Generalized Constraint-Based Inference

Le Chang and Alan K. Mackworth

Department of Computer Science, University of British Columbia
2366 Main Mall, Vancouver, B.C. Canada V6T 1Z4
{lechang, mack}@cs.ubc.ca

Abstract. Constraint-Based Inference (CBI) is a unified framework that subsumes many practical problems in different research communities. These problems include probabilistic inference, decision-making under uncertainty, constraint satisfaction, propositional satisfiability, decoding problems, and possibility inference. Recently, researchers have presented various unified representation and algorithmic frameworks for CBI problems in their fields, based on the increasing awareness that these problems share common features in representation and essentially identical inference approaches. As the first contribution of this paper, we explicitly use the semiring concept to generalize various CBI problems into a single formal representation framework that provides a broader coverage of the problem space based on the synthesis of existing generalized frameworks. Second, the proposed semiring-based unified framework is also a single formal algorithmic framework that provides a broader coverage of both exact and approximate inference algorithms, including variable elimination, junction tree, and loopy message propagation methods. Third, we discuss inference algorithm design and complexity issues. Finally, we present a software toolkit named the Generalized Constraint-Based Inference Toolkit in Java (GCBIJ) as the last contribution of this paper. GCBIJ is the first concrete software toolkit that implements the abstract semiring approach to unify the CBI problem representations and the inference algorithms. The discussion and the experimental results based on GCBIJ show that the generalized CBI framework is a useful tool for both research and applications.

1 Introduction

Constraint-Based Inference (CBI) is a general term covering many different problems in several research communities. It consists of discovering new constraints from a set of given constraints over individual entities. These new constraints reveal previously undiscovered properties of those entities. Practical problems from many different fields can be seen as constraint-based inference problems, including probabilistic inference, decision-making under uncertainty, constraint satisfaction problems, propositional satisfiability, decoding problems, and possibility inference.

Along with the development of inference approaches for concrete CBI problems, researchers are increasingly aware that these problems share common abstract representation features. Many inference algorithms, described differently, implicitly have essentially identical ideas underlying them. Understanding the common features and characteristics of CBI representations helps research communities exchange ideas and design more efficient inference algorithms.

The purpose of this paper is to use the algebraic semiring concept to generalize a wide range of CBI problems and different exact and approximate inference algorithms into a single formal framework based on the synthesis of existing generalized frameworks. We aim to analyze the common characteristics of inference algorithms based on the abstract problem representation, then apply the general knowledge learned from the unified framework to improve the concrete inference algorithm design in specific application domains.

To demonstrate the representation power of the proposed semiring-based unified framework, a toolkit named Generalized Constraint-Based Inference Toolkit in Java (GCBIJ) is implemented. It can be used to verify, test, compare, and analyze generalized approaches for CBI problems. The flexible architecture of GCBIJ enables implemented generalized inference algorithms to be applied to solve concrete problems simply through specifying different semirings. GCBIJ's extensibility enables users to design their own task-specific semirings and use available generalized inference algorithms. This not only demonstrates the feasibility of using semirings to unify the CBI problem representations and the various inference algorithms, but also shows that GCBIJ is a suitable platform for both research and practical problem-solving.

2 Related Work

Generalized representation and inference algorithms for CBI problems have been studied for the past ten years. All of these studies are based on the following observation: there are two essential operations in constraint-based inference: (1) **combination**, which corresponds to an aggregation of constraints, and (2) **marginalization**, which corresponds to the projection of a specified constraint onto a narrower domain. Different generalized representations express these two operations through various tools or notions. Given the distributivity property of the two operations, generalized inference algorithms can be abstracted that exploit that property.

2.1 Generalized CBI Representations

Semiring-based CSP (SCSP) [1] was the first generalized framework for soft constraint programming, which includes max CSPs, fuzzy CSPs, weighted CSPs, and probabilistic CSPs. It generalizes these soft constraint programming problems, as well as classic CSPs, into a single abstract framework. The c-semiring, a commutative semiring with the additive idempotency property, is the key notion in SCSP that represents both the combination and marginalization operations. Given that the CSP is an instance of the generalized CBI problem, the SCSP framework inspired us to propose the semiring-based unified framework in this paper. In addition to the generalized representation of hard and soft CSPs, SCSP also generalizes the soft local consistency approaches. Extended from local consistency techniques for classic CSPs, generalized soft local consistency can be instantiated to solve a wide range of soft constraint programming problems. The success of the SCSP framework [2] demonstrates that a generalized problem representation will provide opportunities to migrate an existing approach for one problem to solve another problem, if the two problems have the same abstract representation. That is our motivation for proposing a semiring-based unified framework to represent CBI problems from other disciplines as well as constraint programming.

The valuation algebra framework [3] is an abstract framework for describing probabilistic inference problems in expert systems. It is inspired by the formulation of simple axioms that solve global problems through local computing. Knowledge or information in the framework is called valuation. Inferences are made by computing the marginal of a combination of several valuations. The framework of the valuation algebra is abstracted to include many different formalisms. Bayesian networks, Dempster-Shafer belief function models, Spohn's epistemic belief models, and possibility theory are shown [3] to fit into the valuation algebra framework. The authors also suggested that constraint satisfaction problems, propositional logic, and discrete optimization problems could be expressed in the valuation algebra framework.

2.2 Generalized Inference Algorithms

Many algorithms have been proposed for probabilistic inference. The variable elimination algorithm [4] and the junction tree algorithms [5–7] were among the first inference algorithms for probability inference in Bayesian networks. Originating in dynamic programming, the variable elimination or bucket elimination algorithms explicitly use the distributivity of arithmetic additive and multiplicative operations. The generalized bucket elimination algorithm [8] can be used in belief assessment, most probable explanation, maximum a posteriori hypothesis, and maximum expected utility problems. The bucket elimination approach was also applied in constraint satisfaction problems. It was proved that the generalized bucket elimination algorithm is applicable to both probabilistic and deterministic inference [9].

The Generalized Distributive Law (GDL) [10] is a general message-passing algorithm, a synthesis of work in the information theory, signal processing, statistics, and artificial intelligence communities. GDL generalizes the computational problems as "Marginalize a Product Function" (MPF) problems through using commutative semirings to represent the combination and marginalization operations. GDL can be seen, in the AI community, as a junction tree algorithm. As a semiring-based generalized inference algorithm, GDL can represent a wide range of inference algorithms.

More recently, a unified algorithmic framework was proposed [11] to represent tree-decomposition algorithms as a generalized bucket-tree elimination algorithm for automated reasoning tasks. Different from GDL, the generalized bucket-tree elimination does not rely on semirings. Abstract concepts are used to represent combination and marginalization operations, though the distributivity is also the key idea behind the algorithm. Using those concepts, the message-passing within the tree structure can be abstractly represented. The authors also suggested using their generalized algorithmic framework for constraint satisfaction problems.

