided polynomially.

Theorem 4.3. *Deciding whether a problem is in class i takes* $O((2nk)^{2(K_i+1)})$ *steps.*

Proof. First, we run the problem through a strong $(K_i + 1)$-consistency algorithm[1] which takes $O((2nk)^{2(K_i+1)})$. The resulting problem may have new constraints of arity between $K_{i-1} + 1$ and K_i. For each such constraint we need to determine whether it is expressible using constraint of arity not exceeding K_{i-1}. Let C be one such constraint. For each such constraint we generate a

[1] Note that Definition 4.1 is procedure-based in that it requires decomposability of the constraints recorded by the consistency enforcing algorithm, BFC(i). A more "declarative" definition would have rendered the membership problem intractable.

network, $R_C^{(i)}$ of K_{i-1}-ary constraints by projecting C on each subset of K_{i-1} variables. This operation is bounded by the cardinality of C, $O(k^{K_i})$. We then have to solve the resulting network $R_C^{(i)}$ in order to check if it has the same solution set as the original relation C. This corresponds to solving a constraint network problem having at most K_i variables and k values, which takes $O(k^{K_i})$ steps. The number of constraints, like C, that we may need to process is $O(n^{K_i})$ hence, the overall complexity is $O((2nk)^{2(K_i+1)})$. □

Clearly, class 1 is the most useful class; it is both most tractable and easiest to recognize. We will now define a subclass of class 1 which is particularly useful and which is frequently encountered. As already stated, a problem is in class 1 if it can be made strong $(k(r-1)+1)$-consistent with r-ary constraints. If this level of consistency is achieved by enforcing only $(r+1)$-consistency, we know that the problem is globally consistent, since $(r+1)$-consistency records at most r-ary constraints. Accordingly, we define:

Definition 4.4. A k-valued r-ary network that can be made strong $(k(r-1)+1)$-consistent by enforcing strong $(r+1)$-consistency is called *regular*.

Lemma 4.5. *The complexity of regular networks is* $O((2nk)^{2(r+1)})$.

Proof. Follows immediately from Theorem 3.1. □

Theorem 4.6. *Membership in the regular class can be determined in* $O((2nk)^{k(r-1)+2})$.

Proof. We first perform strong $(r+1)$-consistency which is bounded by $(2nk)^{2(r+1)}$. Then we have to check if the resulting network is $(r+2)$-consistent, $(r+3)$-consistent, and so on, until we reach $(k(r-1)+1)$-consistency. If all these levels of local consistency are verified, we know, based on Theorem 3.1, that the problem is globally consistent. The verification of all levels of local consistency between $r+2$ and $k(r-1)+1$ is bounded by

$$\sum_{i=r+2}^{k(r-1)+1} (2nk)^i = O((2nk)^{k(r-1)+2}) . \qquad \square \tag{8}$$

Thus, regular networks are more tractable (once recognized) and also are easier to recognize than general class-1 problems. Examples are given in Section 6.

In the next section we show that complexity can be further reduced by restricting processing to only *directional consistency*.

5. Directional consistency

The notion of directional consistency was introduced in [7] as a speed-up device which weakens local consistency while still ensuring global consistency along a given ordering. It allows all the constraints to be processed just once, resulting in a brute-force complexity bound of $O(n^{i+1}k^i2^i)$ (see Appendix B). This weaker consistency condition is easier to enforce than full i-consistency while ensuring backtrack-free search using the selected ordering. The directional bound is a square-root of the undirectional one.

Definition 5.1. A network of constraints is said to be directional i-consistent, with respect to an ordering $d = X_1, \ldots, X_n$, if every subnetwork of size $i-1$ which is locally consistent can be consistently extended by any variable that succeeds all the subnetwork's variables in the ordering d. It is *directional strong i-consistent* if it is directional j-consistent for $j \leq i$. A network of constraints is *directional globally consistent*, with respect to an ordering d, if it is directional i-consistent for every i.

Following is the directional version of Theorem 3.1.

