Diagnosis as Semiring-based Constraint Optimization

Martin Sachenbacher[1] and **Brian Williams**

Abstract.

Constraint optimization is at the core of many problems in Artificial Intelligence. In this paper, we frame model-based diagnosis as a constraint optimization problem over lattices. We then show how it can be captured in a framework for "soft" constraints known as semiring-CSPs. The well-defined mathematical properties of a semiring-CSP allow to devise efficient solution methods that are based on decomposing diagnostic problems into trees and applying dynamic programming. We relate the approach to SAB and TREE*, two diagnosis algorithms for tree-structured systems, which correspond to special cases of semiring-based constraint optimization.

1 INTRODUCTION

Many problems in Artificial Intelligence can be framed as optimization problems where the task is to find a best assignment to a set of variables, such that a set of constraints is satisfied. Formalisms for soft constraints [19, 2] aim at more closely integrating constraint satisfaction and optimization. Soft constraints extend hard constraints by defining preference levels for the constraints, such that assignments are associated with an element from an ordered set. This element can be interpreted as weight, cost, utility, probability, or preference. A general framework for soft constraints are semiring-CSPs [2], which are based on a semiring (a set with two operations $+$ and $\times$ on it). The semiring operations ($+$ and $\times$) model constraint projection and combination, respectively.

In this paper, we show how model-based diagnosis, and in general optimization problems composed of a lattice preference structure and hard constraints, can be framed as semiring-CSPs. The approach is based on breaking down a global objective function and defining preference levels locally per each constraint. It enhances the practical usefulness of semiring-CSPs, and leads to a general framework where different notions of model-based diagnosis found in the literature (cardinality-minimal diagnosis, subset-minimal diagnosis, probabilistic diagnosis) can be easily obtained by choosing an appropriate semiring. In the process, we interpret and exploit assumptions commonly made in model-based diagnosis as special properties of the optimization problem behind it.

For classical constraint satisfaction problems (CSPs), local consistency techniques [14] provide the basis for effective solution methods. The mathematical properties of semiring-constraints ensure that local consistency is still applicable, except that it has to be organized as directional consistency in a tree-structured evaluation scheme. Methods for decomposition of constraint networks [12] can be extended to turn semiring-CSPs into equivalent, tree-structured instances. Expanding on previous work [6, 7, 13], we present algorithms for solving semiring-CSPs based on tree decompositions and directional consistency (an instance of dynamic programming) that can be used for efficiently computing a number of leading solutions to a diagnostic problem.

The paper is organized as follows. The next section formally defines model-based diagnosis as constraint optimization over lattices. Section 3 reviews semiring-CSPs. Section 4 frames constraint optimization over lattices, and in particular diagnosis, as a semiring-CSP, and defines conditions under which the global objective function can be folded into the constraints to define preference levels locally. Section 5 presents algorithms for solving semiring-CSPs efficiently based on tree decompositions and an instance of dynamic programming. Finally, in Section 6 we show that SAB and TREE*, two diagnosis algorithm for tree-structured systems [9, 20], can be understood as special instances of semiring-based constraint optimization.

2 DIAGNOSIS AS CONSTRAINT OPTIMIZATION OVER LATTICES

Definition 1 (Constraint System) *A* constraint system *over* $\{\top, \bot\}$ *is a tuple* (X, D, F) *where* $X = \{x_1, \ldots, x_n\}$ *is a set of variables,* $D = \{D_1, \ldots, D_n\}$ *is a set of finite domains, and* $F = \{f_1, \ldots, f_m\}$ *is a set of constraints. The constraints* f_j *are functions defined over var*(f_j) *where allowed tuples have value* $\top$ *and disallowed tuples have value* $\bot$.

For example, the boolean polycell circuit [21] shown in Fig. 1 can be framed as a constraint system with variables $X = \{a, b, c, d, e, f, g, x, y, z, o1, o2, o3, a1, a2\}$. Variables a to z are boolean variables with domain $\{0, 1\}$, whereas variables $o1$ to $a2$ describe the mode of a component and have domain $\{\text{G,B}\}$. If a component is good (denoted G) then it correctly performs its boolean function. If a component is broken (denoted B) then no assumption is made about its behavior. This "unknown mode" captures the concept of constraint suspension, introduced by Davis [3]. For the moment, we assume that observations (as stated in Fig. 1) are included in the set of constraints. We will come back later to the issue how they can be added at run-time.

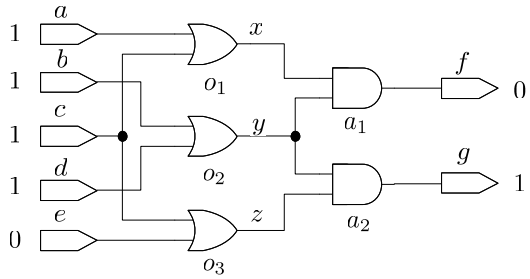

Figure 1. The Boolean Polycell example consists of three OR gates and two AND gates. Input and output values are observed as indicated.

