Generalized Arc Consistency with Application to MaxCSP and SCSP Instances

Michael C. Horsch[1], William S. Havens[2], and Aditya K. Ghose[3]

[1] Department of Computer Science,
University of Saskatchewan,
Saskatoon, SK, Canada S7N 5A9
horsch@cs.usask.ca
[2] Intelligent Systems Laboratory,
School of Computing Science,
Simon Fraser University,
Burnaby, B.C., Canada V5A 1S6
havens@cs.sfu.ca
[3] Dept. of Information Systems
University of Wollongong
Wollongong NSW 2522 Australia
aditya@uow.edu.au

Abstract. We present an abstract generalization of arc consistency which subsumes the definition of arc consistency in classical CSPs. We show that this generalization leads to useful application in classical CSPs as well ans non-classical CSPs such as MaxCSP, and instances of the Semi-ring CSP formalism recently developed by Bistarelli et al. [2]. We demonstrate the development of a selection of the possible applications, and show empirically that these can be of significant value in solving CSPs.

In classical constraint satisfaction problems (CSPs), the purpose is to find an assignment of values to variables such that all constraints on these variables are satisfied. There may be many such satisficing assignments, just one, or none at all. In such problems, there may not be any reason to prefer one solution over another. When such preferences do exist in an application domain, the problem becomes an optimization problem, in which the object is to find an assignment that maximizes the preference measure. For example, if a classical CSP is over-constrained, (i.e., no satisficing solution exists), it may be desirable to maximize the number of satisfied constraints. Other "non-classical" constraint problems include weighted CSPs, probabilistic CSPs, fuzzy CSPs, [2] etc., in which the objective is to find an assignment that optimizes the total weight, probability, rough membership, or other global property.

Arc consistency algorithms, such as the AC-3 [7], are local propagation methods used during search. However, recent results show that it is analogous to computing solution probabilities in classical CSPs [5]. An alternative view of arc consistency is that it approximates satisfiability, and expresses the approximation in terms of the domains of single variables.

Based on this view, a generalization of the arc consistency algorithm has been developed, and applied to classical and non-classical CSPs. We call the approach "xAC."The features of the approach are as follows:

- Local information is used to approximate global properties.
- Inference is based on operations defined by the optimization problem.
- Exact results are attainable in special cases, where sub-problem independence holds.
- In applications that violate the assumption of sub-problem independence, the local operations can be applied iteratively, to generate approximate solutions.

This approach can be used to derive methods for approximating some kinds of global optimization.

This approach has instances that include arc consistency algorithms [7], and probabilistic arc consistency [5], both of which pre-date the xAC theory. The approach was used to develop an algorithm, called MaxAC, for use in solving the MaxCSP problem. It can also be used to derive local propagation methods that can be used in solving other non-classical CSPs.

In the next sections, we present the approach in detail, starting from a review of the semiring CSP (SCSP) framework of Bistarelli et al. [2]. The presentation of xAC is aided by the notation of SCSPs, but the central results do not depend on the specific properties of SCSPs. Section 3 shows how arc consistency and pAC are instances of the approach. In Section 4 we show how the technique can be used to derive algorithms for MaxCSP. which we evaluate empirically. In Section 5 we summarize the approach and point to future work.

1 Semiring CSPs

In this section, we review the *semi-ring constraint satisfaction problem* (SCSP) notation of [2] as it provides a common language for our work. The SCSP framework provides a common language for many kinds of soft constraint problems, including classical, fuzzy, valued.

A *semiring* is a tuple $\langle A, \times, +, 0, 1\rangle$, in which A is a set, and $0, 1 \in A$. The operator $\times$ is a closed associative operator over pairs of elements of A, with 1 as its unit element, and 0 as its absorbing element. The operator $+$ is closed, commutative and associative, with unit element 0. Note that $0, 1, +, \times$ do not necessarily denote the usual meanings in arithmetic.

A *c-semiring* is a semiring in which $\times$ is commutative, $+$ is idempotent (i.e., $a+a = a$), and 1 is the absorbing element of $+$.

In the SCSP framework, we have a finite set of variables V, each taking values from a common domain D. Classically, a constraint is defined as a relation on tuples of values from D. Equivalently, such a relation can be expressed as a function on tuples, that returns $\top$ (i.e., true) if the tuple is in the relation, and $\bot$ (i.e., false) if it is not. The SCSP framework generalizes this idea by defining a constraint as a pair $\langle def, con\rangle$, where $con \subseteq V$ is the *type* of the constraint, and $def : D^{|con|} \rightarrow A$ is the constraint function, which indicates the degree to which a tuple of the appropriate type satisfies the constraint. As a departure from the notational conventions of [2], we will sometimes write $c.def$ and $c.con$ to refer to the components of a constraint c.