Theorem 5.2. *Any k-valued r-ary constraint network that is directional strong $(k(r-1)+1)$-consistent with respect to d is also directional globally consistent.*

Proof. Same as the proof of Theorem 3.1. □

For completeness sake we present the algorithm *adaptive-consistency* that enforces *directional consistency* to any desirable level. The algorithm can be defined in terms of a procedure **adaptive**(*level*, d) [6] that enforces directional strong (*level* + 1)-consistency along an ordering d, where *level* is a parameter indicating the highest cardinality of constraints that are recorded, and d is an ordering $X_1, \ldots, X_n$, on the set of variables.

adaptive(*level*, $X_1, \ldots, X_n$)
Begin
1. for $i = n$ to 1 by -1 do
2. compute PARENTS(X_i) /*PARENTS(X_i) is the set of variables that are currently adjacent to and precede X_i in the constraint network.
3. perform **new-record**(*level*, X_i, PARENTS(X_i))
4. for *level* ≥ 2, connect all elements in PARENTS(X_i) (if they are not yet connected)

End

Procedure **new-record**(*level*, *var*, *set*) records only constraints of size less than or equal to *level* from the subset *set* and is defined as follows:

new-record(*level*, *var*, *set*)
Begin
1. if $level \leq |set|$ then
2. for every subset S in *set* such that $|S| = level$ do
3. **record-constraint**(*var*, S)
4. end
5. else, **record-constraint**(*var*, *set*)
End

The procedure **record-constraint**(*var*, *set*) generates and records (in the form of a new constraint on set) those locally consistent tuples of variables in *set* that are also consistent with at least one value of *var*. The new constraint is added to the original problem. Clearly, the algorithm enforces directional strong $(level + 1)$-consistency, and its complexity is bounded by both $O((nk)^{level+1}(level + 1)^r)$ and $O((2nk)^{level+1})$. The extension to *adaptive-consistency* is made by recording constraints on all the parent set (modifying line 3 of **adaptive**), and its complexity is bounded by both $O(k^{W^*+1}W^{*r})$ and $O((2k)^{W^*+1})$ where W^* is the maximum parent size resulting from applying the algorithm. See Appendix B and [8].

Definition 5.3. Given an ordering d, an r-ary k-valued constraint network is in *directional class i* if directional strong $(K_i + 1)$-consistency "can be enforced" using constraints of arity K_{i-1} or less.

As we saw, the cost of enforcing this level of directional consistency by adaptive $(K_i + 1, d)$ is $O((2nk)^{K_i+1})$ compared to $O((2nk)^{2(K_i+1)})$ in the undirectional version, namely the bound for the directional case is a square-root of the undirectional bound. We can also conclude that:

Theorem 5.4. *Given an ordering of the variables, d, the membership of a problem in directional class i can be determined in* $O((2nk)^{K_i+1})$ *steps.*

6. Examples

Example 6.1. Tasks of reasoning with time and space provide many examples of constraint networks having special local consistency properties. Allen's algebra [1], for example, defines thirteen possible relations between time intervals and provides a transitivity table for their propagation. This algebra can be described as a traditional constraint network where the variables are the relationships between two intervals, each having 13 values, and the transitivity

table defines ternary constraints on triplets of variables. Having $k = 13$ and $r = 3$, we conclude (Theorem 3.1) that *if* we can enforce strong 27-consistency without introducing higher than ternary constraints, we can generate a globally consistent network using a polynomial algorithm of degree 27. However, since Allen's algebra is known to be NP-complete, this is not a realistic assumption.

Example 6.2. A second example comes from temporal constraints limited to relationships between time points. Vilain and Kautz [27] suggested that if we consider three temporal relations between any two time points $\{<, =, >\}$ and define the transitive relationship induced by such relations, we get a simpler network that can be made globally consistent using a 3-consistent algorithm. This claim was later corrected by van Beek and Cohen [26], showing that 4-consistency is necessary for global consistency.