3 Background

3.1 Graphical Representations of a CBI Problem

In this paper, a constraint-based inference problem is characterized in terms of a set of variables with values in a finite domain and a set of constraints on these variables. The inference task for a CBI problem is then to make explicit new constraints over some variables, based on the given constraints.

We can use graphs to represent CBI problems. The graphical representations of CBI problems provide opportunities for researchers to design efficient inference approaches.

Hypergraph Representations The most straightforward graphical representation of a CBI problem is the hypergraph representation, where nodes represent variables and hyperarcs represent constraints.

Primal Graph Representations Although the hyper-graph representation is straightforward, it is hard to represent practical CBI problems using hyperarcs. In a primal graph the hyperarcs in a hypergraph are replaced by cliques. The primal graph representation of a CBI problem is an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{V_1, \cdots, V_n\}$ is a set of vertices, each vertex V_i corresponding to a variable X_i of the CBI problem; and $\mathcal{E} = \{(V_i, V_j) | V_i, V_j \in \mathcal{V}\}$ is a set of edges among vertices. There exists an edge (V_i, V_j) if and only if the corresponding variables X_i and X_j appear in the scope of the same constraint.

Junction Tree Representations Sometimes the primal graph of a CBI problem is re-organized as a secondary structure to achieve better computational efficiency. The junction tree is a widely used secondary structure in graphical models. A junction tree is an undirected graph $\mathcal{T} = (\mathcal{C}, \mathcal{S})$. $\mathcal{C} = \{C_1, \cdots, C_n\}$ is a set of clusters, where each cluster C_i corresponds to an aggregation of a subset of vertices $V_{C_i} \subseteq \mathcal{V}$ in the primal graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. $\mathcal{S} = \{S_{ij}, \cdot, S_{lm}\}$ is a set of separators between clusters, where S_{ij} is the separator of clusters C_i and C_j, corresponding to the vertices of $V_{C_i} \cap V_{C_j}$. In addition, the following junction tree properties have to be satisfied:

1. *Singly connected property*: $\mathcal{T} = (\mathcal{C}, \mathcal{S})$ is a tree;
2. *Running intersection property*: $\forall C_i, C_j \in \mathcal{C}$, $V_{C_i} \cap V_{C_j} \subseteq V_{C_k}$ holds for any cluster C_k on the path between C_i and C_j;
3. *Constraint allocation property*: For any constraint f of the CBI problem, $\exists C_i \in \mathcal{C}$ s.t. $\mathit{Scope}(f) \subseteq C_i$.

Factor Graph Representations The factor graph [12] is another graphical representation for CBI problems, widely used in information theory research. A factor graph is a bipartite graph that expresses the structure of the factorization. A factor graph has a variable node for each variable, a factor node for each constraint, and an edge connecting a variable node to a factor node if and only if the variable is in the scope of the constraint.

3.2 Semiring and Commutative Semiring

Definition 1 (Semiring). *Let S be a set. Let $\oplus$ and $\otimes$ be two closed binary operations on S. Here we assume operation $\otimes$ has precedence over operation $\oplus$. $(S, \oplus, \otimes)$ is a semiring if the operations satisfy the following axioms:*

- *Additive associativity: $\forall a, b, c \in S$, $(a \oplus b) \oplus c = a \oplus (b \oplus c)$;*
- *Additive commutativity: $\forall a, b \in S$, $a \oplus b = b \oplus a$;*
- *Multiplicative associativity: $\forall a, b, c \in S$, $(a \otimes b) \otimes c = a \otimes (b \otimes c)$;*
- *Left and right distributivity: $\forall a, b, c \in S$, $a \otimes (b \oplus c) = a \otimes b \oplus a \otimes c$ and $(b \oplus c) \otimes a = b \otimes a \oplus c \otimes a$.*

A semiring $(S, \oplus, \otimes)$ is a ring that need not have either additive ($\oplus$) inverses or an additive identity element. In other words, a semiring is a commutative semigroup under addition and a semigroup under multiplication.

Definition 2 (Commutative Semiring). *A commutative semiring is a semiring that satisfies the following additional conditions:*

- *Multiplicative commutativity: $\forall a, b \in S$, $a \otimes b = b \otimes a$;*
- *Multiplicative identity: there exists a multiplicative identity element $\mathbf{1} \in S$, such that $a \otimes \mathbf{1} = \mathbf{1} \otimes a = a$ for any $a \in S$;*
- *Additive identity: there exists an additive identity element $\mathbf{0} \in S$, such that $a \oplus \mathbf{0} = \mathbf{0} \oplus a = a$ for any $a \in S$.*

A commutative semiring $(S, \oplus, \otimes)$ is a commutative monoid under both addition and multiplication. We say that $\oplus$ is idempotent if $a \oplus a = a$ and $\otimes$ is idempotent if $a \otimes a = a$ for any $a \in S$. Furthermore, we say a commutative semiring $(S, \oplus, \otimes)$ is additive invertible if $\forall a \in S$, $\exists {-a} \in S$, s.t. $a \oplus -a = \mathbf{0}$ and $(S, \oplus, \otimes)$ is multiplicative invertible if $\forall a \in S$, $\exists a^{-1} \in S$, s.t. $a \otimes a^{-1} = \mathbf{1}$.

Here we define ***k*-semiring** as a generalization of semiring with a *1*-semiring corresponding to the definition of a semiring.

Definition 3 ((Commutative) *k*-Semiring). *A (commutative)* k*-semiring is a tuple $(S, op_0, op_1, \cdots, op_k)$ such that for each $i \in \{1, \cdots, k\}$, (S, op_j, op_i) is a (commutative) semiring for every $j < i$.*

The definition of (commutative) k-semiring is sometimes too strong in practice. For example, we may need only (S, op_{i-1}, op_i) to satisfy the (commutative) semiring properties. We do not necessarily need (S, op_j, op_i) to be a (commutative) semiring for $\forall j < i$ always, although the strong definition provides computational flexibility in algorithm designs. $([0, \infty), \max, +, \times)$ is an example of a commutative *2*-semiring.

4 A Semiring-Based Generalized Framework for CBI

4.1 A Semiring-Based Generalized Framework

A constraint-based inference problem is defined in terms of a set of variables with values in finite domains and a set of constraints on these variables. We use commutative semirings to unify the representation of constraint-based inference problems from various disciplines. Formally:

Definition 4. *(Constraint-Based Inference (CBI) Problem) A constraint-based inference (CBI) problem* $\mathbf{P}$ *is a tuple* (X, D, R, F) *where:*

- $X = \{X_1, \cdots, X_n\}$ *is a set of variables;*
- $D = \{D_1, \cdots, D_n\}$ *is a collection of finite domains, one for each variable;*
- $R = (S, \oplus, \otimes)$ *is a commutative semiring;*
- $F = \{f_1, \cdots, f_r\}$ *is a set of constraints. Each constraint is a function that maps value assignments of a subset of variables, its scope, to values in* $\mathbf{S}$*.*

Before defining various tasks for CBI problems, we define the two basic constraint operations as follows, using the two binary operation $\otimes$ and $\oplus$ of the commutative semiring R. Please note that we use the same two symbols $\otimes$ and $\oplus$ to represent the two constraint level operations. The meaning of them can be easily distinguished given the context.

