Constraint-Based Inference Using Local Consistency in Junction Graphs

Le Chang and **Alan K. Mackworth** [1]

Abstract.
The concept of local consistency plays a central role in constraint satisfaction and has been extended to handle general constraint-based inference (CBI) problems. We propose a family of novel generalized local consistency concepts for the junction graph representation of CBI problems. These concepts are based on a general condition that depends only on the existence and property of the multiplicative absorbing element and does not depend on the other semiring properties of CBI problems. We present several local consistency enforcing algorithms and their approximation variants. Theoretical complexity analyses and empirical experimental results for the application of these algorithms to both MaxCSP and probability inference are given. We also discuss the relationship between these local consistency concepts and message passing schemes such as junction tree algorithms and loopy message propagation.

1 Introduction

The concept of local consistency plays a central role in constraint satisfaction. Given a constraint satisfaction problem (CSP), local consistency can be characterized as deriving new, possibly tighter, constraints based on local information. The derived constraints simplify the representation of the original CSP without the loss of solutions. This can be seen as a preprocessing procedure. For example, a value may be removed from a variable domain by the preprocessing because it violates these derived constraints. Both systematic approaches, such as inference or propagation algorithms, and stochastic approaches, such as local searches, benefit from these simplifications or domain size reduction. Among the family of local consistency enforcing algorithms or filtering algorithms, arc consistency [16] is one of the most important techniques for binary classic CSPs. It is straightforward to extend it as generalized arc consistency [17] to handle non-binary classic CSPs. Many stronger local consistencies [18, 25, 14] have been studied within the constraint programming community. Based on the Semiring CSP [4] and Valued CSP [22] frameworks, arc consistency has also been extended, as soft arc consistency [7, 3], to handle over-constrained and preference-based problems that can be modelled as soft CSPs. Recently, we presented a weaker condition [6] based on a semiring based framework for constraint-based inference (CBI) problems [5]. More specifically, we reduce a CBI problem to its underlying classic CSP according to the weaker condition and then apply the generalized arc consistency approach to general CBI problems beyond classic and soft CSPs. The weaker condition proposed in [6] has also been relaxed to fit generalized approximate preprocessing schemes.

In this paper we take a step beyond the generalized arc consistency approach and propose a new family of generalized local consistency concepts for the junction graph representation of CBI problems. These concepts are based on a general condition that depends only on the existence and properties of the multiplicative absorbing element and does not depend on other semiring properties of CBI problems. We present several local consistency enforcing algorithms with various levels of enforcement and corresponding theoretic and empirical complexity analyses. We show in this paper that some of these algorithms can be seen as generalized versions of well-known local consistency enforcing techniques in CSPs and can be exported to other domains. Other abstract local consistency concepts are novel to the constraint programming community and provide efficient preprocessing results with a user-specified approximation threshold. We also discuss the relationship between these local consistency concepts and message passing schemes such as junction tree algorithms and loopy message propagation. Local consistencies can be achieved along with message propagation and can improve the efficiency of message passing schemes.

In this paper, we use bold letters to denote sets of elements and regular letters to denote individual elements. Given a set of elements $\mathbf{X}$ and an element $Z \in \mathbf{X}$, $\mathbf{X}_{-Z}$ denotes the set of elements $\mathbf{X} - \{Z\}$.

2 The Constraint-Based Inference Framework

Constraint-Based Inference (CBI) is an umbrella term for a class of various superficially different problems including probabilistic inference, decision-making under uncertainty, constraint satisfaction problems, propositional satisfiability problems, decoding problems, and possibility inference. We abstract these problems into a single formal framework [5] using the algebraic semiring structure $\mathbf{S} = \langle \mathbf{A}, \oplus, \otimes \rangle$ where constraint combination (constraint aggregation) is represented by the abstract multiplicative operator $\otimes$ and constraint marginalization (domain reduction) is represented by the abstract additive operator $\oplus$. This framework, based on the synthesis of the existing abstract representation and algorithmic frameworks from various fields [4, 22, 11, 10, 1, 21], provides a broader coverage of both the problem space and the inference algorithm space.

A CBI problem $\mathbf{P}$ in this framework is a tuple $(\mathbf{X}, \mathbf{D}, \mathbf{S}, \mathbf{F})$, where $\mathbf{X}$ is a set of variables, $\mathbf{D}$ is a corresponding set of finite domains for the variables, $\mathbf{S} = \langle \mathbf{A}, \oplus, \otimes \rangle$ is a commutative semiring, and $\mathbf{F}$ is a set of constraints. Each constraint is a function that maps a subset of variables to values in $\mathbf{A}$. More specifically, we use $Scope(f)$ to denote the subset of variables that is in the scope of the constraint f and $\mathbf{D}_X$ to represent the domain of variable X. Two constraint operations then are defined: (1) A