This $PA^{\neq}$ algebra (as termed by van Beek and Cohen [26]) can be described as a traditional constraint network, where the variables are the relationships between two points and the transitivity table defines ternary constraints. This yields a constraint network with $k = 3$, $r = 3$. Theorem 3.1 suggests that if such a network is strong 7-consistent, it is globally consistent. To show that the problem belongs to class 1, one has to show that it is feasible to enforce 7-consistency with ternary constraints. It can be shown that since 6-consistency for our formulation is equivalent to 4-consistency in the $PA^{\neq}$ formulation, and since the latter can be achieved using ternary constraints, so is 7-consistency. Thus implying global consistency, as also shown in [26]. It is still unclear whether this formulation of the point algebra (when the arcs are the variables) is a *regular* network.

Example 6.3. Consider a temporal constraint network, denoted INT^*, where the variables' domains are finite sets of integers and the constraints between pairs of variables are taken from the set $\{<, \leqslant, =, >, \geqslant\}$, with no inequalities. This is a binary constraint network. Thus, if $(k+1)$-consistency can be enforced with binary constraints we have a tractable problem (Theorem 3.1). We will show, however, that this is a *directional regular* problem, namely, given a certain ordering any level of directional local consistency is enforceable by directional 3-consistency. In fact, for these networks even directional 2-consistency suffices.

We associate a given problem with a directed acyclic graph as follows: the nodes denote variables and for each constraint of the form $X < Y$ or $X \leqslant Y$, we direct an arc from X to Y and label it with the constraint's inequality. Cycles in this graph can be easily detected (using a connected component algorithm). If a cycle contains a strict inequality, we can conclude that there is no solution. Otherwise, we can conclude that all variables on the cycle have the same value and they can be collapsed to one, thus resulting in an acyclic directed graph.

Theorem 6.4. *Networks in class INT*, having an acyclic graph representation, can be made directional globally consistent using directional 2-consistency.*

Proof. Given an acyclic graph representation of the problem, we can generate an ordering, d, of the nodes which is consistent with the partial order dictated by the graph. We will show that the problem is *directional regular* along the ordering d by showing that enforcing directional 2-consistency is sufficient to render it directional $(k+1)$-consistent and, therefore, directional globally consistent (Theorem 5.2). Suppose we want to enforce directional $(k+1)$-consistency using algorithm *adaptive-consistency* (with *level* $= k$). At each step we have a parent node P, a set of child nodes $C_1, \ldots, C_t$ and a set of t constraints. Each constraint is between a C_i and P, stating either $C_i < P$ or $C_i \leq P$. By enforcing directional 2-consistency on each C_i separately, we achieve any higher level of consistency, thus $(k+1)$-consistency. The reason is that any value remaining in C_i's domain after enforcing *directional 2-consistency* with P, has the property that it is less than or equal to some value p_i in P. Thus, any value assignment to all $C_1, \ldots, C_t$ from these restricted domains is extensible by the $\max_i\{p_i\}$ in P. □

Van Beek [25] and independently Meiri and Pearl [22] have treated the *full* point-algebra problem using similar techniques and showed the same complexity bounds. However, once integer domains are introduced the full algebra (containing the $\neq$ relation) becomes intractable.

Example 6.5. A different family of constraint networks arises in the domain of scene labeling. Huffman [15] and Clowes [2] developed a basic labeling scheme for blocks world picture graphs. Given a basic labeling set: + (convex), − (concave), → (occluding, object on arrowhead side with two possible senses), and a standard set of simplifying assumptions on scene content, the physically realizable junction labelings are just those shown in Fig. 2. Freuder [11] provided algorithms for labeling this restricted set while Waltz [28] explored a richer label set.