Definition 5. *(The Combination of Two Constraints) The combination of two constraints* f_1 *and* f_2 *is a new constraint* $g = f_1 \otimes f_2$*, where* $Scope(g) = Scope(f_1) \cup Scope(f_2)$ *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 the scope of the constraint* g*.*

Definition 6. *(The Marginalization of a Constraint) The marginalization of* X_i *from a constraint* f*, where* $X_i \in Scope(f)$*, is a new constraint* $g = \bigoplus_{X_i} f$*, where* $Scope(g) = Scope(f) \setminus X_i$ *and* $g(\mathbf{w}) = \bigoplus_{x_i \in Domain(X_i)} f(x_i, \mathbf{w})$ *for every value assignment* $\mathbf{w}$ *of variables in the scope of the constraint* g*.*

Definition 7. *(The Inference Task for a CBI problem) Given a variable subset of interest* $Z \subseteq X$*, let* $Y = X \setminus Z$*, then the inference task for a CBI problem* $\mathbf{P} = (X, D, R, F)$ *is defined as computing:*

$$g_{CBI}(Z) = \bigoplus_{Y} \bigotimes_{f \in F} f \tag{1}$$

Given a CBI problem $\mathbf{P} = (X, D, R, F)$, if $\oplus$ is idempotent, we can define the assignment task for a CBI problem.

Definition 8. *(The Assignment Task for a CBI problem) Given a variable subset of interest* $Z \subseteq X$*, let* $Y = X \setminus Z$*, the assignment task for a CBI problem* $\mathbf{P} = (X, D, R, F)$ *is to find a value assignment for the marginalized variables* Y*,* *which gives the result of the corresponding inference task $g_{CBI}(Z)$. Formally, a solution to the assignment task is:*

$$\mathbf{y} = \arg \bigoplus_{Y} \bigotimes_{f \in F} f \tag{2}$$

where arg *is a prefix of operation $\oplus$. In other words,* $\arg \oplus_x f(x)$ *returns $x_\oplus$ s.t. $f(x_\oplus) = \oplus_x f(x)$.*

If a total ordering of S exists, we can define the optimization task for a CBI problem as maximizing (or minimizing) the computed result of the corresponding CBI inference task by finding a value assignment to variables Z. Formally :

Definition 9. *(The Optimization Task for a CBI problem) Given a variable subset of interest $Z \subseteq X$, let $Y = X \backslash Z$ and $R = (S, max, \oplus, \otimes)$ be a commutative* 2*-semiring, then the optimization task for a CBI problem $\mathbf{P} = (X, D, R, F)$ is defined as computing:*

$$g_{OPT} = \max_{Z} \left(\bigoplus_{Y} \bigotimes_{f \in F} f \right) \tag{3}$$

The assignment task for an optimization task is then to compute:

$$\mathbf{z} = \arg \max_{Z} \left(\bigoplus_{Y} \bigotimes_{f \in F} f \right) \tag{4}$$

In general, $\otimes$ is a combination operation for CBI problems that combines a set of constraints into a constraint with a larger scope; $\oplus_Y = \oplus_{X \backslash Z}$ is a marginalization operation that projects a constraint over the scope X onto its subset Z, through enumerating all possible value assignments of $Y = X \setminus Z$.

Practically, sometimes there are observations of, or evidence about, the values of some variables. In such cases, constraints with observed variables in their scopes can be modified through instantiation before performing the inference. Although there are many observation-handling techniques to accelerate the inference process, we will not incorporate them into our framework here. Another issue concerning practical CBI tasks is the normalization after the computation. In this paper, we omit normalization constants since they are simple to define and compute in specific application scenarios.

4.2 Instances of the Generalized CBI Framework

Many CBI problems from different disciplines can be embedded into the proposed generalized CBI framework. In this section we will briefly introduce these problems as instances of the framework.

Constraint Satisfaction Problems A constraint satisfaction problem (CSP) is defined over a set of variables of discrete domains and a set of constraints or evaluation functions, where each constraint is defined on a subset of variables. The inference task for a CSP is to find if there exists a value assignment that satisfies all the constraints. The assignment task for a CSP is to find such a value assignment or all such assignments. We use the commutative semiring $R = (\{false, true\}, \vee, \wedge)$ to represent a CSP in the framework.

Soft CSPs are used to model over-constrained problems by introducing soft constraints that may be (partially) violated. Following the results of the Semiring CSP [1] and Valued CSP [13] frameworks, various soft CSPs including Max CSPs, Weighted CSPs, Probability CSPs, and Fuzzy CSPs can also be embedded into our framework [14].

Propositional Satisfiability Problems The propositional satisfiability problem (SAT) is a central problem in logic and artificial intelligence, which consists of a logical propositional formula in n variables. Analogous to the CSP, both SAT and MaxSAT can be embedded into our framework by specifying the commutative semirings $R = (\{false, true\}, \vee, \wedge)$ and $R = ([0, \infty), \max, +)$, respectively.

Probability Inference Problems Probability inference problems can be seen as constraint-based inferences by treating conditional probability distributions (CPDs) as soft constraints over variables. Generally, there are three types of tasks in probability inference: probability assessment, most probable explanation, and maximum a posteriori hypothesis (maximum likelihood). All of these tasks can be generalized into our semiring-based framework, by specifying the commutative (*2*-)semirings R as $([0, \infty), +, \times)$, $([0, \infty), \max, \times)$, and $([0, \infty), \max, +, \times)$, respectively.

Dynamic Bayesian Networks Many discrete-time stochastic processes can be graphically represented as dynamic Bayesian networks (DBN) [15]. As an extension of Bayesian networks (BN), inference tasks can be performed over a DBN. We can abstract DBNs into our framework using the same commutative semiring $R = ([0, \infty), +, \times)$.

Decision-Making Problems The Decision Network (DN) or the influence diagram [16], is a graphical representation for decision-making problems. A decision network is a directed acyclic graph with three types of nodes: random nodes, decision nodes, and value nodes. The goal of one-off decision-making problems in decision networks is to find the policies for each decision node that maximize the summation of expected utilities. A decision-making problem can be embedded into our framework using a commutative *2*-semiring $R = ([0, \infty), \max, +, \times)$. We do not integrate sequential decision-making problems [17] in decision networks into our framework and leave it in our future work.