[1] Department of Computer Science, University of British Columbia, Canada, email: {lechang,mack}@cs.ubc.ca

constraint $g = f_1 \otimes f_2$ is the combination of two constraints f_1 and f_2 if $Scope(g) = Scope(f_1) \cup Scope(f_2)$ and $g(\mathbf{w}) = f_1(\mathbf{w}_{\downarrow Scope(f_1)}) \otimes f_2(\mathbf{w}_{\downarrow Scope(f_2)})$ for every value assignment $\mathbf{w}$ of variables in $Scope(g)$; and (2) A constraint $g = \bigoplus_X f$ is the constraint that marginalizes out $X \in Scope(f)$ from a constraint f if $Scope(g) = Scope(f) - \{X\}$ and $g(\mathbf{w}) = \bigoplus_{x_i \in \mathbf{D}_X} f(x_i, \mathbf{w})$ for every value assignment $\mathbf{w}$ of variables in $Scope(g)$. Given a CBI problem and the two constraint operations specified above, the inference task is defined as computing $g_{CBI}(\mathbf{Z}) = \bigoplus_{\mathbf{Y}} \bigotimes_{f \in \mathbf{F}} f$, where $\mathbf{Z}$ is a subset of variables of interest and $\mathbf{Y} = \mathbf{X} - \mathbf{Z}$. If $\oplus$ of the semiring $\mathbf{S}$ is idempotent, the allocation task is defined as computing $\mathbf{y} = \arg \bigoplus_{\mathbf{Y}} \bigotimes_{f \in \mathbf{F}} f$, where $\arg$ is a prefix of operator $\oplus$. In other words, $\arg \oplus$ is an operator that returns the arguments of the $\oplus$ operator. For example, the classic CSP can be seen as a CBI problem using the commutative semiring $\mathbf{S_{CSP}} = \langle\{FALSE, TRUE\}, \vee, \wedge\rangle$ to embed it into the framework. The inference task of classic CSP is to compute the truth value of $g_{CBI}() = \bigvee_{\mathbf{X}} \bigwedge_{f \in \mathbf{F}} f$ and the allocation task is find a complete assignment of variables that satisfies g_{CBI}.

Our local consistency concepts are based on this CBI framework and apply to CBI problems with commutative semirings that are eliminative. A commutative semiring $\mathbf{S} = \langle \mathbf{A}, \oplus, \otimes \rangle$ is *eliminative* [6] if there exists a *multiplicative absorbing element* $\alpha_\otimes \in \mathbf{A}$ ($\alpha_\otimes \otimes a = \alpha_\otimes, \forall a \in \mathbf{A}$) and $\alpha_\otimes$ is equal to the *additive identity element* $\mathbf{0}$ ($\mathbf{0} \oplus a = a, \forall a \in \mathbf{A}$). Furthermore, our approximate local consistency concepts apply to CBI problems with commutative semirings that are eliminative and monotonic. A commutative semiring $\mathbf{S} = \langle \mathbf{A}, \oplus, \otimes \rangle$ is *monotonic* [6] if there exists a total order $\leq_{\mathbf{S}}$ on $\mathbf{A}$, the additive identity element $\mathbf{0}$ is the minimum element w.r.t. $\leq_{\mathbf{S}}$, $a \leq_{\mathbf{S}} b$ implies $a \oplus c \leq_{\mathbf{S}} b \oplus c$ and $a \otimes c \leq_{\mathbf{S}} b \otimes c$, $\forall a, b, c \in \mathbf{A}$. For example, Weighted CSPs can be embedded into the CBI framework using the semiring $\mathbf{S_{WCSP}} = \langle \mathbb{N}, \min, + \rangle$. Because the multiplicative absorbing element $\alpha_\otimes$ of the semiring $\mathbf{S_{WCSP}}$ is equal to the additive identity element $\mathbf{0}$ that is equal to $+\infty$, $\mathbf{S_{WCSP}}$ is eliminative. Also we can show that $\mathbf{S_{WCSP}}$ is monotonic. More details on eliminative and monotonic semirings can be found in [6].

A junction graph $\mathcal{J} = (\mathcal{C}, \mathcal{S})$ of a CBI problem $\mathbf{P} = (\mathbf{X}, \mathbf{D}, \mathbf{S}, \mathbf{F})$ is defined as follows: $\mathcal{C} = \{C_1, \cdots, C_n\}$ is a set of clusters, each cluster C_i is an aggregation of variables that is a subset of $\mathbf{X}$ and has attached initially a local constraint ϕ_{C_i} with null scope and $\mathbf{1}$ as the default constraint value or cost ($\mathbf{1}$ is the identity element for $\otimes$); $\mathcal{S} = \{S_{ij} | C_i, C_j \in \mathcal{C}\}$ is a set of separators between C_i and C_j if $C_i \cap C_j \neq \emptyset$ and S_{ij} is an aggregation of variables that consists of $C_i \cap C_j$. A junction graph satisfies the condition that for any constraint $f \in \mathbf{F}$, there exists a cluster $C_i \in \mathcal{C}$ s.t. $Scope(f) \subseteq C_i$. The definition of junction graph ensures that the subgraph induced by any variable is connected. We say a junction graph is *initialized* if for each constraint $f \in \mathbf{F}$, we choose a cluster C_i s.t. $Scope(f) \subseteq C_i$ and update ϕ_{C_i} by $\phi_{C_i} \otimes f$ [2].