A network composed of the junctions in Fig. 2 can be viewed in two ways: Each line can be viewed as a variable having four values while the constraints are binary or ternary depending on whether two or three lines intersect. From this view, Theorem 3.1 states that if a given network is also strong 9-consistent, it is already globally consistent. The second view treats each junction as a variable, each variable has 3, 4 or 6 values (the number of possible labeling combinations of a junction) and the constraints are binary. Theorem 3.1 states that if the problem is 7-consistent, it is globally consistent. The second view, therefore, provides a weaker consistency demand for guaranteeing global consistency. Nevertheless, since this problem is known to be NP-complete [17],

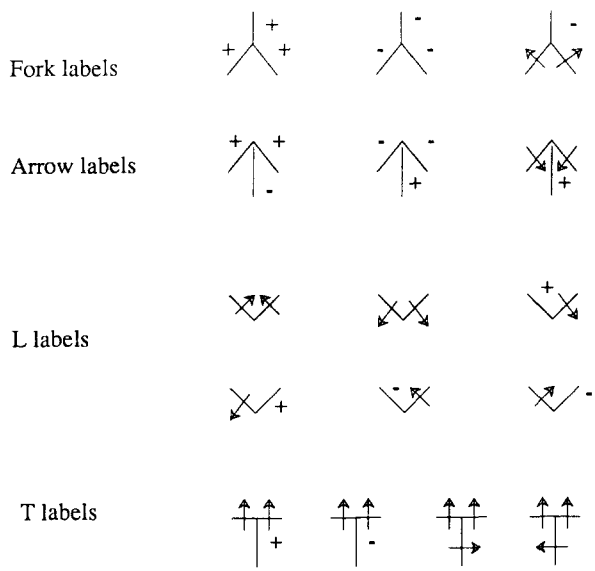

Fig. 2. Junction labels.

we know that there are problem instances where such a consistency level cannot be enforced without increasing the constraints arity.

We conclude with two additional examples of *regular* classes. One, mentioned earlier, is the class of bi-valued binary constraint networks. In this case the set of class-1 problems and regular problems coincide since $k(r-1)+1 = r+1$ when $k=2$ and $r=2$. A different class are those termed "distributive" by Montanari [24]. He showed that when the constraints satisfy the *distributivity* property, 3-consistency will make the network globally consistent.

7. Conclusions

A globally consistent network permits the construction of a consistent solution in linear time. We showed that the amount of local consistency required for achieving global consistency is dependent on the product of two parameters: the number of values in each variable and the constraint arity. The complexity of achieving the required local consistency is exponential in this product.

Based on this parameter we introduced a tractability classification of constraint networks, where each k-valued r-ary network falls into one of $\log_k n$ classes. Problems in class i can be solved in $O((2nk)^{2(k^i(r-1))})$ steps, and deciding whether a problem belongs to class i takes also $O((2nk)^{2(k^i(r-1))})$ steps. A special class of tractable problems, called *regular* was identified and special examples were presented.

As a consequence of Theorem 3.1, we showed that if a bi-valued relation is not representable by a binary constraint network, it cannot be helped by any number of hidden variables. This could be viewed as a generalization to Peirce's relation thesis. Peirce claimed that any relation could be decomposable into ternary relationships if the domain of the hidden variable is not bounded [14]. Here we showed that when one limits the domain into two values only, no expressiveness is gained. In another paper [5] we show that if the hidden variables have three or more values in their domains, then any relation is h-network-decomposable for some h.

Appendix A. Proof of Theorem 3.1

Theorem 3.1. *Any k-valued r-ary constraint network that is strong* $(k(r-1)+1)$*-consistent is globally consistent.*

Proof. We will show that strong $(k(r-1)+1)$-consistency implies that the network is also $(k(r-1)+i+1)$-consistent for any $i>0$. We need to show that, if $\bar{x}=(x_1, x_2, \ldots, x_{k(r-1)+i})$ is a locally consistent subtuple over variables $\{X_1, X_2, \ldots, X_{k(r-1)+i}\}$, and if $X_{k(r-1)+i+1}$ is any additional variable, then there is an assignment $x_{k(r-1)+i+1}$ to $X_{k(r-1)+i+1}$ that is consistent with $\bar{x}$. As for the binary network case, we will call a *unary assignment* any assignment to a single variable and we will view partial assignments both as sets (of unary assignments) or as sequences.