Possibility Inference Problems Possibility theory [18] is another approach to characterizing uncertainties. Each constraint in a possibility inference problem specifies the degree of possibility of the configuration of variables. The degree of possibility, or possibility potential, is a value in $[0, 1]$, representing non-possibility to full certainty. In possibility theory, a binary operation T, also called a t-norm, is a function $\mathrm{T} : [0, 1] \times [0, 1] \rightarrow [0, 1]$, which satisfies the boundary, monotonicity, and commutativity conditions [18]. There are many t-norms in possibility theory research. The most frequently used t-norms are:

- Product t-norm: $a\mathrm{T}b = a \times b$;
- Gödel's t-norm: $a\mathrm{T}b = min(a, b)$;
- Lukasziewcz's t-norm: $a\mathrm{T}b = max(0, a + b - 1)$.

Given a t-norm, a possibility inference problem can be embedded into our framework using a commutative semiring $R = ([0, 1], max, \mathrm{T})$ if the corresponding t-norm satisfies the commutative semiring properties.

Maximum Likelihood Decoding In digital communication, decoding is an important step in recovering the original information transmitted through a discrete memoryless channel. Maximum Likelihood Decoding (MLD), or equivalently minimum log-likelihood, is one widely used approach. A semiring-based general massage-passing approach for MLD was first proposed in [10]. Basically, the commutative semiring $R = ([0, \infty), min, +)$ can be used to abstract the minimum log-likelihood decoding problem.

5 Exact Inference Algorithms

The distributivity and the commutativity properties of a commutative semiring provide computational opportunities for solving constraint-based inference problems more efficiently. Different computation sequences lead to different costs, given the same computation task defined by a commutative semiring. Many inference algorithms have been developed independently, explicitly or implicitly exploiting the distributivity of commutative semirings. The key idea of these algorithms is to rearrange the computation sequences of the task solutions. In this section we generalize these algorithms into two categories: variable elimination algorithms and junction tree algorithms.

5.1 Generalized Variable Elimination Algorithm

The basic idea behind variable elimination (VE) algorithms comes from nonserial dynamic programming [19]. These algorithms work by eliminating variables one by one while computing the effect of each eliminated variable on the remaining problem. See Fig. 1 for the description of our generalized variable elimination algorithm for the CBI inference task, abstracted from the variable elimination algorithm [4] in probability inference.

Input: A CBI problem (X, D, R, F) and a variable subset of interest Z
Output: $g_{CBI}(Z) = \bigoplus_{X-Z} \bigotimes_{f \in F} f$

Let $Y = X \setminus Z$
Choose an elimination ordering $\sigma =< Y_1, \cdots, Y_k >$ of Y
for $i = k$ to 1 **do**
    $F' := \emptyset$
    for each $f \in F$ **do**
        if $Y_i \in Scope(f)$ **then**
            $F' := F' \cup \{f\}$
            $F := F \setminus \{f\}$
        end if
    end for
    $f' := \bigoplus_{Y_i} \bigotimes_{f \in F'} f$
    $F := F \cup \{f'\}$
end for
Return $g_{CBI}(Z) := \bigotimes_{f \in F} f$

Fig. 1. Generalized Variable Elimination Algorithm (GVE) for the CBI Inference Task

A common concern of applying VE algorithms is how to find an optimal elimination ordering to minimize the computation cost. Finding an optimal elimination ordering, which is equivalent to finding an optimal triangulated graph or finding the treewidth, is $\mathcal{NP}$-hard [20]. Several heuristic search methods [21] can be used to generate near-optimal elimination orderings.

For the assignment task for a CBI problem, backtracking approaches can be applied. All intermediate constraints during the elimination procedure are cached. The final computation value of $g_{CBI}(Z)$ is used as an index to track valid value assignments in the intermediate constraints.

To apply VE algorithms, commutativity of $\otimes$ is required; for the CBI inference task that enables us to rearrange the combination sequence of local functions. The commutativity of $\oplus$ is a desired property as well; that enables us to exploit different elimination orderings. Identity elements for the two operations are not required, so we can relax the requirement of commutative semiring here.

An extension of the generalized VE algorithm works with CBI problems defined on a commutative k-semiring $(S, op_0, op_1, \cdots, op_k)$ [14]. In such cases, we repeatedly use the generalized VE for the last two operations, then replace F by the new marginalized constraints. Interestingly, it is another variable elimination process that eliminates the second last operation repeatedly.

Many concrete algorithms from different fields can be seen as instances of variable elimination. For classic CSPs, [22] proposed a variable elimination algorithm in the early 80s. The Adaptive Consistency algorithm [23] is another instance of variable elimination for CSPs. For propositional SAT problems, directional resolution is the core of the well-known Davis-Putnam (DP) algorithm [24]. The variable elimination [4] and the bucket elimination [8] algorithms have been widely studied in tackling inference tasks of probability assessment, MPE, and MAP problems. For decision-making problems, [25] reduced the influence diagrams to Bayesian networks; the variable elimination algorithm for probabilistic inferences is then applied to solve decision-making problems. For maximum likelihood decoding, Viterbi decoding [26] is an instance of variable elimination. We can design other concrete VE algorithms for problems that can be abstractly represented by the semiring-based unified framework, through instantiating different semirings. In general, variable elimination is a variant of dynamic programming, which is the key idea of these algorithms.

According to the complexity analysis given in [14], both the space and time complexities of the generalized VE algorithm are linear in the size of the problem, but exponential in the induced width of an elimination ordering or the width of a tree decomposition of the corresponding primal graph representation. For a large CBI problem with a complex graphical representation, treewidth (the lower bound of widths of all tree decompositions of the problem) is often intractable to compute, which makes the direct application of the VE algorithm infeasible. There are basically two approaches to cope with that: the first one is to seek approximate inference solutions in terms of some criteria; the second is to incorporate value properties of the variables and exploit the structural properties of the problem.

5.2 Generalized Junction Tree Algorithm

The major motivation that prompts researchers to use junction tree (JT) algorithms to solve CBI problems is handling multiple queries. Junction tree algorithms can share intermediate computational results among different queries, which is an advantage relative to variable elimination algorithms. In fact, junction tree algorithms can be seen as memorized dynamic programming [27], where solutions for subproblems are memorized for later use. A junction tree is a structure that efficiently divides the original problem into subproblems.

In general, junction tree algorithms assign constraints to clusters and combine constraint in the same cluster. The combined constraint is marginalized and passed as a message between clusters. Following a specified message-passing scheme, the junction tree reaches consistency and any variable of interest can be queried through marginalizing out other variables in the cluster that contains that variable. Formally, the generalized junction tree algorithm for the inference task of a CBI problem is shown in Fig. 2, which is abstracted from the message-passing scheme in the Shenoy-Shafer architecture [5] of probabilistic inference.