3 Local Consistency for CBI Problems

We present here novel local consistency concepts for initialized junction graphs of a CBI problem with an eliminative semiring. If the semiring used to represent a CBI problem is both eliminative and monotonic, it is straightforward to modify these concepts to approximate local consistencies using an element $\epsilon \in \mathbf{A}$ to approximate the multiplicative absorbing element $\alpha_\otimes$ that is equal to the additive identity element $\mathbf{0}$ for an eliminative commutative semiring, and using $\leq_{\mathbf{S}}$ to replace $\neq$ in the following definitions.

[2] Alternatively, we can keep a set of constraints inside each cluster and not initialize the potential with a combination of the constraints. Combination of constraints can be computed along with the local consistency enforcement using more space-efficient methods.

```
Input: A CBI problem $\mathbf{P} = (\mathbf{X}, \mathbf{D}, \mathbf{S}, \mathbf{F})$ and its initialized junction graph representation $\mathcal{J} = (\mathcal{C}, \mathcal{S})$
Output: A Single Cluster Consistent CBI problem $\mathbf{P}' = (\mathbf{X}, \mathbf{D}', \mathbf{S}, \mathbf{F}')$
 1: for each $C_i \in \mathcal{C}$ do
 2:   for each $X \in Domain(\phi_{C_i})$ do
 3:     if REVISE$(X, \phi_{C_i})$ then
 4:       for each $C_j \in \mathcal{C}$ do
 5:         if $X \in Scope(\phi_{C_j})$ then
 6:           Remove all tuples in $\phi_{C_j}$ with the value that is removed from $X$
 7:         end if
 8:       end for
 9:     end if
10:   end for
11: end for
12: Return $\mathbf{P}' := \mathbf{P}$
```

Figure 1. Single cluster consistency enforcing algorithm (*SCC-Enforcing*).

3.1 Single, Directional and Neighborhood Cluster Consistencies

The fundamental concept of local consistency for an initialized junction graph of a CBI problem with an eliminative commutative semiring is *single cluster consistency*. Here we consider only the local constraints attached to a single cluster and do not consider the effects of other clusters. Formally:

Definition 1 (Single Cluster Consistency (SCC)) *A cluster C_i of an initialized junction graph is locally consistent if $\forall X \in Scope(\phi_{C_i})$, $\forall x \in \mathbf{D}_X$, $\exists \mathbf{w}$, a value assignment of variables $Scope(\phi_{C_i})_{-X}$, s.t. $\phi_{C_i}(x, \mathbf{w}) \neq \alpha_\otimes$. An initialized junction graph of a CBI problem is Single Cluster Consistent if all the clusters are consistent.*

Single cluster consistency covers the definition of Generalization of Generalized Arc Consistency (GGAC) [6], which abstracts Generalized Arc Consistency (or Hyper-Arc Consistency) in constraint programming. If the junction graph of a CBI problem is primal, in other words, there is one cluster that corresponds to exactly one constraint (bijection), SCC is identical to GGAC. If the junction graph is constructed without satisfying this special structural requirement, SCC is stronger than GGAC in general.

Figure 1 shows a generalized routine for enforcing single cluster consistency for an initialized junction graph of a CBI problem $\mathbf{P} = (\mathbf{X}, \mathbf{D}, \mathbf{S}, \mathbf{F})$ with an eliminative commutative semiring $\mathbf{S}$. The procedure *REVISE* of *SCC-Enforcing* is shown in Figure 2.

We also introduce two other local consistencies for an initialized junction graph of a CBI problem that are stronger than Single Cluster Consistency. They are Directional Cluster Consistency and Neighborhood Cluster Consistency. Effects of other clusters in the junction graph are taken into account. The distinction between these two local consistencies is based on which clusters are selected for consideration.

```
**Input:** A variable $X \in \mathbf{X}$ and a constraint $f$ with $X \in Scope(f)$
**Output:** TRUE if a value is removed from the domain of $X$, FALSE for else
1:  $flag :=$ TRUE
2:  for each $x \in \mathbf{D}_X$ do
3:    for each value assignment $\mathbf{w}$ of $Scope(f)_{-X}$ do
4:      if $f(x, \mathbf{w}) \neq \alpha_{\otimes}$ then
5:        $flag :=$ FALSE
6:        Break loop
7:      end if
8:    end for
9:    if $flag$ then
10:     Remove $x$ from $\mathbf{D}_X$
11:     Return TRUE
12:   end if
13: end for
14: Return FALSE
```

Figure 2. Procedure *REVISE*(X, f) for eliminating a domain value from a variable X according to the local constraint f.

Definition 2 (Directional Cluster Consistency (DCC)) *Given a total ordering of the clusters and a cluster C_i of an initialized junction graph. Let S_{ij} be a separator that connects another cluster C_j to C_i and $\mathbf{L}(C_i)$ be a subset of clusters that consist of lower order neighbor clusters of C_i. Define $g_i = \phi_{C_i} \otimes \bigotimes_{C_j \in \mathbf{L}(C_i)} (\bigoplus_{C_j - S_{ij}} \phi_{C_j})$. We say C_i is directional consistent if $\forall X \in Scope(g_i)$, $\forall x \in \mathbf{D}_X$, $\exists \mathbf{w}$, a value assignment of variables $Scope(g_i)_{-X}$, s.t. $g_i(x, \mathbf{w}) \neq \alpha_{\otimes}$. An initialized junction graph of a CBI problem is Directional Cluster Consistent given a total ordering of the clusters if all clusters of the graph are directional consistent.*