The SCSP framework defines *projection* of tuples, as follows. Let con be a set of variables, and $t \in D^{|con|}$. Let $con' \subset con$, and let $t' \in D^{|con'|}$. The projection of t onto to con' is the set of values in t for the variables in con', and is written $t' = t\downarrow^{con}_{con'}$.

The SCSP framework allows the combination of of two constraints, as follows. If c_1 and c_2 are two constraints, then define $c = c_1 \otimes c_2$ where $c.con = c_1.con \cup c_2.con$, and

$$c.def(t) = c_1.def(t\downarrow^{c.con}_{c_1.con}) \times c_2.def(t\downarrow^{c.con}_{c_2.con})$$

Note that here $\times$ is the multiplicative operator for the semiring.

The SCSP framework defines the projection of a constraint onto a set of variables. The *projection of constraint c onto $I \subseteq V$* is a constraint written $c' = c \Downarrow_I$ where $c'.con = c.con \cap I$ and

$$c'.def(t') = \sum_{t \models t'} c.def(t)$$

Note that the operator $\sum$ is the prefix expression for the additive semiring operator $+$. Also, here we have taken another slight departure from the notation of [2]: for two tuples $t \in D^{|con|}, t' \in D^{|con'|}$, $t \models t'$ whenever $t\downarrow^{con}_{con'} = t'$. Thus the sum is over all the tuples t that project into t'.

The solution of a SCSP is also a generalization of a classical CSP concept. Consider the problem of finding all tuples of values satisfying all the constraints in a classical CSP. This is equivalent to finding the *relational join* of all the classical constraints. In the SCSP framework, a SCSP *problem* is a tuple $\langle C, con\rangle$ where C is a set of constraints, and con is a set of variables of interest. The solution is a constraint defined by $Sol(P) = \left(\bigotimes_{c \in C} c\right) \Downarrow_{con}$. In this paper, the variables of interest will be the union of the types of all constraints in C.

In the next section, we use this notation to derive a notion of local consistency which generalizes arc consistency.

2 xAC: a framework for generalized arc consistency

The results in this section are presented in the context of the Semiring CSP framework as presented above. It is convenient to do so, because they have developed a framework for discussion generalized CSPs. Our results do not depend on many of the specific assumptions of this framework, however.

For classical CSPs, even though complete solutions could be found by computing the relational join of all the constraints, i.e., $\left(\bigotimes_{c \in C} c\right)$, search was the technique of choice for reasons of space efficiency, and the possibility of using heuristics to avoid many useless inferences.

Arc consistency algorithms can be seen as a satisfiability algorithm for constraint satisfaction: in the special case of tree-structured CSPs (that is, when there is exactly one path between any two nodes in the corresponding constraint graph), the values eliminated by AC algorithms do not appear in any solution of the CSP, and also, each remaining domain value appears in some solution of the CSP. Furthermore, the global property of satisfiability is represented in terms of the domains of each variable. In CSPs with more general constraint graphs, arc consistency algorithms can still be applied; however, in the general case, they eliminate domain values soundly, but not completely.

Definition 1. *Let $P = \langle C, con\rangle$ be a semiring constraint problem with constraints C and variables con. The* marginal solution *of P on variables $I \subseteq con$, written $M_I(P)$ is defined as follows:*

$$M_I(P) = \left(\bigotimes_{c\in C} c\right)\Downarrow_I$$

When I contains a single variable X, we write $M_X = \left(\bigotimes_{c\in C} c\right)\Downarrow_{\{X\}}$.

The idea is to project the solution of the SCSP onto a single variable. The marginal solution is what arc consistency algorithms compute when the CSP is tree structured, and it is what they approximate when the CSP is not tree structured.

The term "marginal" is borrowed from the probability calculus. Since $\otimes$ and $\Downarrow$ are defined in terms of $\times$ and $+$, a marginal solution $M_I(P)$ is strongly related to the computation of marginal probabilities in graphical probability models such as Bayesian networks [9]. We will discuss this relationship further in Section 3.

This definition is not an efficient means for computing the marginal solution.