One interesting case of applying JT algorithms is that the variable subset of interest Z in query is not contained in a single cluster. A solution is to find a subtree $\mathcal{T}_Z = (\mathcal{C}_Z, \mathcal{S}_Z)$ of $\mathcal{T} = (\mathcal{C}, \mathcal{S})$, where $Z \subseteq \mathbf{C} = \bigcup_{C \in \mathcal{C}_Z} C$. The answer to the query is computed as the marginal of the combination of all the local constraints together with all the incoming messages of the subtree $\mathcal{T}_Z$. The basic idea here is to treat the subtree $\mathcal{T}_Z$ as a virtual cluster and compute the marginal as treating normal concrete clusters.

We can modify the generalized JT algorithm to solve the assignment task for a CBI problem. After computing $g_{CBI}(Z)$ in some cluster C_i, backtracking is

Input: A junction tree $\mathcal{T} = (\mathcal{C}, \mathcal{S})$ of a CBI problem (X, D, R, F), a variable subset of interest Z

Output: $g_{CBI}(Z) = \bigoplus_{X \setminus Z} \bigotimes_{f \in F} f$

Attach to each cluster C_i a potential $\phi_{C_i} = \mathbf{1}$
for each $f \in F$ **do**
    Find a cluster C_i such that $Scope(f) \subseteq C_i$
    $\phi_{C_i} := \phi_{C_i} \otimes f$
end for
for each Edge S_{ij} which is from clusters C_i to C_j **do**
    if C_i has received messages from all its neighbors other than C_j **then**
        $N_{i-j} := Neighbor(C_i) \setminus \{C_j\}$
        % $m(C_i, C_j)$ is the message sent from C_i to C_j;
        $m(C_i, C_j) := \bigoplus_{C_i \setminus S_{ij}} (\phi_{C_i} \otimes \bigotimes_{C_l \in N_{i-j}} m(C_l, C_i))$
    end if
end for
for each $C_i \in \mathcal{C}$ **do**
    if $Z \subseteq C_i$ **then**
        $\phi_{C_i} := \phi_{C_i} \otimes \bigotimes_{C_l \in Neighbor(C_i)} m(C_l, C_i)$
        Return $g_{CBI}(Z) := \bigoplus_{C_i \setminus Z} \phi_{C_i}$
    end if
end for

Fig. 2. Generalized Junction Tree Algorithm (GJT) for the CBI Inference Task

applied to the variables contained in C_i. After a value assignment for these variables is decided, the (partial) assignment is passed to all the neighboring clusters of C_i. The procedure recursively instantiates values of variables according to the messages passing from the instantiated clusters to the un-instantiated ones.

Following the result of [5], commutativity of both $\oplus$ and $\otimes$ is required, which ensures the correctness and completeness of the JT algorithm. The identity elements of the two operations are required as well. Other special semiring properties, such as the combination (or additive) invertible property and the combination (or additive) idempotency property, are not mandatory, but these properties do enable us designing more efficient message-passing schemes.

A commutative semiring $(R, \oplus, \otimes)$ is idempotent under combination if $\forall a \in R$, $a \otimes a = a$. Since combination corresponds to information (or constraint) gathering, the idempotency of combination implies that repeatedly combining the same information will not produce new information. Furthermore, considering the marginalization of a set of information $m = \bigoplus_i m_i$ in a semiring with the combination idempotency property, we get $m = m \otimes m = m \otimes \bigoplus_i m_i = \bigoplus_i (m \otimes m_i)$. This induction implies $m = \bigoplus_i m_i = \bigoplus_i (m \otimes m_i)$, which means that combining the marginalization with original information will not produce new information after another marginalization. This implies we can ignore the double-counted messages in the generalized junction tree algorithm, without loss of correctness. We abstracted a variant of the generalized JT algorithm for semirings with the combination idempotency property [14], GJT-Idemp, based on the previous observation.

Another variant of the generalized JT algorithm is based on the invertible property of combination. The basic idea of utilizing the invertible property is caching the combination of all the incoming messages and dividing the specific incoming message to get the outgoing message for that separator. Dividing here eliminates duplicated information from the combined message. We abstracted a variant of the generalized JT algorithm for semirings with the combination invertible property, GJT-Inv, based on this observation. Two other variants for semirings with the combination invertible property are GJT-LS and GJT-HUGIN, corresponding to the Lauritzen-Spiegelhalter (LS) architecture [28] and the HUGIN architecture [7] in probabilistic inference, respectively. Details of these algorithms can be found in [14].

Many concrete algorithms from different fields can be seen as instances of the generalized junction tree algorithm. In the probabilistic inference community, junction tree algorithms have been widely studied since the late 1980s. These algorithms include message-passing schemes in Shenoy-Shafer [5], Lauritzen-Spiegelhalter [28], and HUGIN [7] architectures. The application of junction tree algorithms for constraint satisfaction problems can be traced back to [29], which is essentially the same scheme as Shenoy-Shafer architecture, except that the relational join and project operations are used as constraint combination and marginalization operations. In [30], the arc consistency algorithm for the rooted join tree (TAC) was presented to solve CSPs represented by constraint networks. The contribution of their work is proposing and analyzing parallel (PTAC) and distributed (DTAC) message-passing schemes based on the TAC, which can be generalized and applied to junction tree algorithms of other disciplines. The junction tree algorithm was explicitly generalized as the Generalized Distributive Law (GDL) [10]. Many concrete decoding algorithms, including the Baum-Welch algorithm, the Gallager-Tanner-Wiberg decoding algorithm, and the BCJR algorithm, are all special cases of the GDL algorithm.

In general, junction tree algorithms are memorized dynamic programming, which cache solutions for subproblems to answer multiple queries. Both the time and space complexities [14] of junction tree algorithms are polynomial in the size of the junction tree, with a constant factor exponential in the maximum subproblem size. So the key to applying JT algorithms is to divide the original CBI problem into subproblems with tractable sizes. However, many practical CBI problems have large treewidth. As the lower bound of the width of any tree decomposition, a large treewidth makes exact inference in the junction tree infeasible. There are two ways to perform approximate inferences in such a situation: (1) Splitting the oversized clusters, which corresponds to removing some edges from the primal graph representation or retracting some constraints in the original CBI problem; (2) Using junction graphs to perform inference, instead of junction trees. In a junction graph, messages can pass in loops and may not terminate, which means that information may be counted repeatedly. Some criteria are used to terminate the message-passing after reaching preset thresholds. Both of these approaches will be discussed in the following section.