Definition 3 (Neighborhood Cluster Consistency (NCC)) *Given a cluster C_i of an initialized junction graph, Let $\mathbf{N}(C_i)$ be a subset of clusters that are neighbor clusters of C_i. Define $g_i = \phi_{C_i} \otimes \bigotimes_{C_j \in \mathbf{N}(C_i)} (\bigoplus_{C_j - S_{ij}} \phi_{C_j})$. We say C_i is neighborhood consistent if $\forall X \in Scope(g_i)$, $\forall x \in \mathbf{D}_X$, $\exists \mathbf{w}$, a value assignment of variables $Scope(g_i)_{-X}$, s.t. $g_i(x, \mathbf{w}) \neq \alpha_{\otimes}$. An initialized junction graph of a CBI problem is Neighborhood Cluster Consistent if all clusters are neighborhood consistent.*

We revise the single cluster consistency enforcing algorithm in Figure 1 to directional cluster consistency and neighborhood cluster consistency enforcing algorithms by updating the local potential ϕ_{C_i} according to the definition, as shown in Figure 3 and 4, respectively.

3.2 Approximate Local Consistencies

Given a CBI problem $\mathbf{P} = (\mathbf{X}, \mathbf{D}, \mathbf{S}, \mathbf{F})$, if the commutative semiring $\mathbf{S} = \langle \mathbf{A}, \oplus, \otimes \rangle$ is both eliminative and monotonic, we propose an approximation scheme to enforce local consistency for its initialized junction graph representation with a user-controlled threshold. More specifically, we use an element $\epsilon \in \mathbf{A}$ to approximate the multiplicative absorbing element $\alpha_{\otimes}$ that is equal to the additive identity element $\mathbf{0}$ for an eliminative commutative semiring, and use $\leq_{\mathbf{S}}$ to replace $\neq$ in the previous local consistency definitions. The monotonic properties for both multiplicative and additive operators in a monotonic semiring ensure that this approximation always returns a lower bound estimate of the inference task for a given CBI problem [6]. Correspondingly, the procedure *REVISE* in Figure 2 is modified to handle approximate local consistency enforcing tasks, as shown in Figure 5. All the local consistency enforcing algorithms discussed in the previous section then can be modified respectively.

```
**Input:** A CBI problem $\mathbf{P} = (\mathbf{X}, \mathbf{D}, \mathbf{S}, \mathbf{F})$, its initialized junction graph representation $\mathcal{J} = (\mathcal{C}, \mathcal{S})$, and a total ordering $\mathcal{OC}$ of clusters in $\mathcal{C}$
**Output:** A Directional Cluster Consistent CBI problem $\mathbf{P}' = (\mathbf{X}, \mathbf{D}', \mathbf{S}, \mathbf{F}')$
1:  for $i = |\mathcal{C}| - 1$ to 1 do
2:    Let $C_i = \mathcal{OC}[i]$
3:    $\phi_{old} := \phi_{C_i}$
4:    for $j = |\mathcal{C}|$ to $i + 1$ do
5:      Let $C_j = \mathcal{OC}[j]$
6:      $\phi_{C_i} := \phi_{C_i} \otimes (\bigoplus_{C_j - S_{ij}} \phi_{C_j})$
7:    end for
8:    if REVISE$(X, \phi_{C_i})$ then
9:      for each $C_k \in \mathcal{C}$ do
10:       if $X \in Scope(\phi_{C_k})$ then
11:         Remove all tuples in $\phi_{C_j}$ with the value that is removed from $X$
12:       end if
13:     end for
14:   end if
15:   $\phi_{C_i} := \phi_{old}$
16: end for
17: Return $\mathbf{P}' := \mathbf{P}$
```

Figure 3. Directional cluster consistency (*DCC*) enforcing algorithm.

```
**Input:** A CBI problem $\mathbf{P} = (\mathbf{X}, \mathbf{D}, \mathbf{S}, \mathbf{F})$ and its initialized junction graph representation $\mathcal{J} = (\mathcal{C}, \mathcal{S})$
**Output:** A Single Cluster Consistent CBI problem $\mathbf{P}' = (\mathbf{X}, \mathbf{D}', \mathbf{S}, \mathbf{F}')$
1:  for each $C_i \in \mathcal{C}$ do
2:    $\phi_{old} := \phi_{C_i}$
3:    for each $C_j \in \mathbf{N}(C_i)$ do
4:      $\phi_{C_i} := \phi_{C_i} \otimes (\bigoplus_{C_j - S_{ij}} \phi_{C_j})$
5:    end for
6:    if REVISE$(X, \phi_{C_i})$ then
7:      for each $C_k \in \mathcal{C}$ do
8:        if $X \in Scope(\phi_{C_k})$ then
9:          Remove all tuples in $\phi_{C_j}$ with the value that is removed from $X$
10:       end if
11:     end for
12:   end if
13:   $\phi_{C_i} := \phi_{old}$
14: end for
15: Return $\mathbf{P}' := \mathbf{P}$
```