Since the network has constraints of arity r or less it means that a unary assignment

$$X_{k(r-1)+i+1} = x_{k(r-1)+i+1}$$

has to be independently consistent with each subset of $(r-1)$-ary assignments of the set $\bar{x}=\{x_1, x_2, \ldots, x_{k(r-1)+i}\}$. The compatibility of such a subset with a unary assignment to $X_{k(r-1)+i+1}$ is verified via the relevant constraints, having arity r or less. These are defined on the variable $X_{k(r-1)+i+1}$ and on a subset of the $r-1$ variables from $\{X_1, X_2, \ldots, X_{k(r-1)+1}\}$. In other words, all the constraints having arity r or less that involve variable $X_{k(r-1)+i+1}$ have to be verified in checking the consistency of any extension. Each such constraint can be uniquely identified by its c-set.

Variable $X_{k(r-1)+i+1}$ may take on k possible values $\{1, 2, \ldots, k\}$. Accordingly, we define k sets $A_1, \ldots, A_k$ as follows. A_j is the set of all size-$(r-1)$ subsets of unary assignments from $\bar{x}$ that are compatible with the assignment $X_{k(r-1)+i+1}=j$. Note that each subset must be locally consistent as it is a subset of $\bar{x}$, and since the network is strong $(k(r-1)+1)$-consistent, and, in particular, strong r-consistent, any such partial assignment must have at least one matching value in $X_{k(r-1)+i+1}$. The participation of an $(r-1)$-ary assignment,

t, in a subset A_j means that all the constraints involving its variables and variable $X_{k(r-1)+i+1}$ have to be satisfied. Let us denote by $S_j(t)$ the subset of constraints involved in the consistency verification of a tuple t with the value j of $X_{k(r-1)+i+1}$, and by $S(j)$ all the constraints that are relevant to all tuples in A_j. Let S denote all the constraints which are relevant to checking the consistency of $\bar{x}$ with a value of $X_{k(r-1)+i+1}$. Clearly,

$$S(j) = \bigcup_{t \in A_j} S_t(j) \quad \text{and} \quad S = \bigcup_j S(j).$$

We claim that there must be at least one set, say A_1, that requires *all the constraints* in S to be verified, namely that $S = S(1)$. Otherwise, each set, A_j, must have a constraint $C \in S$ such that $C \notin S(j)$. Let

$$var(C) = \{X_{1_j}, \ldots, X_{l_j}, X_{k(r-1)+i+1}\}, \quad l \leq r-1$$

be the constraint's c-set of that unverified constraint. This means that the partial assignment $\bar{x}_j = (X_{1j} = x_{1j}, \ldots, X_{lj} = x_{lj})$ is not in A_j. Therefore, the union of all these *excluded* $(r-1)$-ary assignments, $\bar{x}' = \bar{x}_1 \cup \bar{x}_2, \ldots, \cup \bar{x}_k$, is not consistent with any of $X_{k(r-1)+i+1}$'s values. However, the length of $\bar{x}'$ is less or equal to $k(r-1)$, and since we assumed strong $(k(r-1)+1)$-consistency, it must be consistent with at least one value, hence yielding a contradiction. Assume now, without loss of generality that $S = S(1)$. As a result, all partial assignments of $\bar{x}$ having size $r-1$ or less are consistent with the value "1" of $X_{k(r-1)+i+1}$ and we, consequently, found a value 1 of $X_{k(r-1)+i+1}$ which is consistent with $\bar{x}$. □

Appendix B. The complexity of brute-force consistency algorithm

Following is a worse-case analysis of a brute-force algorithm for enforcing i-consistency. Most of the analysis in the literature is for the special cases of $i = 2$ or $i = 3$ and for binary constraint networks [18, 23]. A detailed analysis of general k-consistency is given in [3]. Nevertheless, Cooper's analysis is quite involved and for the sake of this paper brute-force analysis will suffice. Since we do not restrict our treatment to binary networks we will express the complexity as a function of the constraints' arity as well, while assuming that the consistency level, i, is greater than the constraint's arity, r. Following is a description of the i-consistency algorithm, which is a generalization of Mackworth's PC-1 algorithms [18].