Theorem 1. *Let P_X be the set of constraints whose type includes variable X, i.e., $P_X = \{c \in C | X \in c.con\}$. If there exist subproblems $P_1, \ldots, P_m$ such that for every pair of constraints $c_i \in C_i$ and $c_j \in C_j$, $c_i.con \cap c_j.con = \emptyset, i \neq j$, and $con_X \cap con_i = N_i$, where N_i is a set of neighbours of X in P, then*

$$M_X(P) = \bigotimes_{i=1}^{m} \left\{ \left[C_i \otimes M_{N_i}(P_i) \right] \Downarrow_X \right\}$$

□

The theorem says that if the problem is separable at X, then we can compute the marginal on X if we know the marginal on X's neighbours. There is nothing surprising about this result; however, it provides an alternative means of computing local consistency in SCSPs. The constraint propagation techniques for SCSPs [2, 1] are not based on marginal solutions, but rather, on modifying the constraints themselves.

The proof of the theorem is straightforward. It relies on the commutativity and associativity of the semiring operations $+$ and $\times$, but no other properties. Thus, the result is expressed here in terms of SCSPs, but does not rely on any property specific to SCSPs. In particular, the idempotence of $+$ and the absorbing element 1 of $+$ are irrelevant.

The theorem has a number of consequences. If the theorem holds for every variable in a SCSP, then the marginal solutions can be assessed by purely local computations, and also in polynomial time, as in the following theorem. For simplicity, we present the theorem for binary CSPs. It can be extended to constraints of arbitrary arity.

Theorem 2. *For a binary SCSP constraint problem defined over the c-semiring $\langle A, \times, +, 0, 1\rangle$ with a tree-structured constraint graph, for any variable X having constraints $C_{XY_i} = \langle def_i, \{X, Y_i\}\rangle, i = 1, \ldots, m$ (the Y_i are the neighbours of X in the constraint graph), define the following:*

$$C_{XY_i} = \langle def, con\rangle \in C$$

$$
\begin{aligned}
\text{where} \quad & C_{XY_i}.con = \{X, Y_i\} \\
& C_{XY_i}.def : D_X \times D_{Y_i} \to A \\
\mathrm{S}^{(k)}_{XY_i} &= \left(C_{XY_i} \otimes \mathrm{P}^{(k)}_{XY_i}\right) \Downarrow_X \\
\text{where} \quad & \mathrm{S}^{(k)}_{XY_i}.con = \{X\} \\
& \mathrm{S}^{(k)}_{XY_i}.def(x) = \sum_{y \in D_{Y_i}} \left(C_{XY_i}.def(x,y) \times \mathrm{P}^{(k)}_{XY_i}.def(y)\right) \\
\mathrm{M}^{(k)}_X &= \bigotimes_i \mathrm{S}^{(k)}_{Y_i X} \\
\text{where} \quad & \mathrm{M}^{(k)}_X.con = \{X\} \\
& \mathrm{M}^{(k)}_X.def(x) = \prod_{i=1}^{m} \mathrm{S}^{(k)}_{XY_i}.def(x) \\
\mathrm{P}^{(k+1)}_{Y_i X} &= \bigotimes_{j \neq i} \mathrm{S}^{(k)}_{XY_i} \\
\text{where} \quad & \mathrm{P}^{(k+1)}_{Y_i X}.con = \{X\} \\
& \mathrm{P}^{(k+1)}_{Y_i X}.def(x) = \prod_{j \neq i} \mathrm{S}^{(k)}_{XY_j}.def(x) \\
\mathrm{P}^{(0)}_{Y_i X} & \\
\text{such that} \quad & \mathrm{P}^{(0)}_{Y_i X}.con = \{X\} \\
& \mathrm{P}^{(0)}_{Y_i X}.def(x) = 1
\end{aligned}
$$

If the constraint graph has diameter d, then

$$M_X(P) = \mathrm{M}^{(d)}_X$$

□

The theorem defines the xAC equations, and establishes that they compute marginal solutions in certain cases. The notation $\mathrm{S}^{(k)}_{XY}$ can be interpreted as the support of X from Y in the kth iteration. Likewise, $\mathrm{P}^{(k)}_{XY}$ is the message propagated to X from Y on the kth iteration, and $\mathrm{M}^{(k)}_X$ is the kth approximation of the marginal solution.

The proof is not difficult, and it is only a very slight generalization of well-known results in constraint reasoning and probabilistic reasoning [6, 3, 5]. The proof does not depend on the idempotence of $+$ or the fact that 1 is an absorbing element of $+$.

While the requirement that the CSP be tree-structured seems very strong, we point out that several instances of this approach can be successfully applied to CSPs which are not tree-structured; AC-3 [7] is one instance, and pAC [5] is another. We discuss these applications in the next section.

Ultimately, the usefulness of this approach is to allow the derivation of local consistency algorithms for any problems which have a join operation and a projection op-

eration. The xAC approach provides the necessary theory, and instances of the theory can be created as the need arises. By generalizing arc consistency, we have provided solutions for MaxCSP (or *partial constraint satisfaction*), fuzzy CSPs, etc.