[1] MIT CSAIL, Cambridge, MA, USA email: sachenba@mit.edu

In the following, by $t \downarrow_Y$ we denote the projection of a tuple on a subset Y of its variables. Given a constraint system C and a subset of the variables $Z \subseteq X$, a *solution* is a tuple t_Z over the variables in Z such that there exits an extension t of t_Z to all the variables X that fulfills the constraints, i.e., $t \downarrow_Z = t$ and $f_j(t \downarrow_{var(f_j)}) = \top$ for all $f_j \in F$. We denote the set of solutions of C as sol(C). Optimization extends a constraint system by an objective function to define preference levels on the solutions:

Definition 2 (Objective Function) *An* objective function U *maps tuples over $Z \subseteq X$ to a set A with a partial order $\leq_A$ that forms a complete lattice (i.e., every subset of elements $I \subseteq A$ has a greatest lower bound glb$(I) \in A$ and a least upper bound lub$(I) \in A$).*

In diagnosis, the set Z corresponds to the mode variables. For example, for the boolean polycell in Fig. 1, Z is the set of variables $\{o1, o2, o3, a1, a2\}$. Different notions of diagnosis correspond to different objective functions and lattices. In cardinality-minimal diagnosis [11], A is the set of integer values with total order $\leq$, and U returns for each mode assignment the number fault mode assignments. In probabilistic diagnosis [4], A is the interval $[0, 1]$ with total order $\leq$, and U associates a probability value with each mode assignment. In subset-minimal diagnosis [15, 4], A is the lattice of subsets of Z with partial order $\subseteq$, and each mode assignment is mapped to the subset of variables that represent a fault mode assignment. One can think of further instances, for example, associating a repair cost or partially ordered user preferences with each mode assignment.

For the boolean polycell example in Fig. 1, the cardinality-minimal diagnoses are $o1$=B, $o2$=G, $o3$=G, $a1$=G, $a2$=G with value 1 and $o1$=G, $o2$=G, $o3$=G, $a1$=B, $a2$=G with value 1. If we assume that OR gates have 1% probability of failure and AND gates have .5% probability of failure, then the two leading probabilistic diagnoses are the same assignments with values .0097 and .0048, respectively. The subset-minimal diagnoses are $o1$=B, $o2$=G, $o3$=G, $a1$=G, $a2$=G with value $\{o1\}$, $o1$=G, $o2$=G, $o3$=G, $a1$=B, $a2$=G with value $\{a1\}$, and $o1$=G, $o2$=B, $o3$=G, $a1$=G, $a2$=B with value $\{o2, a2\}$.

3 SEMIRING-CSPS

Semiring-CSPs [2] are a framework for "soft" constraints where the constraints are extended to include a preference level. Semiring-CSPs subsume many other notions of preferences in constraints, such as fuzzy CSPs [17], probabilistic CSPs [8], or partial constraint satisfaction [10].

Definition 3 ([2]) *A* c-semiring *is a tuple $(A, +, \times, \mathbf{0}, \mathbf{1})$ such that*

1. *A is a set and $\mathbf{0}, \mathbf{1} \in A$;*
2. *$+$ is a commutative, associative and idempotent (i.e., $a \in A$ implies $a + a = a$) operation with unit element $\mathbf{0}$ and absorbing element $\mathbf{1}$ (i.e., $a + \mathbf{0} = a$ and $a + \mathbf{1} = \mathbf{1}$;*
3. *$\times$ is a commutative, associative operation with unit element $\mathbf{1}$ and absorbing element $\mathbf{0}$ (i.e., $a \times \mathbf{1} = a$ and $a \times \mathbf{0} = \mathbf{0}$);*
4. *$\times$ distributes over $+$ (i.e., $a \times (b + c) = (a \times b) + (a \times c)$).*

For instance, $S_b = (\{0, 1\}, \vee, \wedge, 0, 1)$ forms a c-semiring. The idempotency of the $+$ operation induces a partial order $\leq_S$ over A as follows: $a \leq_S b$ iff $a + b = b$ (for S_b, $0 \leq_S 1$). In [2] it is shown that $(A, \leq_S)$ forms a lattice. The partial order defines levels of preference and allows to select the "best" solutions for constraints defined over a c-semiring.

Definition 4 (Constraint System over Semiring) *A* constraint system over a c-semiring *is a constraint system where the constraints $f_j \in F$ are functions defined over var(f_j) assigning to each tuple a value in A.*

"Classical" constraints [14] correspond to constraint systems over the semiring S_b, where allowed tuples have value 1 and disallowed tuples have value 0.