5.3 Complexity

The time and space complexities of the generalized junction tree algorithm, as well as the generalized variable elimination algorithm are discussed in [14]. Both the VE and the JT algorithms are exponential in the induced width of an elimination ordering, or the width of a junction tree representation. In the generalized VE and JT algorithms working on semirings with the combination invertible or idempotency property, the time complexity is linear in the size of a problem, whereas in the generalized JT algorithm, it is quadratic in the size of a problem. The space complexity is linear in the size of a problem, regardless of the properties the semiring has.

The complexity expressions do not convey sufficient information about the running times of these algorithms. The upper bounds of different operations of these algorithms are listed in Table 1, in terms of the number of operations. Here $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and $\mathcal{T} = (\mathcal{C}, \mathcal{S})$ are respectively a primal graph representation and a junction tree representation of the CBI problem; d is the maximum domain size of variables; w is the width of $\mathcal{T}$; r is the number of constraints; and sep is the maximum separator size. The binary operation $\oslash$ is the inverse operation that is defined as: $\forall a, b \in S, a \oslash b \equiv a \otimes b^{-1}$, where $b^{-1} \in S$ is the inverse of b. More specifically, for a CBI problem defined on a semiring with the combination invertible property, generally we conclude: (1) GJT-Inv uses the least time, followed by GJT-HUGIN and then GJT-LS; (2) GJT-LS uses the least space while GJT-HUGIN and GJT-Inv use about the same space.

Table 1. Upper bounds of running times for generalized variable elimination (GVE) and generalized junction tree (GJT) and its variants, in terms of the number of operations.

Algor.	# of Operation $\otimes$	# of Operation $\oplus$	# of Operation $\oslash$
GVE	$(r - \lvert\mathcal{V}\rvert) \cdot d^{w+1}$	$\lvert\mathcal{V}\rvert \cdot d^{w+1}$	0
GJT	$(4 \cdot \lvert\mathcal{C}\rvert^2 - \lvert\mathcal{C}\rvert + r) \cdot d^{w+1}$	$2 \cdot \lvert\mathcal{C}\rvert \cdot (d^{w+1} - d^{sep})$	0
GJT-Idemp	$(\lvert\mathcal{C}\rvert + r) \cdot d^{w+1}$	$2 \cdot \lvert\mathcal{C}\rvert \cdot (d^{w+1} - d^{sep})$	0
GJT-Inv	$(\lvert\mathcal{C}\rvert + r) \cdot d^{w+1}$	$2 \cdot \lvert\mathcal{C}\rvert \cdot (d^{w+1} - d^{sep})$	$\lvert\mathcal{C}\rvert \cdot d^{w+1}$
GJT-LS	$(\lvert\mathcal{C}\rvert + r) \cdot d^{w+1}$	$2 \cdot \lvert\mathcal{C}\rvert \cdot (d^{w+1} - d^{sep})$	$2 \cdot \lvert\mathcal{C}\rvert \cdot d^{w+1}$
GJT-HUGIN	$(\lvert\mathcal{C}\rvert + r) \cdot d^{w+1}$	$2 \cdot \lvert\mathcal{C}\rvert \cdot (d^{w+1} - d^{sep})$	$2 \cdot \lvert\mathcal{C}\rvert \cdot d^{sep}$

6 Approximate Inference Algorithms

The analyses in the previous section show that both the variable elimination and the junction tree algorithms can perform exact inference for a constraint-based inference problem in polynomial time and space. However, there always exist constant factors in the time and space complexities that are exponential in the maximum cluster size of the problem's tree decomposition. This means that the generalized exact inference algorithms would be infeasible when the treewidth of the corresponding problem, a lower bound of width for all possible tree decompositions, is intractable. This practical challenge encourages researchers to develop approximate inference algorithms for CBI problems, if approximate inference results with some quality guarantee are acceptable in their application domains.

The key idea of these approximate inference algorithms is to restrict the size of the maximum subproblem, or equivalently, the maximum cluster size of a tree decomposition or the induced width given an elimination ordering, to an acceptable level. In general, there are at least two possible ways to design an algorithm to achieve this purpose. The first approach is to revise the original CBI problems by removing some less important constraints, which makes the structure of the problem's graphical representation much simpler; another approach does not touch the original CBI problem but re-organizes it into more complex graphical representations, e.g., a junction graph with loops. Inference procedures are carefully re-designed to cope with these graphical representations.

6.1 Algebraic Approximation for VE and JT

The algebraic foundation of some approximate inference algorithms is based on Eq. 5. Here F is a set of constraints and F_i is a subset of constraints for $i = 1, \cdots, b$, where $F_1 \cup \cdots \cup F_b = F$ and $F_i \cap F_j = \emptyset$ for any $i, j \in \{1, \cdots, b\}, i \neq j$. The basic idea here is breaking the original CBI problem into b (overlapped) subproblems, solving them individually, then joining them again to get the solution.

$$\bigoplus_{Y} \bigotimes_{f \in F} f \approx \bigotimes_{i=1}^{b} \Big(\bigoplus_{Y} \bigotimes_{f \in F_i} f \Big) \tag{5}$$

Approximate Variable Elimination Algorithm According to the generalized exact variable elimination algorithm in Fig. 1, the bottleneck of computation occurs when we eliminate a variable X_i, too many constraints with X_i in their scopes have to be combined, which implies a large constraint after combining and marginalizing. The way to overcome this bottleneck is to clone X_i with several identical copies $X_i^{(1)}, \cdots, X_i^{(b)}$. The constraints with X_i in their scopes are revised by replacing X_i with $X_i^{(j)}, j \in \{1, \cdots, b\}$ according to specified rules. Then the VE algorithm can be applied. Of course, introducing identical copies $X_i^{(1)}, \cdots, X_i^{(b)}$ of X_i will introduce conflicts of the X_i values and bring errors in the marginalization.

Min-Buckets [31] is the first algorithm proposed for applying this idea to solve probability inference problems approximately. We generalize the Min-Buckets algorithm as the generalized approximate variable elimination algorithm in our semiring-based unified framework [14].

Approximate Junction Tree Algorithm The generalized junction tree algorithm can be modified to perform approximate inference based on Eq. 5. The basic idea of the approximate JT algorithm is to pass a set of messages from one cluster to another, instead of passing a single message. The combination of these messages is an approximation of the message passed in the exact JT algorithm.

The Min-Clustering Tree [32] is an approximate probability inference algorithm following this idea. We generalize the Min-Clustering Tree algorithm as the generalized approximate junction tree algorithm in our semiring-based unified framework [14].

For semirings with the combination invertible property, the inverse of a message is similarly approximated by the combination of a set of the messages' inverses. The approximation can be formalized as Eq. 6.

$$\mathbf{1} \oslash (\bigoplus_{Y} \bigotimes_{f \in F} f) \approx \bigotimes_{i=1}^{b} \left(\mathbf{1} \oslash (\bigoplus_{Y} \bigotimes_{f \in F_i} f) \right) \tag{6}$$

After applying Eq. 6, we can revise any one of these exact junction tree algorithms for semirings with the combination invertible property to cope with approximate inference tasks [14].