3 Current applications of xAC

In this section, we discuss two applications of Theorem 1 that predate the theory. Both of these applications target classical constraint problems, with a set V of variables, and a set C of constraints. Each variable $X \in V$ has a domain D_X, and we will assume that constraints are binary and represented as relation $C_{XY} \subset D_X \times D_Y$ (here, $\times$ is used to denote cross product of sets).

3.1 Arc consistency

For classical CSPs, the technique of arc consistency is intended to remove elements from the domain of variables if these elements, on the basis of purely local information, are known not to occur in a solution [7]. We will assume for simplicity that all constraints are binary. The results apply as well to the more general case.

The SCSP instantiation for classical binary CSPs is $\langle\{\top, \bot\}, \wedge, \vee, \bot, \top\rangle$ [2], i.e., the language of boolean arithmetic. Applying Theorem 2 to this structure, we get the following.

$$
\begin{aligned}
C_{XY_i}.def \;&:\; D_X \times D_{Y_i} \to \{\top, \bot\} \\
\mathrm{S}^{(k)}_{XY_i}.def(x) &= \bigvee_{y \in D_{Y_i}} \left(C_{XY_i}.def(x,y) \wedge \mathrm{P}^{(k)}_{XY_i}.def(y) \right) \\
\mathrm{M}^{(k)}_{X}.def(x) &= \bigwedge_{i=1}^{m} \mathrm{S}^{(k)}_{XY_i}.def(x) \\
\mathrm{P}^{(k+1)}_{Y_iX}.def(x) &= \bigwedge_{j \neq i} \mathrm{S}^{(k)}_{XY_j}.def(x) \\
\mathrm{P}^{(0)}_{Y_iX}.def(x) &= \top
\end{aligned}
$$

Note the definition for $\mathrm{S}^{(k)}_{XY}.def(x)$, which encodes the definition of arc consistency for the single arc (X, Y). The equation for $\mathrm{M}^{(k)}_{X}.def(x)$ determines if a value $x \in D_X$ should be eliminated from the domain of X: if no neighbour Y supports the value, then the conjunction is false, and the kth projection onto X is $\bot$.

3.2 Solution counting

This is the problem of counting the number of solutions to a classical CSP. An efficient method is known for tree-structured CSPs [3]. This method is equivalent to the xAC

equations on the semiring structure $\langle \{0, \ldots, N\}, \times, +, 0, N \rangle$, where $N = |D^n|$, and n is the number of constraints in the problem. Theorem 2 can be applied, resulting in:

$$
\begin{aligned}
C_{XY_i}.def &: D_X \times D_{Y_i} \to \{0, \ldots, N\} \\
\mathrm{S}^{(k)}_{XY_i}.def(x) &= \sum_{y \in D_{Y_i}} \left(C_{XY_i}.def(x,y) \times \mathrm{P}^{(k)}_{XY_i}.def(y) \right) \\
\mathrm{M}^{(k)}_{X}.def(x) &= \prod_{i=1}^{m} \mathrm{S}^{(k)}_{XY_i}.def(x) \\
\mathrm{P}^{(k+1)}_{Y_iX}.def(x) &= \prod_{j \neq i} \mathrm{S}^{(k)}_{XY_j}.def(x) \\
\mathrm{P}^{(0)}_{Y_iX}.def(x) &= 1
\end{aligned}
$$

3.3 Probabilistic arc consistency

The material presented in this section is a brief review of one of the results in [5].

Probabilistic arc consistency (pAC) is a technique used to approximate solution probabilities for a classical CSP. A solution probability for a variable is a frequency distribution over its values, representing the proportion of the number of solutions that make use of that value. The use of this approximation as a dynamic variable ordering heuristic has been found to reduce backtracking in random CSPs by as many as two orders of magnitude [5].

The equations for pAC can be derived from the xAC equations for solution counting with the addition of a normalization constant in the definition of $\mathrm{M}^{(k)}_{X}.def$. In the pAC equations that follow, $\times, +$ are multiplication and addition defined on reals.

$$
\begin{aligned}
C_{XY_i}.def &: D_X \times D_{Y_i} \to [0,1] \\
\mathrm{S}^{(k)}_{XY_i}.def(x) &= \sum_{y \in D_{Y_i}} \left(C_{XY_i}.def(x,y) \times \mathrm{P}^{(k)}_{XY_i}.def(y) \right) \\
\mathrm{M}^{(k)}_{X}.def(x) &= \alpha \prod_{i=1}^{m} \mathrm{S}^{(k)}_{XY_i}.def(x) \\
\mathrm{P}^{(k+1)}_{Y_iX}.def(x) &= \prod_{j \neq i} \mathrm{S}^{(k)}_{XY_j}.def(x) \\
\mathrm{P}^{(0)}_{Y_iX}.def(x) &= 1
\end{aligned}
$$

The quantity α in the definition of $\mathrm{M}^{(k)}_{X}.def$ is a normalization constant that ensures that each marginal distribution sums to 1. It does not come from Theorem2, but is due to the assumption that the marginal can be determined by considering the constraints independently. This assumption is called “causal independence” when made in Bayesian networks, for example.