Figure 4. Neighborhood cluster consistency (*NCC*) enforcing algorithm.

```
Input: A variable $X \in \mathbf{X}$, a constraint $f$, an element $\epsilon \in \mathbf{A}$
Output: TRUE if a value is removed from the domain of $X$; FALSE
    if else
 1: $flag :=$ TRUE
 2: for each $x \in \mathbf{D}_X$ do
 3:   for each value assignment $\mathbf{w}$ of $Scope(f)_{-X}$ do
 4:     if $\epsilon \leq_{\mathbf{S}} f(x, \mathbf{w})$ then
 5:       $flag :=$ FALSE
 6:       Break loop
 7:     end if
 8:   end for
 9:   if $flag$ then
10:     Remove $x$ from $\mathbf{D}_X$
11:     Return TRUE
12:   end if
13: end for
14: Return FALSE
```

Figure 5. Procedure ϵ-*REVISE*(X, f, ϵ) for eliminating a domain value from a variable X according to the approximate threshold ϵ of a local constraint f.

4 Complexities and Discussion

The worst case space complexities of all three local consistency enforcing algorithms are the same: linear in the number of clusters in the junction graph and exponential in the maximal cluster size. The worst case time complexities are linear in the size of the junction graph and exponential in maximal cluster size too. We compare their upper bounds for time and space of local consistency enforcing algorithms for initialized junction graphs in Table 1. All of them use the same space, though achieving Single Cluster Consistency uses the least time, followed by Directional Cluster Consistency, and then Neighborhood Cluster Consistency.

Table 1. Time and space upper bound comparison among various local consistency enforcing algorithms for a junction graph $\mathcal{J} = (\mathcal{C}, \mathcal{S})$ of a given CBI problem, where $d = \max_{D_i \in \mathbf{D}} |D_i|$ and $k = \max_{C_i \in \mathcal{C}} |C_i|$.

	SCC	DCC	NCC
Time	$\|\mathcal{C}\|d^{k+1}$	$(\|\mathcal{S}\| + \|\mathcal{C}\|)d^{k+1}$	$(2\|\mathcal{S}\| + \|\mathcal{C}\|)d^{k+1}$
Space	$\|\mathcal{C}\|d^{k+1}$	$\|\mathcal{C}\|d^{k+1}$	$\|\mathcal{C}\|d^{k+1}$

As shown in Table 1, the upper bounds of both time and space for achieving local consistencies using cluster consistency enforcing algorithms proposed in this paper are bounded by the maximum cluster size as well as the structure of the junction graph for a given CBI problem. Intuitively a simple junction graph implies large cluster sizes, so there is a tradeoff between the size of the graph and the largest cluster when constructing a junction graph. Various heuristic search approaches that can be used to construct junction graphs are discussed in [5].

The space complexity of SCC can be improved using the GAC approaches for global constraints such as in [2] and [24] for constraint networks. The basic idea is not to initialize with a combination of the constraints that are allocated to a single cluster into a large constraint, but to keep a set of constraints inside each cluster. When we need to compute a combination of constraints during SCC enforcing, we only need to compute new tuples that are not marked as deleted. The same approach can also be applied to DCC and NCC enforcing, though the upper bounds for space will not change.

The junction tree representation is a special case of junction graphs that satisfies the tree property. The junction tree algorithm [23] is a widely studied inference algorithm in probability inference that utilizes the properties of the junction tree structure. It is also generalized to handle constraint-based inference problems [5], based on the seminal work on constraint programming [8, 26] and the latest general algorithmic framework [10]. Given the identical message representation and updating scheme in the inward phase of the junction tree algorithm and our directional cluster consistency enforcing algorithm (with a cluster order given by the width-first traverse starting from the root cluster), it is straightforward to show that directional cluster consistency can be achieved along with the inward message passing in the junction tree algorithm, if the junction graph of a given CBI problem satisfies the junction tree properties. In this case, the potential reset steps (Lines 3 and 15 in Figure 3) are not necessary if the constraint combination operator $\otimes$ is idempotent. If it is not, the complement operator $\oslash$ of $\otimes$ should be introduced to cancel the duplicated costs or potentials passing from the children of the child clusters [5]. We refer readers to [12] for further discussion of this technique from the semiring perspective. Analogous approaches to cancelling the double-counted information in local consistency methods can also be found in [13, 7]. A CBI problem processed by such a *DCC-enforcing* procedure then can be solved by a search starting from the root cluster, which is a process equivalent to outward message propagation in the junction tree algorithm. This observation ensures that we can perform the message passing of junction tree algorithms and at the same time simplify the original problem representation according to the DCC enforcement. Performing the message propagation and the simplification together reduces both the time and space complexities of the junction tree algorithm. The nature of the message passing scheme in the junction tree algorithm ensures that the directional cluster consistency enforcing can be performed in parallel for clusters in different branches of the tree. We plan to investigate different parallel and hybrid DCC-enforcing techniques following the results of [26] in future work.