Discussion Algebraic approximation is the foundation of many approximate inference approaches for CBI problems. On the CBI problems level, it is equivalent to retracting some given constraints. On the primal graph representation level, it is equivalent to removing some edges from the graph. On the junction tree representation level, it is equivalent to splitting some clusters in the junction tree. The purpose of these approximations is to restrict the size of the maximum subproblem to a tractable level. Though the idea of algebraic approximation is straightforward, it is hard to analyze the error bounds of these approaches, especially when the combination and marginalization operations are abstract. So far there are few theoretical guidelines for choosing which constraints should be released (which edges should be moved, how to split a cluster); only empirical analyses currently apply in such cases.

6.2 Loopy Message Propagation

The algebraic approximation implies that the original CBI problems are revised and simplified to make computation feasible. Loopy message propagation, by contrast, does not revise the original CBI problems.

As already known, in junction tree algorithms both the time and space complexities are bounded by the maximum cluster size. To maintain junction tree properties, the maximum cluster size is usually large in practical problems. If junction tree properties are relaxed, in other words, if the secondary structure is not necessarily a tree but a graph with loops, the maximum cluster size can be dramatically reduced.

At the same time, the message-passing may not terminate due to the introduction of loops. Also messages will be repeatedly counted. Both of these bring errors of inferences. However, empirical results in probability inference [33] and Turbo decoding [34] show that the same message-passing schemes in exact junction tree algorithms work well in junction graphs with loops. When the junction graph is singly connected, the exact inference result is produced after one iteration of message-passing. When there exists only a single loop in the junction graph, it is guaranteed to converge to the correct result under some conditions [35]. Based on these observations, we generalized the loopy message propagation algorithm in our semiring-based unified framework [14].

The basic idea of the generalized loopy message propagation is the same as message-passing in the generalized junction tree algorithm, except that messages at iteration t are updated by incoming messages at iteration $t-1$. The initial messages are produced based on the local constraints and do not depend on the incoming messages from the neighboring clusters. If a CBI problem is defined on a semiring with the combination invertible property, the algorithm can be revised slightly to save computational cost.

A special case of applying the loopy message propagation is performing inference on CBI problems defined on semirings with the combination idempotency property. Since the idempotency of combination implies that repeatedly counted messages will not introduce errors, the loopy message propagation returns exact answers after finite iterations. Classic CSPs can be embedded into our semiring-based unified framework using the commutative semiring $R = (\{false, true\}, \vee, \wedge)$. It is easy to show that the combination operation $\wedge$, logical AND, is idempotent. Then the loopy message propagation should be an exact inference algorithm for classic CSPs. In practice, Arc Consistency [36], the key technique for CSPs, can be seen as a special case of the generalized loopy message propagation for CBI problems defined on semirings with the combination idempotency property. In Arc Consistency, the messages are the possible values of the variable in a separator. Combining messages from other clusters will remove illegal values, in other words, revise the outgoing message. The algorithm terminates when all the messages remain the same.

6.3 Hybrid Junction Tree Algorithm

The last approach for the approximate inference of CBI problems is the generalized hybrid junction tree algorithm [14]. Generally, it passes messages exactly to or from clusters with tractable sizes. When a cluster with intractable size is encountered, the hybrid junction tree algorithm builds a local junction graph based on the local constraints and incoming messages. In other words, large clusters are treated as subproblems. Loopy message propagation is used in this local junction graph. In addition to the local constraints of a subproblem, all the incoming messages are seen as constraints of the subproblem. All outgoing messages are initially seen as unitary constraints. After performing the loopy message propagation over the junction graph of the subproblem for several iterations, we can approximately produce outgoing messages. These approximate messages then pass to the neighbors clusters, as in the generalized approximate junction tree algorithm.

7 Framework Implementation

We here implemented the proposed semiring-based unified framework for CBI problems as a software toolkit, named the Generalized Constraint-Based Inference Toolkit in Java (GCBIJ) [14]. The toolkit provides a way to represent various concrete CBI problems and a series of generalized exact and approximate inference algorithms. By specifying various semirings, these generalized inference algorithms can be instantiated as concrete algorithms that solve CBI problems from different disciplines. GCBIJ also provides a collection of parsers to translate concrete problems from various fields with domain-specific formats, such as the DIMACS, BIF, XMLBIF, and CSPIF formats.

The architecture of GCBIJ is flexible, making it easy to extend. Users can implement their own task-specified semirings to fulfill their purposes. All the implemented inference algorithms use the abstract semiring class to access basic operations, without relating to the properties of concrete semirings. On the contrary, to design a new inference algorithm, users do not need knowledge of specific semirings, which ensures the generality of the algorithm. Given the common parser interface, users can also design a new parser to translate their domain problems into our internal CBI problem representation. GCBIJ is a concrete system for implementing the ideas and concepts of the proposed unified framework.

We use GCBIJ to do a series of experiments [14] for CBI problems from different fields, including CSP, MaxCSP, SAT, and probability inference. The results of these experiments are not totally new to research communities. However, these results as a whole show that (1) the proposed semiring-based unified framework and generalized inference algorithms are suitable for representing and solving various concrete CBI problems; (2) GCBIJ is a good platform for studying CBI problems; (3) GCBIJ has the potential to tackle practical applications. Although more optimization and implementation work is required, the extendibility and flexibility of GCBIJ make it a suitable toolkit for both CBI research and applications.

8 Conclusion

As the first contribution of this paper, we propose a semiring-based unified framework, a single formal representation framework that provides a broader coverage of the constraint-based inference problem space based on the synthesis of existing generalized frameworks. Our framework explicitly subsumes and unifies many concrete CBI problems, such as probabilistic inference, decision making under uncertainty, constraint satisfaction problems, propositional satisfiability, decoding problems, and possibility inference, in an abstract representation.

The unified framework is also a single formal algorithmic framework that provides a broader coverage of both exact and approximate inference algorithms. This is the second contribution of this paper. We unify various widely used inference algorithms, such as the exact and approximate variable elimination algorithms, the exact and approximate junction tree algorithms, and loopy message propagation, based on the framework. Many of these algorithms depend only on the basic properties of the commutative semirings. Based on other special properties of different commutative semirings, we also generalize the variants of these algorithms that arise in different application scenarios.

Abstract representations of CBI problems, as well as abstract inference algorithms, provide several opportunities for researchers from various fields: (1) they can study the most important common characteristics of various CBI problems without representation barriers; (2) they can analyze and compare different inference approaches; (3) they can borrow design ideas from other fields and improve the inference approaches' efficiency in their own domains; and (4) implementations at the abstract level significantly reduce the amount of work targetted previously at the individual problems. In other words, researchers from different fields may reinterpret many familiar approaches in their domains at a higher level. The algorithm discussions and the complexity analyses in this paper are examples of applying the abstract knowledge to the concrete application domains.

