Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence

Filtering Decomposable Global Cost Functions *

D. Allouche[1] **C. Bessiere**[2] **P. Boizumault**[3] **S. de Givry**[1]
P. Gutierrez[4] **S. Loudni**[3] **JP. Métivier**[3] **T. Schiex**[1]
[1]UBIA UR 875, INRA, F-31320 Castanet Tolosan, France
[2]U. Montpellier, France
[3]GREYC-CNRS, UMR 6072, U. Caen, France
[4]IIIA-CSIC, U. Autònoma de Barcelona, Bellaterra, Spain

Abstract

As (Lee and Leung 2012) have shown, weighted constraint satisfaction problems can benefit from the introduction of global cost functions, leading to a new Cost Function Programming paradigm. In this paper, we explore the possibility of decomposing global cost functions in such a way that enforcing soft local consistencies on the decomposition achieves the same level of consistency on the original global cost function. We give conditions under which directional and virtual arc consistency offer such guarantees. We conclude by experiments on decomposable cost functions showing that decompositions may be very useful to easily integrate efficient global cost functions in solvers.

Introduction

Graphical model processing is a central problem in artificial intelligence. The optimization of the combined cost of local cost functions, central in the valued/weighted constraint satisfaction problem frameworks (Schiex, Fargier, and Verfaillie 1995) federates a variety of famous problems including CSP, SAT, Max-SAT, but also the *Maximum A posteriori Problem* (MAP) in Random Markov fields, the *Maximum Probability Explanation* (MPE) problem in Bayes nets (Koller and Friedman 2009) and polynomial pseudo-Boolean optimization (Boros and Hammer 2002). It has applications in resource allocation or bioinformatics.

The main approach to solve such problems in the most general situation relies on Branch and Bound combined with dedicated lower bounds for pruning. Such lower bounds can be provided by enforcing soft local consistencies (Cooper et al. 2010), as in Constraint Programming (CP) solvers. CP solvers are also equipped with global constraints which are crucial for solving large difficult problems. Dedicated algorithms for filtering such constraints have been introduced. For some global constraints such as REGULAR, CONTIGUITY, AMONG, it has been shown that a decomposition into a Berge-acyclic network of fixed arity constraints can lead to simpler implementation, without any loss in efficiency or effectiveness in filtering (Beldiceanu et al. 2005; Bessiere et al. 2008).

The notion of global constraints has been recently extended to weighted CSP, defining Global Cost Functions (Zytnicki et al. 2009; Lee and Leung 2012) with associated efficient filtering algorithms. In this paper, after some preliminaries, we define cost function decomposition and show how decomposable global constraints can be softened in families of decomposable global cost functions with the same decomposition structure. For Berge-acyclic decomposable global cost functions, we show that enforcing directional arc consistency or virtual arc consistency on the decomposition is essentially equivalent to a direct application on the original global cost function. Finally, we experimentally compare the efficiency of decomposed and monolithic versions of different global cost functions and observe important speedups using decompositions.

Preliminaries

Cost function network. A Cost Function Network (CFN) is a pair (X, W) where $X = \{1, \ldots, n\}$ is a set of n variables and W is a set of cost functions. Each variable $i \in X$ has a finite domain D_i of values that can be assigned to it. A value a in D_i is denoted (i, a). The maximum domain size is d. For a set of variables $S \subseteq X$, D^S denotes the Cartesian product of the domains of the variables in S. For a given tuple of values t, $t[S]$ denotes the projection of t over S. A cost function $w_S \in W$, with scope $S \subseteq X$, is a function $w_S : D^S \mapsto [0, k]$ where k is a maximum integer cost (finite or not) used to represent forbidden assignments (expressing hard constraints). To faithfully capture hard constraints, costs are combined using the bounded addition defined by $\alpha \oplus \beta = \max(k, \alpha + \beta)$. In this paper, a hard constraint is therefore represented as a cost function using only costs in $\{0, k\}$. If $\forall t \in D^S, z_S(t) \leq w_S(t)$, we say that the cost function z_S is a relaxation of w_S, denoted by $z_S \leq w_S$. A cost β may be subtracted from a larger cost α using the operation $\ominus$ where $\alpha \ominus \beta$ is $(\alpha - \beta)$ if $\alpha \neq k$ and k otherwise. Without loss of generality, we assume that every network contains one unary cost function w_i per variable and a 0-arity (constant) cost function $w_\varnothing$.

The central problem in CFN is to find an optimal *solution*: a complete assignment t minimizing the combined cost function $\bigoplus_{w_S \in W} w_S(t[S])$. This optimization problem

*This work has been funded by the "Agence nationale de la Recherche", reference ANR-10-BLA-0214.

has an associated NP-complete decision problem and restrictions to Boolean variables and binary constraints are known to be APX-hard (Papadimitriou and Yannakakis 1991).