Loopy message propagation [19] is another widely studied approximate inference approach based on the junction graph representation in probability inferences. It is also generalized to apply to other CBI problems [5] using the semiring concepts. Neighborhood cluster consistency can be achieved along with each message updating step in the generalized loopy message propagation. This can be achieved without additional computational cost except for invalid value detection at each cluster. Similar to the junction tree algorithm, the potential reset steps (Lines 2 and 13 in Figure 4) are not necessary if the constraint combination operator $\otimes$ is idempotent. If $\otimes$ is not idempotent, we can cancel the duplicated costs or potentials coming from the same cluster via different paths by introducing the complement operator $\oslash$ of $\otimes$. The time and space complexities of loopy message propagation are reduced after invalid values are removed from the CBI problem following NCC enforcement. The message updating step as well as NCC enforcement of the generalized loopy message propagation can be performed in all clusters in parallel, saving significant computational cost if parallel computing is feasible.

5 Experimental Results

We discuss in this section experimental results of applying the local consistency enforcing algorithms proposed in this paper to the junction graph representation of Weighted CSP and Probability Assessment that can be modelled as CBI problems. These preprocessing or filtering algorithms simplify the original problem so that inference algorithms can then be applied with less computational complexity. A workstation with a Pentium 4 3.0GHz CPU and 1 GB memory running SuSE Linux 9.1 was used to run the experiments in this section.

5.1 Weighted CSP

Weighted CSP is a direct extension of MaxCSP where each value assignment in a constraint corresponds to a non-negative integer or weight instead of 0 for legal and 1 for forbidden in MaxCSP. Two constraint tuple weights are combined with arithmetic plus and the goal of the inference is to find a value assignment of all variables that minimizes the combination of all constraints in the problem. Weighted CSPs can be easily embedded into the semiring-based CBI framework using the semiring $\mathbf{S_{WCSP}} = \langle \mathbb{N}, \min, + \rangle$. Because the multiplicative absorbing element $\alpha_{\otimes}$ of the semiring $\mathbf{S_{WCSP}}$ is equal to the additive identity element $\mathbf{0}$ that is equal to $+\infty$, $\mathbf{S_{WCSP}}$ is eliminative. Also we can show that $\mathbf{S_{WCSP}}$ is monotonic, so both the exact and approximate local consistency enforcing schemes in this paper apply to Weighted CSPs.

We study a random binary Weighted CSP with 100 variables and 200 constraints. The domain of each variable consists of 5 values. We choose randomly a weight from 0 to 10 for each value assignment of every constraint. We construct junction graphs through restricting the maximum cluster size from 2 to 4. Then we apply SCC, DCC and NCC enforcing algorithms to preprocess this Weighted CSP with various ϵ that approximate the multiplicative absorbing element $\alpha_{\otimes} = \infty$. The efficiency of the preprocessing algorithms is characterized as the average variable domain size. We show the experimental results in Table 2. For the purpose of comparison, we normalize the local constraint at each cluster before performing invalid value detection. Given these experimental results, we conclude: (1) For all of these approximate local cluster consistency enforcing algorithms, the closer ϵ is to the exact multiplicative absorbing element $\alpha_{\otimes} = \infty$, the fewer domain values are eliminated during the preprocessing. (2) The preprocessing time for each local cluster consistency enforcing algorithm is affected by the structure of the junction graph, but it does not change monotonically with the maximal cluster size. (3) In sequential computing schemes, SCC uses the least preprocessing time, followed by DCC and then NCC. The time used by DCC or NCC can be reduced if parallel computing is introduced. (4) DCC has the strongest preprocessing ability due to its "global" property. In other words, message passing from lower order clusters contains information from clusters that are lower than them. NCC with one step message updating is slightly better than SCC in that a cluster in NCC collects information from all its immediate neighbors. If we perform several steps (3 in our experiments) of message updating, more values are removed.

5.2 Probability Assessment

Probability inference problems can be seen as constraint-based inference by treating conditional probability distributions (CPDs) as soft constraints over variables. A Bayesian network (BN) [20] is a graphical representation for probability inference under conditions of uncertainty. BN is defined as a directed acyclic graph (DAG) where vertices $\mathbf{X} = \{X_1, \cdots, X_n\}$ denote *n* random variables and directed edges denote causal influences between variables. $\mathbf{D} = \{D_1, \cdots, D_n\}$ is a collection of finite domains for the variables. A set of conditional probability distributions $\mathbf{F} = \{f_1, \cdots, f_n\}$, where $f_i = P(X_i|Parents(X_i))$ is attached to each variable (vertex) X_i. The probability distribution over all the variables is given by the

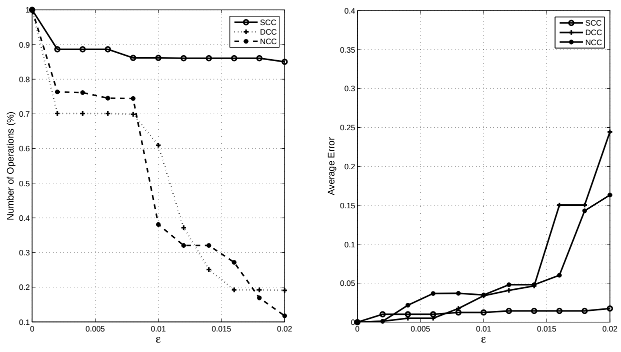

Figure 6. (a) Number of operations: the number of binary operations required for probability assessment after using the local cluster consistency enforcing algorithms (shown as a fraction of the number required without preprocessing) as a function of ϵ and (b) Average error: the resultant average error of the marginal probability for the Insurance network as a function of ϵ

combination of the CPDs using the arithmetic product. The probability assessment problem computes the posterior marginal probability of a subset of variables, given values for some variables as known evidence. The probability assessment problem can be represented as a CBI problem using the commutative semiring $\mathbf{S_{PROB}} = \langle \mathbb{R}^+ \cup \{0\}, +, \times \rangle$. It is easy to show that $\alpha_{\otimes} = \mathbf{0} = 0$ in $\mathbf{S_{PROB}}$ and $\mathbf{S_{PROB}}$ is monotonic that both the exact and approximate local consistency enforcing schemes in this paper apply to probability assessment problems.