Brute-force-consistency BFC(i).

1. Begin
2. repeat until there is no change
3. for each subnetwork, R_{i-1}, of size $i-1$ on variables $X_1, \ldots, X_{i-1}$ do

```
4.          for each X ∈ R − R_{i−1} do
5.            record-constraint(R_{i−1}, X).
6.          end-for
7.        end-for
8.    end-repeat
9. end
```

Proposition B.1. *Algorithm BFC is bounded by* $O(n^{2i}k^{2i}i^r)$ *and by* $\Omega((nk)^i i^r)$ *steps, and, when r is unbounded it is bounded by* $O((2nk)^{2i})$ *and by* $\Omega((2nk)^i)$ *steps respectively.*

Proof. The algorithm has to process all subnetworks of size i (loop 3–7). The number of such networks is:

$$\binom{n}{i} = O(n^i) . \tag{B.1}$$

On each subnetwork the algorithm records constraints of arity $i-1$ that ensure its consistency relative to any one additional variable (inner loop, lines 4–6). Checking the consistency of one tuple of length $i-1$ against an ith variable may require checking constraints which are defined on every subset of the variables, namely, if the constraints' arity is bounded by r, the number of constraints of size r is bounded by

$$\binom{i}{r} = O(i^r) ,$$

and otherwise it can be bounded by 2^i.

Since there are at most k^i tuples whose consistency is verified, processing each size-i network is bounded by both

$$O(k^i i^r) \quad \text{and} \quad O((2k)^i) . \tag{B.2}$$

From (B.1) and (B.2) we get that one (3–7) loop is bounded by $O((nk)^i i^r)$ and $O((2nk)^i)$. The number of times these loops are executed (i.e., the number of cycles through loop 2–8) until convergence is bounded by the number of tuples of length i, namely by $O((nk)^i)$ (assuming that only one tuple is deleted for each loop). We get, therefore, overall bounds of $O((nk)^{2i}i^r)$ and $O(2nk)^{2i}$, for the cases that r is bounded and unbounded, respectively.

A lower bound for achieving i-consistency is derived by observing that just to verify that the network is i-consistent, requires checking all the constraints, a procedure that is equivalent to one (3–7) loop in the *BFC* algorithm, thus resulting in both $\Omega((nk)^i i^r)$ and $\Omega((2nk)^i$. □

Directional consistency algorithms [7] allow all subnetworks to be processed exactly once, hence we can show that:

Proposition B.2. *Directional i-consistency is both* $O(n^{i+1}k^i i^r)$ *and* $O(n^{i+1}k^i 2^i)$ *for bounded and unbounded r, respectively.*

Proof. The algorithm processes the constraints in a decreasing order of d, each time making the subnetwork restricted to variables $1, 2, \ldots, j-1$ directional i-consistent with respect to variable j. Since there are at most $O(n^i)$ subnetworks, and since each should be processed by $O(k^i i^r)$ steps, we get that making the network i-directional consistent with respect to one variable is $O((nk)^i i^r)$ and (if r is not bounded) $O((2nk)^i)$. □

Proposition B.3. *The complexity of the algorithm adaptive-consistency is* $O(k^{W^*+1}i^r)$ *and* $O((2k)^{W^*+1})$, *for bounded and unbounded r, respectively.*

Proof. We have to process at most n constraints, each of arity bounded by W^*, which yields the above bounds. For more details see [8]. □

Acknowledgement

I would like to thank Itay Meiri and Judea Pearl for their useful comments on different versions of this manuscript, and Caroline Ehrlich for proofreading it.