A Constraint Network (CN) is a CFN where all cost functions are hard constraints (i.e., only using costs in $\{0, k\}$). Such cost functions are simply called constraints.

Local consistency. Algorithms searching for solutions in CNs usually enforce local consistency properties to reduce the search space. In CNs, the standard level of local consistency is generalized arc consistency (GAC). A constraint c_S is GAC iff every value in the domain of every variable in S has a support on c_S, where a support on c_S is a tuple $t \in D^S$ such that $c_S(t) = 0$. Enforcing GAC on c_S will often be called *filtering* c_S. General exact methods for solving the minimization problem in CFNs usually rely on branch and bound algorithms equipped with dedicated lower bounds. We consider here the incremental lower bounds provided by maintaining soft local consistencies such as directed arc consistency (DAC) (Cooper 2003; Larrosa and Schiex 2004) and virtual arc consistency (VAC) (Cooper et al. 2010).

Global cost function. A global constraint $c(S, \theta)$ is a family of constraints with a precise semantics parameterized by the set of variables S involved and possible extra parameters represented as θ. Global constraints usually have efficient associated local consistency enforcing algorithm (compared to generic filtering algorithms). Global constraints have been extended to define soft global constraints such as SOFTALLDIFF(S) (Petit, Régin, and Bessiere 2001) or SOFTREGULAR$(S, \mathcal{A}, d)$ (van Hoeve, Pesant, and Rousseau 2006)).

These "soft" global constraints are in fact hard global constraints including one extra variable in their scope representing the amount of violation of the assignment of the original variables. This amount of violation depends on the semantics of violation used for the softening of that global constraint. For several such constraints, efficient dedicated algorithms for enforcing GAC have been proposed.

Recently, different papers (Zytnicki et al. 2009; Lee and Leung 2012) have shown that it is possible to define soft global constraints as parameterized cost functions $z(S, \theta)$ directly providing the cost of an assignment. This approach allows to directly enforce soft local consistencies with dedicated algorithms providing stronger lower bounds. Indeed, compared to the previous cost variable based approach using constraints and GAC, cost functions and soft local consistencies offer improved filtering, thanks to the enhanced communication between cost functions enabled by the use of Equivalence Preserving Transformations (Cooper and Schiex 2004).

Hypergraph. The hypergraph of a CFN (or CN) (X, W) has one vertex per variable $i \in X$ and one hyperedge per scope S such that $\exists w_S \in W$. We consider CFNs with connected hypergraphs. The incidence graph of an hypergraph (X, E) is a graph $G = (X \cup E, E_H)$ where $\{x_i, e_j\} \in E_H$ iff $x_i \in X, e_j \in E$ and x_i belongs to the hyperedge e_j. An hypergraph (X, E) is Berge acyclic iff its incidence graph is acyclic.

Decomposing Global Cost Functions

Some global constraints may be efficiently decomposed into a logically equivalent subnetwork of constraints of bounded arities (Bessiere and Van Hentenryck 2003; Bessiere 2006). Similarly, global cost functions may be decomposed into a set of bounded arity cost functions. Notice that the definition below applies to any cost function, including constraints (cost functions using only costs in $\{0, k\}$).

Definition 1 *A decomposition of a global cost function $z(T, \theta)$ is a polynomial transformation δ_p (p being an integer) that returns a CFN $\delta_p(T, \theta) = (T \cup E, F)$ such that $\forall w_S \in F, |S| \leq p$ and $\forall t \in D^T, z(T, \theta)(t) = \min_{t' \in D^{T \cup E}, t'[T]=t} \bigoplus_{w_S \in F} w_S(t'[S])$.*

We assume, w.l.o.g, that every extra-variable $i \in E$ is involved in at least two cost functions in the decomposition.[1] Clearly, if $z(T, \theta)$ appears in a CFN $P = (X, W)$ and decomposes into $(T \cup E, F)$, then the optimal solutions of P can directly be obtained by projecting the optimal solutions of the CFN $P' = (X \cup E, W \setminus \{z(T, \theta)\} \cup F)$ on X.

Example Consider the ALLDIFF(S) constraint and its associated softened variant SOFTALLDIFF(S, dec) using the *decomposition* measure (Petit, Régin, and Bessiere 2001) where the cost of an assignment is the number of pairs of variables taking the same value. It is well known that ALLDIFF decomposes in a set of $\frac{n.(n-1)}{2}$ binary difference constraints. Similarly, the SOFTALLDIFF(S, dec) cost function can be decomposed in a set of $\frac{n.(n-1)}{2}$ soft difference cost functions. A soft difference cost function takes cost 1 iff the two involved variables have the same value and 0 otherwise. In these cases, no extra variable is required. Notice that the two decompositions have the same hypergraph structure.