The Bayesian network used here is the Insurance network from the Bayesian Network Repository [9]. The network has 27 variables and 27 non-binary constraints (CPDs). In our experiments, we randomly choose one variable as observed. The approximate local cluster consistency enforcing algorithms are used to preprocess the problem based on a junction graph representation with the maximal cluster size of 5. The junction tree algorithm in Lauritzen-Spiegelhalter architecture [15] is used to infer the marginal probability of every unobserved variable. We compare the number of binary operations required for probability assessment after using the preprocessing algorithms (shown as a fraction of the number required without preprocessing) and the resultant error of the marginal probability for the Insurance network as a function of ϵ in Figure 6. At each value of ϵ, we collect data for 5 runs. It is clear that ϵ controls the tradeoff of the precision and the speed of the inference. DCC and NCC have stronger ability to speed up the inference but introduce more errors than SCC, so they are more sensitive to the selection of the approximation threshold ϵ.

6 Conclusion

As the first contribution of this paper we propose a family of novel generalized local consistency concepts for the junction graph representation of CBI problems. These concepts apply to a broader coverage of inference problems from various fields based only on the general condition that depends on the properties of semirings that are used to abstract these problems. Second, we present several local consistency enforcing algorithms, including single, directional and neighborhood cluster consistency enforcing algorithms and their corresponding approximation variants. Third, theoretical complexities

Table 2. Average variable domain sizes for a Weighted CSP that is preprocessed by three approximate local cluster consistency enforcing algorithms with different junction graph representations and different approximate element ϵ. Here SCC stands for Single Cluster Consistency; DCC stands for Directional Cluster Consistency; NCC-n stands for Neighborhood-Cluster Consistency with n steps of message updating. k is the size of the maximal cluster in a junction graph $\mathcal{J} = (\mathcal{C}, \mathcal{S})$. The original average domain size is 5 and we normalize the local constraint at each cluster for comparison.

	$k = 2$				$k = 3$				$k = 4$			
$\lvert\mathcal{C}\rvert$	200				107				69			
$\lvert\mathcal{S}\rvert$	1500				772				520			
Max Degree	26				24				26			
Algorithm	SCC	DCC	NCC-1	NCC-3	SCC	DCC	NCC-1	NCC-3	SCC	DCC	NCC-1	NCC-3
$\epsilon = 7$	4.84	2.40	3.17	1.38	4.70	2.69	2.90	1.44	4.41	2.94	3.02	1.50
$\epsilon = 10$	5.00	2.88	4.21	1.58	4.97	3.36	3.97	1.65	4.94	3.51	4.05	1.87
$\epsilon = 15$	5.00	3.35	4.85	1.87	5.00	4.05	4.79	2.02	4.98	4.22	4.76	2.28
$\epsilon = 25$	5.00	3.97	5.00	2.36	5.00	4.44	4.96	2.61	5.00	4.65	4.97	2.88
$\epsilon = 50$	5.00	4.32	5.00	3.09	5.00	4.88	5.00	3.39	5.00	4.91	5.00	3.75
$\epsilon = 75$	5.00	4.55	5.00	3.53	5.00	4.94	5.00	3.93	5.00	4.97	5.00	4.22
$\epsilon = 100$	5.00	4.66	5.00	3.88	5.00	4.97	5.00	4.22	5.00	4.98	5.00	4.46
$\epsilon = 125$	5.00	4.77	5.00	4.08	5.00	4.98	5.00	4.36	5.00	4.99	5.00	4.59
$\epsilon = 150$	5.00	4.82	5.00	4.24	5.00	4.99	5.00	4.53	5.00	5.00	5.00	4.66
$\epsilon = 200$	5.00	4.90	5.00	4.46	5.00	5.00	5.00	4.65	5.00	5.00	5.00	4.75
$\epsilon = 500$	5.00	4.94	5.00	4.89	5.00	5.00	5.00	4.99	5.00	5.00	5.00	4.92
Avg. Time (s)	0.16	16.25	42.88	180.65	2.62	12.72	30.86	137.48	3.55	16.25	43.85	234.80

of these preprocessing or consistency enforcing algorithms are discussed and experimental results of applying them to both MaxCSP and probability assessment problems are given. Finally, we discuss the relationship between these local cluster consistency concepts and message passing schemes such as junction tree algorithms and loopy message propagation. We intend to study efficient approaches to combining local cluster consistency enforcing with message propagation for general CBI problems in future work.