A Bayesian network is a DAG in which the nodes are "random variables" and the arcs represent probabilistic dependence. Each node X in the graph has parents $pa(X)$, except for "root" nodes, for whom $pa(X) = \emptyset$. The joint probability distribution encoded by the Bayesian network with a set of variables $U = \{X_1, \ldots, X_n\}$ is assumed to have a factorization in terms of prior and conditional probabilities. These distributions can be assessed by experts, or learned from data. Each variable X_i has a set of possible values D_i; the probability of a tuple $t = (x_1, \ldots, x_n) \in D_1 \times \ldots \times D_n$ is given by

$$\mathrm{P}(t) = \prod_{i=1}^{n} \mathrm{P}(t{\downarrow}^{U}_{X_i} \mid t{\downarrow}^{U}_{pa(X_i)})$$

For a variable X_j, the marginal probability of $x_j \in D_j$ is

$$\begin{aligned} \mathrm{P}(x_j) &= \sum_{t \models x_j} \mathrm{P}(t) \\ &= \sum_{t \models x_j} \mathrm{P}(x_j | t{\downarrow}^{U}_{pa(X_j)}) \mathrm{P}(t{\downarrow}^{U}_{pa(X_j)}) \end{aligned}$$

Here, the summation is over all tuples that contain x_j. Note that $\mathrm{P}(x_j | t \downarrow^{U}_{pa(X_j)})$ is given information, and that $\mathrm{P}(t{\downarrow}^{U}_{pa(X_j)})$ is a marginal probability on a tuple from the parents of X_j. The reader is invited to compare this equation with Theorem 1.

If we assume that the parents of X_j are conditionally independent, then we can write

$$\mathrm{P}(t{\downarrow}^{U}_{pa(X_j)}) = \prod_{X \in pa(X_j)} \mathrm{P}(t{\downarrow}^{U}_{X})$$

This last assumption is equivalent to the assumption that the Bayesian network is tree structured (or *singly-connected*).

Finally, we consider the very special case that the conditional probability distribution of a variable X_j given its parents, can be represented as a product of simpler distributions:

$$\mathrm{P}(x_j | t{\downarrow}^{U}_{pa(X_j)}) = \alpha \prod_{X \in pa(X_j)} \mathrm{P}(X_j | t{\downarrow}^{U}_{X})$$

The α is a normalization constant, which ensures that

$$\sum_{x \in D_{X_j}} \mathrm{P}(x | t{\downarrow}^{U}_{pa(X_j)}) = 1$$

This last assumption may seem unrealistic for most probabilistic models, but it is routinely made for CSPs: each constraint on a variable acts on that variable independently of any other constraint on that variable. It is this assumption that results in the normalization constant for pAC, as mentioned above.

These assumptions allow us to express the marginal probability of $x_j \in D_j$:

$$\mathrm{P}(x_j) = \alpha \prod_{Y \in pa(X_j)} \sum_{y \in D_Y} \mathrm{P}(x_j | y) \mathrm{P}(y)$$

In other words, when the assumptions hold, the marginal on X_j can be determined from the marginals on its parents. This gives a top-down algorithm for determining marginals in the Bayesian network, which is equivalent to directed arc consistency [3]. The reader is invited to compare this equation with pAC equation for $\mathrm{M}_X^{(k)}.def$.

4 MaxCSP

In this section, we focus on the application of Theorem 2 to the partial CSP problem, also called MaxCSP. In this problem, we take an over-constrained classical CSP, and the goal is to find an assignment that maximizes the number of satisfied constraints. The usual approach involves branch and bound search, possibly with heuristics. In the following, we derive a local consistency algorithm for MaxCSP that can be used as a heuristic during search.