Definition 5 (Combination and Projection) *Let f and g be two constraints defined over var(f) and var(g), respectively. Then,*

1. *The* combination *of f and g, denoted $f \otimes g$, is a new constraint over $\mathrm{var}(f) \cup \mathrm{var}(g)$ where each tuple t has value $f(t \downarrow_{\mathrm{var}(f)}) \times g(t \downarrow_{\mathrm{var}(g)})$;*
2. *The* projection *of f on a set of variables Y, denoted $f \Downarrow_Y$, is a new constraint over $S \cap \mathrm{var}(f)$ where each tuple t has value $f(t_1) + f(t_2) + \ldots + f(t_k)$, where $t_1, t_2, \ldots, t_k$ are all the tuples of f for which $t_i \downarrow_Y = t$.*

Given a constraint system (X, D, F) over a c-semiring, the constraint optimization problem is to compute a function g over $Z \subseteq X$ such that $g(t)$ is the best value attainable by extending t to X, i.e. $g(t) = (\bigotimes_{j=1}^{m} f_j) \Downarrow_Z$.

4 DIAGNOSIS AS SEMIRING-BASED CONSTRAINT OPTIMIZATION

In this section we investigate how optimization over lattices, as defined in Sec. 2, and in particular diagnosis, can be framed as a semiring-CSP. Since the mathematical properties of semiring-CSPs ensure that local constraint propagation is applicable, this will be the basis for efficient solution methods for these problems.

We first show that it is possible to "reconstruct" an equivalent semiring-CSP from a constraint system over $\{\top, \bot\}$ and a lattice. We then investigate under which conditions it is possible to break down the global objective function and to define preference levels locally, i.e., per each constraint, such that the ranking of solutions is still preserved. This builds on conditions that were defined in [6] in the context of cost-based optimization in tree-structured CSPs. We illustrate how these conditions correspond to assumptions commonly made in model-based diagnosis.

Definition 6 (Composed Objective Function) *An objective function U is* $\times$-composed *of a set of functions $u_1, \ldots, u_k$, if $\times$ is a commutative, associative operation on A with unit element lub(A), absorbing element glb(A), and $u_1 \otimes \ldots \otimes u_k = U$.*

Theorem 1 (Optimization as Semiring-CSP) *Let $C = (X, D, F)$ be a constraint system over $\{\top, \bot\}$ and U an objective function $\times$-composed of $u_1, \ldots, u_k$. Define a constraint system (X, D, F') over A as follows: For each $f_j \in F$, let f'_j be defined over var(f_j) as $f'_j(t)$ = glb(A) if $f_j(t) = \bot$ and $f'_j(t)$ = lub(A), else. Let $F' = f'_1 \cup \ldots \cup f'_m \cup u_1 \cup \ldots \cup u_k$. Then $(A, \mathrm{lub}, \times, \mathrm{glb}(A), \mathrm{lub}(A))$ is a c-semiring, and $(\bigotimes_{i=1}^{m+k} f'_i) \Downarrow_Z = U(\mathrm{sol}(C))$.*

Every objective function U is trivially $\times$-composed of itself, by choosing $a \times b$ = glb$(\{a, b\})$. Together with Theorem 1, this implies that every constraint system C over $\{\top, \bot\}$ with objective function U can be turned into a semiring-CSP over A that has the same set of solutions as C and ranks them in the same way as U.

For instance, the objective function U for subset-minimal diagnosis (Sec. 2) is $\times$-composed of unary functions u_i defined over

$z \in Z$, where $\times \equiv \cup$, $u_i(t) = \emptyset$ if t represents a correct assignment, and $u_i(t) = \{z\}$ if t represents a faulty assignment. Likewise, the objective functions for cardinality-minimal diagnosis and probabilistic diagnosis are $\times$-composed of unary functions where $\times \equiv +$ and $\times \equiv \cdot$, respectively. For model-based diagnosis, non-trivially $\times$-composed objective functions correspond to the assumption that faults or sets of faults occur independently of each other.

Together with the results in [2], Theorem 1 establishes a one-to-one correspondence between lattice preference structures over "hard" constraints (i.e., $\{\top, \bot\}$ functions) and semiring-CSPs.

Up to now, we have two separate types of constraints in the semiring-CSP: functions that are defined only over variables from the set Z of variables of interest, and bi-valued functions that are defined over variables from the set X of all variables.

Definition 7 (Containment) *A function $f_i \in F$ is* contained *in $f_j \in F$, if* $\mathrm{var}(f_i) \subseteq \mathrm{var}(f_j)$.