ACKNOWLEDGEMENTS

We thank Dr. Juerg Kohlas for his comments on this paper. This research was supported by NSERC and IRIS. Le Chang is a Precarn Scholar and Alan K. Mackworth holds a Canada Research Chair.

REFERENCES

[1] Srinivas M. Aji and Robert J. McEliece, ‘The generalized distributive law’, *IEEE Transactions on Information Theory*, **46**, 325–343, (2000).

[2] Christian Bessière and Jean-Charles Régin, ‘Enforcing arc consistency on global constraints by solving subproblems on the fly.’, in *CP*, pp. 103–117, (1999).

[3] Stefano Bistarelli, *Semirings for Soft Constraint Solving and Programming*, Springer-Verlag, 2004.

[4] Stefano Bistarelli, Ugo Montanari, and Francesca Rossi, ‘Semiring-based constraint satisfaction and optimization’, *J. ACM*, **44**(2), 201–236, (1997).

[5] Le Chang, *Generalized Constraint-Based Inference*, Master’s thesis, Dept. of Computer Science, Univ. of British Columbia, 2005.

[6] Le Chang and Alan K. Mackworth, ‘A generalization of generalized arc consistency: From constraint satisfaction to constraint-based inference’, in *IJCAI05 Workshop on Modelling and Solving Problems with Constraints*, pp. 68–75, Edinburgh, (July 2005).

[7] Martin Cooper and Thomas Schiex, ‘Arc consistency for soft constraints’, *Artificial Intelligence*, **154**(1-2), 199–227, (2004).

[8] Rina Dechter and Judea Pearl, ‘Tree clustering for constraint networks’, *Artif. Intell.*, **38**(3), 353–366, (1989).

[9] N. Friedman, M. Goldszmidt, D. Heckerman, and S. Russell. Bayesian network repository, http://www.cs.huji.ac.il/labs/compbio/repository/.

[10] Kalev Kask, Rina Dechter, Javier Larrosa, and Avi Dechter, ‘Unifying cluster-tree decompositions for reasoning in graphical models’, *Artificial Intelligence*, **166**, 165–193, (August 2005).

[11] J. Kohlas and P.P. Shenoy, ‘Computation in valuation algebras’, in *Handbook of Defeasible Reasoning and Uncertainty Management Systems, Volume 5: Algorithms for Uncertainty and Defeasible Reasoning*, 5–40, Kluwer, Dordrecht, (2000).

[12] J. Kohlas and N. Wilson, ‘Exact and approximate local computation in semiring induced valuation algebras’, Technical Report 06-06, Department of Informatics, University of Fribourg, (2006).

[13] Javier Larrosa, ‘Node and arc consistency in weighted CSP’, in *Proc. of AAAI02*, pp. 48–53, Menlo Park, CA, USA, (2002).

[14] Javier Larrosa and Thomas Schiex, ‘In the quest of the best form of local consistency for weighted CSP’, in *Proc. of IJCAI-03*, pp. 239–244, Acapulco, Mexico, (2003).

[15] S. L. Lauritzen and D. J. Spiegelhalter, ‘Local computations with probabilities on graphical structures and their application to expert systems’, *Journal of the Royal Statistical Society, Series B*, **50**, 157–224, (1988).

[16] Alan K. Mackworth., ‘Consistency in networks of relations’, *Artificial Intelligence*, **8**, 99–118, (1977).

[17] Alan K. Mackworth, ‘On reading sketch maps.’, in *IJCAI77*, pp. 598–606, (1977).

[18] R. Mohr and T.C. Henderson, ‘Arc and path consistency revisited’, *Artificial Intelligence*, **28**(2), 225–233, (1986).

[19] Kevin P. Murphy, Yair Weiss, and Michael I. Jordan, ‘Loopy belief propagation for approximate inference: An empirical study’, in *UAI99*, pp. 467–475, (1999).

[20] Judea Pearl, *Probabilistic reasoning in intelligent systems: networks of plausible inference*, Morgan Kaufmann Publishers Inc., 1988.

[21] Cedric Pralet, Gerard Verfaillie, and Thomas Schiex, ‘Composite graphical models for reasoning about uncertainties, feasibilities, and utilities’, in *CP05 Workshop on Preferences and Soft Constraints*, (2005).

[22] Thomas Schiex, Hélène Fargier, and Gerard Verfaillie, ‘Valued constraint satisfaction problems: Hard and easy problems’, in *IJCAI95*, pp. 631–637, Montreal, (1995).

[23] P. P. Shenoy and G. Shafer, ‘Axioms for probability and belief-function propagation’, in *UAI90*, 169–198, (1990).

[24] Gerard Verfaillie, David Martinez, and Christian Bessière, ‘A generic customizable framework for inverse local consistency’, in *AAAI ’99*, pp. 169–174, Menlo Park, CA, USA, (1999).

[25] Richard Wallace, ‘Directed arc consistency preprocessing as a strategy for maximal constraint satisfaction’, in *Proc. of ECAI’94 Workshop on Constraint Processing*, Amsterdam, (1994).

[26] Ying Zhang and Alan K. Mackworth, ‘Parallel and distributed finite constraint satisfaction: Complexity, algorithms and experiments’, in *Parallel Processing for Artificial Intelligence*, chapter 1, 305–334, Elsevier Science Publishers, (1994).