References

[1] J.F. Allen, Maintaining knowledge about temporal intervals, *Commun. ACM* **26** (11) (1983) 832–843.
[2] M.B. Clowes, On seeing things, *Artif. Intell.* **2** (1971) 79–116.
[3] M.C. Cooper, An optimal k-consistency algorithm, *Artif. Intell.* **41** (1) (1990) 89–95.
[4] R. Dechter, Studies in the use and generation of heuristics, Ph.D. Thesis, UCLA, Los Angeles, CA (1985).
[5] R. Dechter, On the expressiveness of networks with hidden variables, in: *Proceedings AAAI-90*, Boston, MA (1990).
[6] R. Dechter and I. Meiri, Experimental evaluation of preprocessing techniques in constraint satisfaction, in: *Proceedings IJCAI-89*, Detroit, MI (1989).
[7] R. Dechter and J. Pearl, Network-based heuristics for constraint-satisfaction problems, *Artif. Intell.* **34** (1) (1987) 1–38.
[8] R. Dechter and J. Pearl, Tree clustering for constraint networks, *Artif. Intell.* **38** (1989) 353–366.
[9] J. Doyle, A truth maintenance system, *Artif. Intell.* **12** (1979) 231–272.
[10] E.C. Freuder, Synthesizing constraint expression, *Commun. ACM* **21** (11) (1978) 958–965.
[11] E.C. Freuder, On the knowledge required to label a picture graph, *Artif. Intell.* **15** (1) (1980) 1–17.
[12] E.C. Freuder, A sufficient condition for backtrack-free search, *J. ACM* **29** (1) (1982) 24–32.
[13] M.R. Garey and D.S. Johnson, *Computer and Intractability, A guide to NP-Completeness* (Freeman, San Francisco, CA, 1979).

[14] H.G. Herzberger, Pierce's remarkable theorem, in: L.W. Sumner, J.G. Slater, F. Wilson, eds., *Pragmatism and Purpose: Essays Presented to Thomas A. Goudge* (University of Toronto Press, Toronto, Ont., 1981).
[15] D.A. Huffman, Impossible objects as nonsense sentences, in: B. Meltzer and D. Michie, eds., *Machine Intelligence* **6** (Edinburgh University Press, Edinburgh, Scotland, 1971) 195–234.
[16] S. Kasif, Private communication (October 1990).
[17] L.M. Kirousis and C.H. Papadimitriou, The complexity of recognizing polyhedral scenes, in: *Proceedings Symposium on Foundations of Computer Science*, Portland, OR (1985) 175–185.
[18] A.K. Mackworth, Consistency in networks of relations, *Artif. Intell.* **8** (1) (1977) 99–118.
[19] A.K. Mackworth and E.C. Freuder, The complexity of some polynomial network consistency algorithms for constraint satisfaction problems, *Artif. Intell.* **25** (1) (1985) 65–74.
[20] D.A. McAllester, An outlook on truth-maintenance, Tech. Rept., AI Memo No. 551, MIT, Boston, MA (1980).
[21] D.A. McAllester, Truth maintenance, in: *Proceedings AAAI-90*, Boston, MA (1990) 1109–1115.
[22] I. Meiri and J. Pearl, Faster constraint satisfaction algorithms for temporal reasoning, Tech. Rept. R-151, Cognitive Systems Lab, UCLA, Los Angeles, CA (1990).
[23] R. Mohr and T.C. Henderson, Arc and path consistency revisited, *Artif. Intell.* **28** (2) (1986) 225–233.
[24] U. Montanari, Networks of constraints: fundamental properties and applications to picture processing, *Inf. Sci.* **7** (1974) 95–132.
[25] P. van Beek, Reasoning about qualitative temporal information, in: *Proceedings AAAI-90*, Boston, MA (1990) 728–734.
[26] P. van Beek and R. Cohen, Approximation algorithms for temporal reasoning, in: *Proceedings IJCAI-89*, Detroit, MI (1989).
[27] M. Vilain and H. Kautz, Constraint propagation algorithms for temporal reasoning, in: *Proceedings AAAI-86*, Philadelphia, PA (1986) 377–382.
[28] D. Waltz, Understanding line drawings of scenes with shadows, in: P.H. Winston, ed., *The Psychology of Computer Vision* (McGraw-Hill, New York, 1975).