4.1 MaxAC: Constraint propagation for MaxCSP

In MaxCSP, constraints are classical, but the quantity we are optimizing is not satisfiability, but the number of satisfied constraints. Thus, if there are n constraints in the problem, the semiring structure for MaxCSP is $\langle \{0, \ldots, n\}, +, \max, 0, n \rangle$ where n is the number of constraints. The multiplicative operator of the semiring, $\times$, is instantiated here as integer addition. In other words, if two constraints are combined, then any given tuple can satisfy either, both, or neither constraint, and the number of constraints satisfied is determined by simply adding $def(t)$ values. The additive operator of the semiring is $\max$.

The xAC equations for MaxCSP are as follows:

$$
\begin{aligned}
C_{XY_i}.def &: D_X \times D_{Y_i} \to \{0, 1\} \\
\mathrm{S}_{XY_i}^{(k)}.def(x) &= \max_{y \in D_{Y_i}} \left(C_{XY_i}.def(x, y) + \mathrm{P}_{XY_i}^{(k)}.def(y) \right) \\
\mathrm{M}_X^{(k)}.def(x) &= \sum_{i=1}^{m} \mathrm{S}_{XY_i}^{(k)}.def(x) \\
\mathrm{P}_{Y_i X}^{(k+1)}.def(x) &= \sum_{j \neq i} \mathrm{S}_{XY_j}^{(k)}.def(x) \\
\mathrm{P}_{Y_i X}^{(0)}.def(x) &= 0
\end{aligned}
$$

We refer to these equations as the MaxAC equations.

Theorem 2 states that under the assumption of tree-structured constraint graphs, the quantity $\mathrm{M}_X^{(k)}.def(x)$ represents the maximum number of satisfied constraints satisfied by any solution involving $X = x$. It is a simple matter to maximize for the best value in D_X. Furthermore, the computation of $\mathrm{M}_X^{(k)}.def(x)$ can be done efficiently. Therefore, we have the following algorithm: for every variable in some order, we maximize the marginal, and fix the variable to the maximizing value. When all the variables have been assigned in this way, a MaxCSP solution has been found.

When the CSP is not tree-structured, the theorem does not apply. The equations as Theorem 2 describes them, result in marginal constraint functions that increase without bound as k increases. This same criticism was made toward solution counting (the integer precursor to pAC; see Section 3.2). In practice, however, convergence of pAC has been demonstrated empirically.

The MaxAC marginals are not constrained by the equations to any finite bound. The pAC marginals are constrained by normalization to be in the range $[0, 1]$.

We hypothesized that if the MaxAC equations were augmented by some means of limiting the range of the marginals, useful convergence might be obtained. We used the following method.

Define $\mathrm{M}_X^{(k)}.def$ so that the marginal distribution is represented in terms of the difference from the minimum value:

$$\mathrm{M}_X^{(k)}.def(x) = -\beta + \sum_Y \mathrm{S}_{YX}^{(k)}.def(x)$$

where

$$\beta = \min_x \sum_Y \mathrm{S}_{YX}^{(k)}.def(x)$$

This approach is motivated by the observation that the count of satisfied constraints may increase with each iteration, but the difference between the counts may not change very much. The value of this approach is the subject of the next section.

4.2 Preliminary evaluation

In very special case of tree-structured constraint graphs, the original unnormalized MaxAC equations are exact. However, we are interested in the possibility of using these equations in more general problems, as the basis for computing heuristics that can be used during search. To deal with CSPs that are not tree-structured, we introduced a "normalization" method to the MaxAC equations. In this section, we briefly present some preliminary empirical evidence that suggests that this kind of constraint propagation can be informative.

The quality of the information provided by the MaxAC equations is shown by a positive correlation of the approximate marginals with the exact marginals. The computation of exact values limits the size of the problems feasible for the comparison. The MaxAC equations were applied (using difference from minimum as the normalizing step) to random over-constrained CSPs. We used two samples of 150 problems of 10 variables and 5 values each. The first sample was computed using the "flawed" random CSP model as discussed by [4], for $p_1 \in [0.5, \ldots, 1.0]$ and $p_2 \in [0.5, \ldots, 0.9]$. The second sample were constructed using a unique random CSP model, which is based on the "flawed" model, but in about one-third of the constraints, the disallowed pairs are chosen from a smaller subset of possible pairs. In effect this model puts clusters, or "holes," of disallowed pairs into the constraint. This model was constructed to avoid "uniformity" of typical random models, in which it is expected that every value will appear in a MaxCSP solution with about the same number of satisfied constraints. Obviously, the uniform random CSPs does cause problems for MaxAC, but they are also

uninteresting, as a random assignment would be expected to be fairly close to optimal. The second sample was generated with $p_1 \in [0.5, \ldots, 1.0]$ and $p_2 \in [0.5, \ldots, 0.9]$, as before.