We can reduce the set of constraints, without changing the solutions, by "absorbing" functions that are contained in other functions:

Theorem 2 (Absorbing Contained Constraints) *Let (X, D, F) be a constraint system over a c-semiring $(A, +, \times, \mathbf{0}, \mathbf{1})$. Let $f_i, f_j \in F$ be functions such that f_i is contained in f_j. Then for the constraint system (X, D, F') where $F' = F \setminus \{f_i, f_j\} \cup (f_i \otimes f_j)$, $(\bigotimes_{j=1}^{m} f_j) \Downarrow_Z = (\bigotimes_{j=1}^{m-1} f'_j) \Downarrow_Z$.*

For model-based diagnosis, assuming that faults are independent for each individual component means that there exists a $\times$-decomposition such that u_i will be contained in at least one f_j, and consequently, the objective function can be completely "absorbed" in the constraints representing the components. Note that this does not exclude cases where a component has more than one mode variable (e.g., sets of mode variables that are temporally indexed for different time steps), and it does not exclude cases where the objective function associates values with tuples of mode variables (e.g., a probability with the transition between two modes).

Table 1. Constraint f_{a1} in the polycell example (Fig. 1) for semirings S_c *(left)*, S_p *(center)*, and S_s *(right)*. Tuples not shown have value $\mathbf{0}$.

$a2$	g	y	z	
G	0	0	0	0
G	0	0	1	0
G	0	1	0	0
B	0	0	0	1
B	0	0	1	1
B	0	1	0	1
B	0	1	1	1

$a2$	g	y	z	
G	0	0	0	.995
G	0	0	1	.995
G	0	1	0	.995
B	0	0	0	.005
B	0	0	1	.005
B	0	1	0	.005
B	0	1	1	.005

$a2$	g	y	z	
G	0	0	0	$\emptyset$
G	0	0	1	$\emptyset$
G	0	1	0	$\emptyset$
B	0	0	0	$\{a1\}$
B	0	0	1	$\{a1\}$
B	0	1	0	$\{a1\}$
B	0	1	1	$\{a1\}$

We can now summarize different notions of model-based diagnosis, introduced in Sec. 2, as special cases of semiring-based constraint optimization. Table 1 shows the resulting constraint (after absorption) for an AND-gate for each of the three notions of diagnosis.

Cardinality-Minimal Diagnosis. Diagnoses where the preference criterion is the number of faulty components, and the goal is to minimize the total number of faulty components, can be obtained by choosing the semiring $S_c = (\mathbb{N}_0^+ \cup \infty, \min, +, \infty, 0)$.

Subset-Minimal Diagnosis. Diagnoses where the preference criterion is the set of faulty components, and the goal is to minimize the set of faulty components with respect to set inclusion, can be obtained by choosing the semiring $S_s = (2^Z, \cap, \cup, Z, \emptyset)$. The operator $\cap$ induces an ordering on $a, b \in 2^Z$ as follows: $a \leq_S b$ iff $a \supseteq b$.

Probability-Maximal Diagnosis. Diagnoses where the preference criterion is the probability of mode assignments, and the goal is to maximize the probability of a mode assignment, can be obtained by choosing the semiring $S_p = ([0, 1], \max, \cdot, 0, 1)$. For probabilistic diagnosis, the objective function being $\times$-decomposable corresponds to the assumption that failures are conditionally independent of each other.

5 DECOMPOSITION AND DYNAMIC PROGRAMMING

The mathematical properties of c-semirings (in particular, associativity and commutativity) guarantee that local constraint propagation, an efficient technique to solve classical ("hard") constraints, works in this extended framework as well. The exception is that the $\times$-operation is not necessarily idempotent, which means that constraint propagation cannot be applied in a "chaotic" way anymore. Research that aims at extending the notion of local consistency to soft constraints [1, 18] has therefore focused on directional consistency, where constraints are propagated in an organized way following a hierarchical (tree) scheme.

However, arbitrary constraint networks are not necessarily tree-structured. The goal of structural decomposition methods [12, 13] is to turn arbitrary constraint networks into equivalent, tree-structured (acyclic) instances, possibly by aggregating constraints together. Decomposition was developed in the context of hard constraints, but the idea can be naturally extended to constraint optimization [6]. Structural decomposition is based on the hypergraph H of a constraint system (X, D, F), which associates a node with each variable $x_i \in X$, and a hyperedge with each constraint $f_j \in F$. Figure 2 shows the hypergraph for the boolean polycell circuit.

Definition 8 (Tree Decomposition [12, 13]) *A* tree decomposition *for a constraint system (X, D, F) is a triple (T, χ, λ), where $T = (V, E)$ is a rooted tree, and χ, λ are labeling functions associating with each node $v \in V$ two sets $\chi(v) \subseteq X$ and $\lambda(v) \subseteq F$, such that*

1. *For each $f_j \in F$, there exists exactly one $v \in V$ such that $f_j \in \lambda(v)$. For this v, var$(f_j) \subseteq \chi(v)$; (covering condition);*
2. *For each $x_i \in X$, the set $\{v \in V \mid x_i \in \chi(v)\}$ induces a connected subtree of T (connectedness condition).*

Figure 3 shows a tree decomposition of the boolean polycell. For a constraint system $C = (X, D, F)$, a tree decomposition T defines an equivalent, tree-structured constraint system (X, D, F') that is found by combining the constraints in $\lambda(v)$, i.e., $F' = \bigcup_{v \in N} (\bigotimes_{f_j \in \lambda(v)} f_j)$. Note that a unary constraint over a variable x_i can be added to the tree decomposition, without violating the covering and connectedness conditions, by adding it as a child of any node v for which $x_i \in \chi(v)$. This allows one to perform decomposition as an off-line step, and to add observations for variables after the tree has been constructed.