The unified representation for CBI problems and inference algorithms is not, of course, a totally novel idea. Much research has been conducted in various disciplines through using different tools or notions. Here we have significantly broadened the scope of the problems and the coverage of the algorithms. The final contribution of this paper is a software toolkit, the Generalized Constraint-Base Inference Toolkit in Java (GCBIJ). GCBIJ is the first concrete toolkit to implement ideas for unifying the representations of CBI problems using semirings and to implement various inference algorithms on the abstract level. The generalization and extensibility of GCBIJ make it a good platform for both CBI research and practical problem solving.

References

1. Bistarelli, S., Montanari, U., Rossi, F.: Semiring-based constraint satisfaction and optimization. J. ACM **44** (1997) 201–236
2. Bistarelli, S.: Semirings for Soft Constraint Solving and Programming. Springer-Verlag (2004)
3. Kohlas, J., Shenoy, P.: Computation in valuation algebras. In: Handbook of Defeasible Reasoning and Uncertainty Management Systems, Volume 5: Algorithms for Uncertainty and Defeasible Reasoning. Kluwer, Dordrecht (2000) 5–40
4. Zhang, N.L., Poole, D.: A simple approach to Bayesian network computations. In: Proc. of the Tenth Canadian Conference on Artificial Intelligence. (1994) 171–178
5. Shenoy, P.P., Shafer, G.: Axioms for probability and belief-function propagation. In: Uncertainty in Artificial Intelligence 4. North-Holland (1990) 169–198

6. Lauritzen, S.L., Spiegelhalter, D.J.: Local computations with probabilities on graphical structures and their application to expert systems. Journal of the Royal Statistical Society, Series B **50** (1988) 157–224
7. Jensen, F.V., Lauritzen, S.L., Olesen, K.G.: Bayesian updating in causal probabilistic networks by local computations. Computational Statistics Quarterly **4** (1990) 269–282
8. Dechter, R.: Bucket elimination: A unifying framework for probabilistic inference. In: 12th Conf. on Uncertainty in Artificial Intelligence. (1996) 211–219
9. Dechter, R.: Constraint Processing. Morgan Kaufmann (2003)
10. Aji, S.M., McEliece, R.J.: The generalized distributive law. IEEE Transactions on Information Theory **46** (2000) 325–343
11. Kask, K., Dechter, R., Larrosa, J.: Unifying cluster-tree decompositions for automated reasoning. Submitted to the AIJ (2003)
12. Kschischang, Frey, Loeliger: Factor graphs and the sum-product algorithm. IEEE Transactions on Information Theory **47** (2001) 498–519
13. Schiex, T., Fargier, H., Verfaillie, G.: Valued constraint satisfaction problems: Hard and easy problems. In: IJCAI95, Montreal (1995) 631–637
14. Chang, L.: Generalized constraint-based inference. Master's thesis, Department of Computer Science, University of British Columbia (2005)
15. Dean, T., Kanazawa, K.: A model for reasoning about persistence and causation. Comput. Intell. **5** (1990) 142–150
16. Howard, R., Matheson, J.: Influence diagrams. In Howard, R., Matheson, J., eds.: Readings on the Principles and Applications of Decision Analysis. Volume II. Strategic Decisions Group, Menlo Park, CA (1981) 721–762
17. Zhang, N.L.: A Computational Theory of Decision Networks. PhD thesis, University of British Columbia (1994)
18. Zadeh, L.A.: Fuzzy sets as a basis for a theory of possibility. Fuzzy sets and systems **1** (1978) 3–28
19. Bertele, U., Brioschi, F.: Nonserial Dynamic Programming. Academic Press, Inc. (1972)
20. Arnborg, S., Corneil, D.G., Proskurowski, A.: Complexity of finding embeddings in a k-tree. SIAM J. Algebraic and Discrete Methods **8** (1987) 277–284
21. Koster, A.M.C.A., Bodlaender, H.L., van Hoesel, C.P.M.: Treewidth: Computational experiments. Technical Report 01–38, ZIB, Berlin, Germany (2001)
22. Seidel, P.: A new method for solving constraint satisfaction problems. In: Seventh national conference on Artificial intelligence, AAAI (1981) 338–342
23. Dechter, R., Pearl, J.: Network-based heuristics for constraint satisfaction problems. Artificial Intelligence **34** (1987) 1–38
24. Dechter, R., Rish, I.: Directional resolution: The Davis-Putnam procedure, revisited. In: Principles of Knowledge Representation and Reasoning. (1994) 134–145
25. Zhang, N.L.: Probabilistic inference in influence diagrams. Computational Intelligence **14** (1998) 475–497
26. Viterbi, A.: Error bounds for convolution codes and an asymptotically optimal decoding algorithm. IEEE Transactions in Information Theory **13** (1967) 260–269
27. Cormen, T.H., Leiserson, C.E., Rivest, R.L.: Introduction to Algorithms. MIT Press/McGraw-Hill (1990)
28. Lauritzen, S.L., Spiegelhalter, D.J.: Local computations with probabilities on graphical structures and their application to expert systems. In: Readings in Uncertain Reasoning. Kaufmann (1990) 415–448
29. Dechter, R., Pearl, J.: Tree clustering for constraint networks (research note). Artif. Intell. **38** (1989) 353–366

30. Zhang, Y., Mackworth, A.K.: Parallel and distributed finite constraint satisfaction: Complexity, algorithms and experiments. In: Parallel Processing for Artificial Intelligence. Elsevier Science Publishers (1994) 305–334
31. Dechter, R., Rish, I.: A scheme for approximating probabilistic inference. In: Uncertainty in Artificial Intelligence (UAI97). (1997) 22–44
32. Mateescu, R., Dechter, R., Kask, K.: Tree approximation for belief updating. In: Eighteenth national conference on Artificial intelligence, AAAI (2002) 553–559
33. Murphy, K.P., Weiss, Y., Jordan, M.I.: Loopy belief propagation for approximate inference: An empirical study. In: Conf. on UAI. (1999) 467–475
34. McEliece, R.J., MacKay, D.J.C., Cheng, J.F.: Turbo decoding as an instance of Pearl's belief propagation algorithm. Journal on Selected Areas in Communications **16** (1998) 150–161
35. Weiss, Y., Freeman, W.T.: Correctness of belief propagation in gaussian graphical models of arbitrary topology. Neural Computation **13** (2001) 2173–2200
36. Mackworth., A.K.: Consistency in networks of relations. Artificial Intelligence **8** (1977) 99–118