	Min	Max	Median	Average	Std. Dev.
Sample 1	-0.0936	0.995	0.625	0.620	0.253
Sample 2	0.292	1.0	0.929	0.853 i	0.163

Table 1. Correlation between approximate marginals, computed by MaxAC, and the exact marginals for two sets of random over-constrained CSPs.

The comparison was determined by computing the correlation coefficient between the approximate marginal and the exact marginal, for each variable in the CSP, and then averaging over all variables. The results are shown in Table 1. For the first sample, the average correlation ranges from just below zero (indicating no correlation) to just below 1.0, indicating perfect correlation. For this sample, the median is 0.625, which indicates that for half of the problems, the MaxAC approximation is strongly correlated with the exact values.

The second sample has a stronger correlation between the approximate and exact marginals, as indicated by the median correlation of 0.929. These results show that MaxAC approximations can be very good.

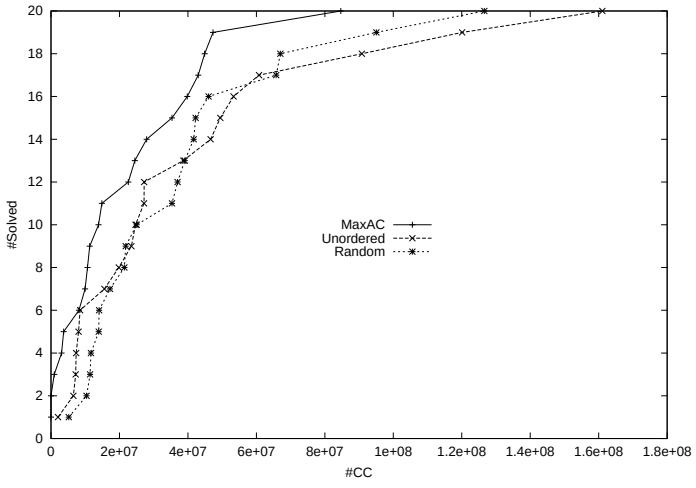

Fig. 1. A plot of the number of consistency checks required to find the MaxCSP assignment for flawed random CSPs, using various heuristics.

The second part of our preliminary results made use of MaxAC as a heuristic during search. We constructed over-constrained random classical CSPs, and employed branch and bound search with MaxAC as a heuristic, without heuristic guidance (i.e., lexicographical ordering) and also with a random value ordering. The MaxAC marginals were used as a static value ordering heuristic (i.e., exploring values with high MaxAC counts first). As well, the MaxAC marginals were used to find a lower bound on the MaxCSP solution. This was done by constructing an assignment based on the MaxAC marginals in a greedy manner: choose the value that maximized the MaxAC marginal for each variable.

The results are shown in Figures 1 and 2. We constructed 20 CSPs with 10 variables and 10 values with $p_1 = 0.8$ and $p_2 = 0.8$. We recorded the number of consistency checks required to find the MaxCSP assignment (the search carries on in vain trying to find a better solution). The graphs show how many of the problems were solved using a given number of consistency checks. Clearly, using MaxAC is superior to the other strategies, and therefore MaxAC does provide information that can be valuable during search.

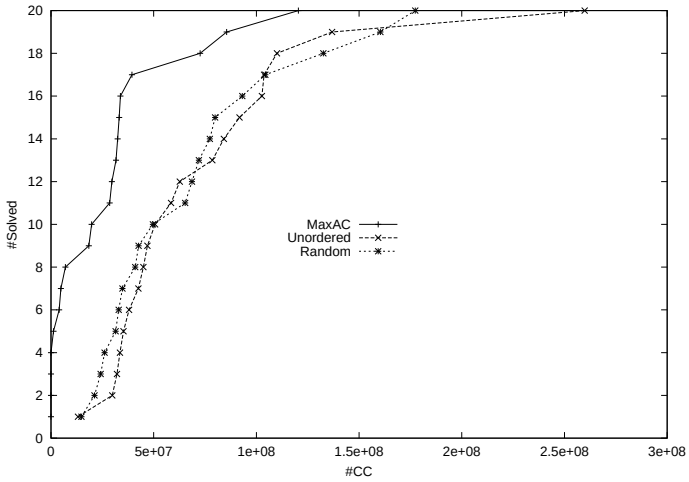

Fig. 2. A plot of the number of consistency checks required to find the MaxCSP assignment for skewed random CSPs using various heuristics.