For model-based diagnosis, absorption of the u_i-constraints stemming from the objective function is complete. This means that the hypergraph will correspond directly to the original device structure, which is often organized in a modular way that can be exploited through structural decomposition. For an arbitrary optimization problem, failure to decompose and absorb the objective function may lead to large constraints that make the problem hard to decompose, i.e., lead to a tree node with a large number of variables.

Decomposition can be understood as a minimal "repair" to the constraint system such that directional consistency techniques

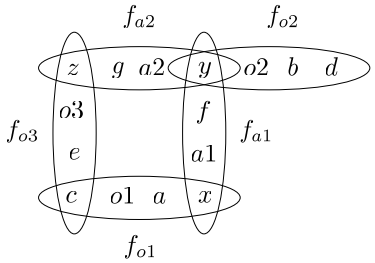

Figure 2. Hypergraph for the example in Fig. 1.

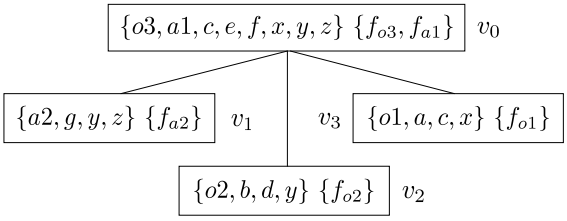

Figure 3. A tree decomposition of the hypergraph in Fig. 2, showing the labels χ and λ for each node.

(dynamic programming) become applicable. Solutions to a tree-structured semiring-CSP can be computed search-free using two steps. The first step computes values for tuples bottom-up using an instance of dynamic programming. This step can be viewed as generating an exact heuristic for search. In a second, top-down step, these values are used to enumerate solutions. This step can be viewed as a search that is guided by an exact heuristic, and is therefore backtrack-free.

Previous work on constraint optimization based on decomposition and dynamic programming [6, 7, 13] has focussed on the task of computing best values for individual variables or a single best assignment to all variables. We extend this work to address important requirements of the diagnosis context. First, in diagnosis it is typical that only a limited number of leading solutions is required. For instance, if the values of the solutions correspond to probabilities, the task could be to find a set of most likely solutions that cover most of the probability density space. We deliver on this requirement by exploiting a monotonicity property of c-semirings in the bottom-up and top-down phase to cut off the search space. Second, in diagnosis it is typical that most of the variables are not mode variables, and it would therefore be infeasible to enumerate solutions to the constraints that differ only in the values for variables $X \setminus Z$. Our approach avoids this by systematically eliminating these variables during the top-down phase.

The pseudocode for the bottom-up dynamic programming phase is shown in Fig. 4. In Fig. 4, function children() returns the set of children of a node. $f(v)$ is the constraint for node v. The operation $f(v) \otimes f(c) \Downarrow_{\mathrm{var}(f(v))}$, also known as semi-join, is the step that establishes directional consistency between a node and its parent. It is a generalization of directional arc consistency for CSPs [14] to the case of soft constraints [1, 18].

The restriction operator $|_{\leq}^{b}$ "prunes" tuples of a constraint by setting their value to $\mathbf{0}$ if it is worse than b. Formally, $f_j \ |_{\leq}^{b}$ returns a function f'_j where $f'_j(t) = f_j(t)$ if $f_j(t) \leq_S b$, and $f'_j(t) = \mathbf{0}$, else. If the bottom-up algorithm is provided with a cut-off parameter b, the restriction operator limits the computation to tuples whose value is

```
function solve(v, b)
  for each c_i ∈ children(v)
    solve(child)
    f(v) ← (f(v) ⊗ f(c_i) ⇓_{var(f(v))}) |_≤^b
    if c(v) ≡ 0 then
      throw inconsistent()
    end if
  end for
```

Figure 4. Bottom-up phase for solving a tree-structured semiring-CSP through dynamic programming

$\leq_S b$. This exploits the property that in a c-semiring, the $\times$-operator is extensive [2], i.e., $(a \times b) \leq_S a$ for all $a, b \in A$.