5 Summary and Future Work

We have presented a constraint propagation algorithm schema, xAC, that generalizes classical arc consistency, based on the idea of approximating the marginal solution of the CSP. The schema can be instantiated by providing operators for combining and projecting constraints. xAC uses local information to compute global properties such as

satisfiability or probability whenever the CSP has a constraint graph that is tree structured. When the constraint graph is not tree structured, xAC can still be applied to these problems, as an approximation method. In some applications, such as classical arc consistency, convergence to a reasonable approximation is guaranteed by the operators. In other cases, convergence is not guaranteed, but in some of these cases, convergence can be assisted by the choice of a normalization operator. Several instances of xAC have been observed to converge to useful approximations when convergence is not guaranteed theoretically, such as pAC [5], belief propagation in Bayesian networks [8], and MaxAC (Section 4).

The approach differs from the k-consistency approach presented in [2] in an important way: xAC does not depend on the idempotence of the multiplicative operator used to combine constraint values. This is an advantage when the operator is not idempotent, but a disadvantage, as it means that in some cases, convergence is not guaranteed.

The xAC schema can be applied to any semiring CSP instances, by instantiating the framework with particular sets and operators. For example, the class of fuzzy CSPs is the SCSP $\langle [0,1], \max, \min, 0, 1 \rangle$ [2]. Thus the xAC equations for fuzzy CSPs are as follows:

$$
\begin{aligned}
C_{XY_i}.def &: D_X \times D_{Y_i} \rightarrow [0,1] \\
\mathrm{S}^{(k)}_{XY_i}.def(x) &= \max_{y \in D_{Y_i}} \left(\min \left\{ C_{XY_i}.def(x,y),\ \mathrm{P}^{(k)}_{XY_i}.def(y) \right\} \right) \\
\mathrm{M}^{(k)}_{X}.def(x) &= \min_{i=1}^{m} \mathrm{S}^{(k)}_{XY_i}.def(x) \\
\mathrm{P}^{(k+1)}_{Y_iX}.def(x) &= \min_{j \neq i} \mathrm{S}^{(k)}_{XY_j}.def(x) \\
\mathrm{P}^{(0)}_{Y_iX}.def(x) &= 1
\end{aligned}
$$

It is intuitively obvious that these equations will converge, even for problems within the class which do not have tree-structured constraint graphs. It is an open empirical question as to whether the fixed point will be useful during search. As with pAC and MaxAC, some form of normalization may be necessary to complete the equations for non-tree structured problem instances. This is also an open question.

Future work includes a more formal treatment of convergence. It may be that the work of [2] in proving convergence for their constraint propagation algorithm can be applied to xAC. As well, we do not yet have a clear picture of the relation between xAC and other proposals for constraint propagation in non-classical CSPs. Our evaluation of xAC applied to MaxCSP is preliminary and encouraging, and further empirical work is on-going.

The xAC schema has been expressed in terms of semiring CSPs, but it does not depend on the semiring structure. The relationship between xAC and Markov random fields is strong, and we are looking at how to formalize the relationship more precisely.

References

[1] S. Bistarelli, R. Gennari, and F. Rossi. Constraint propagation for Soft Constraint Satisfaction Problems: Generalization and Termination Conditions. In *Proceedings of the Sixth International Conference on Principles and Practice of Constraint Programming (CP2000)*, 2000.

[2] S. Bistarelli, U. Montanari, F. Rossi, T. Schieux, G. Verfaille, and H. Fargier. Semiring-Based CSPs and Valued CSPs: Frameworks, Properties and Comparison. *Constraints*, 4(3):199–240, 1999.

[3] Rina Dechter and Judea Pearl. Network-based heuristics for constraint-satisfaction problems. *Artificial Intelligence*, 34:1–34, 1988.

[4] Ian Gent, Ewan MacIntyre, Patrick Prosser, Barbara Smith, and Toby Walsh. Random constraint satisfaction: Flaws and structure. Technical Report 98.23, University of Leeds, School of Computer Studies, 1998.

[5] Michael C. Horsch and William S. Havens. Probabilistic Arc Consistency: A connection between constraint reasoning and probabilistic reasoning. In *Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence*, pages 282–290, 2000.

[6] Jin H. Kim and Judea Pearl. A computational model for causal and diagnostic reasoning in inference systems. In *Proceedings of the Eighth International Joint Conference on Artificial Intelligence*, pages 190–193, 1983.

[7] Alan K. Mackworth. Consistency in networks of relations. *Artificial Intelligence*, 8(1):99–118, 1977.

[8] Kevin P. Murphy, Yair Weiss, and Michael I. Jordan. Loopy belief propagation for approximate inference: An empirical study. In *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*, pages 467–475, 1999.

[9] Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Reasoning*. Morgan Kaufmann Publishers, Los Altos, 1988.