Values for solutions can be found by calling solve(root(T)), where root(T) is the root node of T. After completion of the algorithm, the best value of the tuples in $f(\mathrm{root}(T))$ is the value of the optimal solution. If $\leq_S$ is only a partial order, then the best value of the tuples in $f(\mathrm{root}(T))$ is a lub for the value of the optimal solution. The problem has no consistent solution if and only if there is a node v in the tree for which $f(v) \equiv \mathbf{0}$.

Table 2 shows the resulting root node constraint for probabilistic diagnosis of the boolean polycell example (i.e., using semiring S_p) and parameter b=5.0E-5 that limits computation to single faults of AND-gates and OR-gates and double faults of OR-gates.

Table 2. Constraint $f(v_0)$ for the polycell example (Fig. 1) after the bottom-up phase, using semiring S_p and the tree decomposition in Fig. 3. Tuples not shown have value $\mathbf{0}$.

$o3$	$a1$	c	e	f	x	y	z	
G	G	1	0	0	0	1	1	9.7E-3
G	B	1	0	0	1	1	1	4.8E-3
B	G	1	0	0	0	1	1	9.8E-5

The time complexity of the bottom-up phase is exponential in the maximum number of variables in a tree node (called the tree width), and its space complexity is exponential in the maximal number of variables that are shared between two tree nodes (called the separator size) [7, 13]. Hence, the benefit of tree decomposition is that it breaks down the complexity from being exponential in the number of all variables to being exponential in the number of variables per tree element (node or edge). Note that the complexity does not depend on the semiring, which means that the extension from constraint satisfaction (hard constraints) to constraint optimization (soft constraints) does not increase the complexity of constraint solving.

The pseudocode for the top-down solution enumeration phase is shown in Fig. 5. It enumerates the solutions with value $a \leq_S b$. For instance, in cardinality-minimal diagnosis (semiring S_c), one might perform the bottom-up phase with a limitation to single and double faults (b=2), and, if it turns out that single faults exist, enumerate only the single faults (b=1) in the top-down phase. It is easy to modify the top-down algorithm in such a way that, for example, the total number of enumerated solutions is restricted. In Fig. 5, preorder-node-iterator() enumerates the nodes of the tree T in pre-order (for the tree in Fig. 3, for example, in order v_0, v_1, v_2, v_3). Constraint r contains the resulting solutions. If the operator $\times$ is not idempotent, the bottom-up propagation has to be "canceled" by a semijoin operation $f_i \otimes^{-1} f_j \Downarrow_{\mathrm{vars}(f_i)}$ using the inverse ($\times^{-1}$) of the operator $\times$. As

function extract(T, b)
  $v \leftarrow$ preorder-node-iterator-first(T)
  $m \leftarrow \emptyset$
  $r \leftarrow f(v)\ |^b_{\leq}$
  begin loop
    for each $c_i \in$ children(v)
      $m \leftarrow m \cup (\chi(v) \cap \chi(c_i))$
    end for
    $r \leftarrow r \Downarrow_{(\mathrm{var}(r) \cap m) \cup Z}$
    $v \leftarrow$ preorder-node-iterator-next(T)
    if (v = **nil**) **then**
      return r
    end if
    if not ($\times$ idempotent) **then**
      $r \leftarrow r \otimes^{-1} f(v) \Downarrow_{\mathrm{var}(r)}$
    end if
    $r \leftarrow (r \otimes f(v))\ |^b_{\leq}$
    $m \leftarrow m \backslash (\chi(\mathrm{parent}(v)) \cap \chi(v))$
  end loop

Figure 5. Top-down phase for enumerating solutions to a tree-structured semiring-CSP for which $\times$ is idempotent or has an inverse.

solutions consist only of assignments to the variables $Z \subseteq X$, all other variables $X \setminus Z$ must be eliminated from the result. A variable in $X \setminus Z$ can only be eliminated once it no longer occurs in the remaining (unprocessed) part of the tree. In the algorithm shown in Fig. 5, the variables shared between r and the unprocessed part of the tree are represented by a multi-set m (m is a multi-set rather than a set because the same variable can occur on more than one edge of the tree).

Consider again the boolean polycell example for the semiring S_p, continuing on the example above. Assume that solution enumeration is performed with b = 5.0E-5. Initially, v is v_0, and r is the constraint shown in Table 2. Node v_0 has three children, and after the for-loop, multi-set m is $\{c, x, y, y, z\}$. The projection operation after the for-loop eliminates variables e and f from r. Assume v_1 is the next node of T in pre-order. Since operation $\times$ is not idempotent for S_p, a semi-join is performed between r and f_{a2} using the operation $\times^{-1} \equiv \div$, followed by a combination (full join) of the two constraints. Restriction $|^b_{\leq}$ removes 3 of the 6 tuples since their values fall below 5.0E-5. The loop is repeated after m is updated to $\{c, x, y\}$. Table 3 shows the final solutions.

Table 3. Solutions for the polycell example (Fig. 1) as enumerated by the top-down phase.

$o1$	$o2$	$o3$	$a1$	$a2$	
B	G	G	G	G	9.7E-3
G	G	G	B	G	4.8E-3
B	G	B	G	G	9.8E-5
B	B	G	G	G	9.8E-5

The complexity of the top-down phase is worst-case exponential in the number of type variables Z. The solution enumeration algorithm as stated in Fig. 5 requires that the $\times$-operator of the semiring is idempotent or has an inverse. This is the case for all three semirings S_c, S_s, and S_p.

6 SAB AND TREE*

SAB [5, 9] and TREE* [20] are two diagnostic algorithms for tree-structured systems.

SAB is a dynamic programming algorithm based on "weighting" assignments to mode variables. A correct assignment has weight 0, whereas an abnormal (faulty) assignment has weight 1. The goal is then to minimize the total sum of the weights. This corresponds to the semiring S_c. The assumption that mode variables are not shared between constraints is built into the weighting scheme; SAB would lead to incorrect results if applied to diagnostic models that violate this assumption. SAB has been combined with tree decomposition. However, SAB only extracts a single best solution, and it does not use a restriction operator. In [9], it has been shown that SAB compares favorably to the conflict-based diagnostic algorithm GDE [4].

Like SAB, TREE* computes cardinality-minimal diagnoses. TREE* is based on the idea that the set of consistent assignments to Z is sometimes small enough to associate it directly with each tuple, instead of associating a lub with each tuple that guides the enumeration of these assignments in a separate top-down phase. That is, TREE* collapses the bottom-up and the top-down phase into a single phase.

The set of assignments is concise because a cut-off is used and because mode assignments are compactly represented as subsets of Z. In TREE*, the variables Z (mode variables) are not included in the constraint system. Instead, mode assignments are associated with tuples of the constraints. Mode assignments combine through the operator $\cup$. Since sets of mode assignments are considered, the values of tuples combine through the cartesian product, $A \times B = \{a \cup b \mid a \in A, b \in B\}$. TREE* uses a cut-off to restrict the cardinality of the sets and thus the cardinality of the diagnoses. Since there is no separate solution enumeration phase, solutions are found by combining the values of tuples in the root of the tree (i.e., a special root node with $\chi = \emptyset$ is used).

TREE* treats the constraints and the values for their tuples separately, i.e., it performs semi-joins on bi-valued constraints, and updates the values of the tuples in a subsequent step. However, note that updating the values can become exponential in Z even if the task is only to find a single best diagnosis. Efficient data-structures, such as algebraic decision diagrams (ADDs) [16], exist for constraints (functions) over c-semirings where A is a subset of the real numbers (as is the case for S_c and S_p). For larger constraints and larger Z, it is therefore more efficient to separate the bottom-up and the top-down phases. Also, this allows for using two different cut-off parameters b, which permits better control over the number of diagnoses generated.

TREE* has been combined with a decomposition method for hard constraints called hypertree decomposition [12]. For hard constraints, hypertree decomposition is a more powerful decomposition method because unlike tree decomposition, it allows for reusing constraints in different nodes of the tree. However, in the context of soft constraints, this advantage diminishes because multiple occurrences of the same constraint clash with the possible non-idempotency of the constraint combination operator [13]. In [20] it has been empirically shown that TREE* can outperform SAB, an effect that can be mainly attributed to the use of a cut-off in TREE*.

7 CONCLUSION

This work builds on recent research in constraint programming and optimization, extending and modifying it for the context of model-based diagnosis.

Semiring-CSPs [2] are based on local preferences (defined per each constraint), whereas diagnosis is based on global preferences (defined per each solution). We therefore "reversed" the view in [2], starting from lattices over hard constraints, and investigated ways to fold them into a constraint system. This leads to methods and algorithms that allow to perform diagnosis over the general class of lattice preference structures. In contrast, existing diagnosis algorithms such as SAB and TREE* require that preferences are mutually independent for individual variables; in the terminology of our framework, the objective function must be $\times$-composed of unary functions. This is not required in our framework, although it can still be exploited: if the objective function is $\times$-composed of small (unary) functions, this will lead to better (complete) absorbtion of contained constraints (Theorem 2), and therefore to a smaller constraint system. Objective functions that are not $\times$-composed of unary functions occur in practice if probabilities are associated with mode transitions (it would be possible to introduce an extra variable for the mode transition, but at the cost of increasing the size of the constraint system).

Our work establishes a firm relationship between diagnosis as constraint satisfaction over lattices, semiring-based constraint optimization, and constraint propagation (dynamic programming) algorithms. This can lead to new and interesting insights. Consider, for example, the diagnostic task of maximizing the number of correctly working components instead of minimizing the number of faulty components. This problem might seem similar to cardinality-minimal diagnosis. However, unlike cardinality-minimal diagnosis, it does not form a c-semiring (the unit element of maximization, 0, is not the absorbing element of summation). Theorem 1 implies that it does not correspond to a lattice preference structure, and because semiring-CSPs capture necessary conditions for constraint propagation to work, it also follows that dynamic programming algorithms of the type presented in Section 5 do not exist for this diagnostic problem.

The algorithms presented in this paper have been implemented using a (modified) version of algebraic decision diagrams (ADDs) [16] to represent semiring-constraints. We are currently experimenting with random examples and real-world applications from the spacecraft domain. Current and future work includes incorporating AI and database techniques (such as best-first search and pipelining) in order to perform the constraint operations in an intelligent way, in particular processing large constraints only partially and caching intermediate results for incremental propagation.

ACKNOWLEDGEMENTS

We thank Stefano Bistarelli and Markus Stumptner for useful discussions that helped improve this paper.

REFERENCES

[1] Stefano Bistarelli, Rosella Gennari, and Francesca Rossi, 'Constraint propagation for soft constraints: Generalization and termination conditions', in *Principles and Practice of Constraint Programming*, pp. 83–97, (2000).

[2] Stefano Bistarelli, Ugo Montanari, and Francesca Rossi, 'Semiring-based constraint satisfaction and optimization', *Journal of the ACM*, **44**(2), 201–236, (1997).

[3] Randall Davis, 'Diagnostic reasoning based on structure and behavior', *Artificial Intelligence*, **24**(1–3), 347–410, (1984).

[4] Johan de Kleer and Brian C. Williams, 'Diagnosing multiple faults', *Artificial Intelligence*, **32**(1), 97–130, (1987).

[5] R. Dechter and A. Dechter, 'Belief maintenance in dynamic constraint networks', in *Proceedings of AAAI-88*, pp. 37–42, (1988).

[6] R. Dechter, A. Dechter, and J. Pearl, 'Optimization in constraint networks', in *Influence Diagrams, Belief Nets and Decision Analysis*, eds., R.M. Oliver and J.Q. Smith, 411–425, John Wiley & Sons, (1990).

[7] Rina Dechter, 'Bucket elimination: A unifying framework for reasoning', *Artificial Intelligence*, **113**, 41–85, (1999).

[8] H. Fargier and J. Lang, 'Uncertainty in constraint satisfaction problems: a probabilistic approach', in *Proceedings of ECSQARU-93*, pp. 97–104, (1993).

[9] Yousri El Fattah and Rina Dechter, 'Diagnosing tree-decomposable circuits', in *Proceedings of IJCAI-95*, pp. 1742–1749, (1995).

[10] Eugene C. Freuder, 'Partial Constraint Satisfaction', in *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence, IJCAI-89, Detroit, Michigan, USA*, pp. 278–283, (1989).

[11] Michael R. Genesereth, 'The use of design descriptions in automated diagnosis', *Artificial Intelligence*, **24**(1–3), 411–436, (1984).

[12] Georg Gottlob, Nicola Leone, and Francesco Scarcello, 'A comparison of structural CSP decomposition methods', *Artificial Intelligence*, **124**(2), 243–282, (2000).

[13] Kalev Kask, Rina Dechter, Javier Larrosa, and Fabio Cozman, 'Unifying tree-decomposition schemes for automated reasoning', Technical report, University of California, Irvine, (2001).

[14] Alan K. Mackworth, 'Constraint satisfaction', in *Encyclopedia of Artificial Intelligence*, ed., Stuart C. Shapiro, 285–293, John Wiley & Sons, (1992).

[15] R. Reiter, 'A theory of diagnosis from first principles', *Artificial Intelligence*, **32**(1), 57–95, (1987).

[16] R.I. Bahar, E.A. Frohm, C.M. Gaona, G.D. Hachtel, E. Macii, A. Pardo, and F. Somenzi, 'Algebraic Decision Diagrams and Their Applications', in *IEEE /ACM International Conference on CAD*, pp. 188–191, Santa Clara, California, (1993). IEEE Computer Society Press.

[17] Zsofi Ruttkay, 'Fuzzy constraint satisfaction', in *Proceedings 1st IEEE Conference on Evolutionary Computing*, pp. 542–547, Orlando, (1994).

[18] Thomas Schiex, 'Arc consistency for soft constraints', in *Principles and Practice of Constraint Programming*, pp. 411–424, (2000).

[19] Thomas Schiex, Hélène Fargier, and Gerard Verfaillie, 'Valued constraint satisfaction problems: Hard and easy problems', in *Proceedings IJCAI-95*, (1995).

[20] Markus Stumptner and Franz Wotawa, 'Diagnosing tree-structured systems', *Artificial Intelligence*, **127**(1), 1–29, (2001).

[21] Brian Williams and Robert Ragno, 'Conflict-directed A* and its role in model-based embedded systems', *Journal of Discrete Applied Mathematics*, (2003). to appear.