Softening Decomposable Global Constraints

We now show that there is a systematic way of deriving decomposable cost functions as specific relaxations of existing decomposable global constraints.

As the previous ALLDIFF example showed, if we consider a decomposable global constraint, it is possible to define a softened decomposable global cost function by relaxing every constraint in the decomposition.

Theorem 1 *Let $c(T, \theta)$ be a global constraint that decomposes in a constraint network $(T \cup E, C)$ and f_θ a function that maps every $c_S \in C$ to a cost function w_S such that $w_S \leq c_S$. Then the global cost function $w(T, f_\theta)(t) = \min_{t' \in D^{T \cup E}, t'[T]=t} \bigoplus_{c_S \in C} f_\theta(c_S)(t'[S])$ is a relaxation of $c(T, \theta)$.*

Proof For any tuple $t \in D^T$, if $c(T, \theta)(t) = 0$, then $\min_{t' \in D^{T \cup E}, t'[T]=t} \bigoplus_{c_S \in C} c_S(t'[S]) = 0$ because $(T \cup E, C)$ is a decomposition of $c(T, \theta)$. Let $t' \in D^{T \cup E}$ be the tuple where this minimum is reached. This implies that $\forall c_S \in C$, $c_S(t'[S]) = 0$. Since $f_\theta(c_S)$ is a relaxation of c_S, this implies that $f_\theta(c_S)(t'[S]) = 0$ too. Therefore $\bigoplus_{c_S \in C} f_\theta(c_S)(t'[S]) = 0$ and $w(T, f_\theta)(t) = 0$. □

[1] Otherwise, such a variable can be removed by variable elimination: remove i from E and replace the w_S involving i by the cost function $\min_i w_S$ on $S \setminus \{i\}$. This preserves Berge-acyclicity.

By definition, the global cost function $w(T, f_\theta)$ is decomposable in $(T \cup E, W)$ where W is obtained by mapping f_θ on every element of C. Notice that, since f_θ preserves scopes, the hypergraph of the decomposition is preserved.

This result allows to immediately derive a long list of decompositions for global cost functions from existing decompositions of global constraints such as ALLDIFF, REGULAR, GRAMMAR, AMONG, STRETCH. The parameterization through f_θ allows a lot of flexibility.

Consider the ALLDIFF(V) constraint decomposed into a clique of binary differences. From a graph $G = (V, E)$, one can define a relaxation function f_G that preserves difference constraints $i \neq j$ when $(i, j) \in E$ but otherwise relaxes them to a constant cost function that is always equal to zero. This gives rise to a global cost function $w(V, f_G)$ that captures the graph coloring problem on G, an NP-hard problem. Thus, enforcing DAC or VAC on that single global cost function will be intractable as well, whereas enforcing DAC or VAC on its decomposition into binary cost functions will obviously be polynomial but will hinder the level of filtering achieved.

Consider the REGULAR$(\{X_1, \ldots, X_n\}, \mathcal{A})$ global constraint, defined by a finite automaton $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ where Q is a set of states, Σ the emission alphabet, δ a transition function from $\Sigma \times Q \to 2^Q$, q_0 the initial state and F the set of final states. As shown in (Bessiere et al. 2008), this constraint decomposes into a constraint network $(\{X_1, \ldots, X_n\} \cup \{Q_0, \ldots, Q_n\}, C)$ where the extra variables Q_i have Q as their domain. The set of constraints C in the decomposition contains two unary constraints restricting Q_0 to $\{q_0\}$ and Q_n to F and a sequence of identical ternary constraints $c_{\{Q_i, X_{i+1}, Q_{i+1}\}}$ which authorizes a triple (q, s, q') iff $q' \in \delta(q, s)$, thus capturing δ. An arbitrary relaxation of this decomposition may relax each of these constraints. The unary constraints on Q_0 and Q_n would be replaced by unary cost functions λ_{Q_0} and ρ_{Q_n} stating the cost for using every state as either an initial or final state while the ternary constraints would be relaxed to ternary cost functions $\sigma_{\{Q_i, X_{i+1}, Q_{i+1}\}}$ stating the cost for using any (q, s, q') transition. This relaxation precisely corresponds to the use of a weighted automaton $\mathcal{A} = (Q, \Sigma, \lambda, \sigma, \rho)$ (Culik II and Kari 1993). The cost of an assignment in the decomposition is equal, by definition, to the cost of an optimal parse of the assignment by the weighted automaton. This defines a WEIGHTEDREGULAR$(\{X_1, \ldots, X_n\}, \mathcal{A})$ global cost function. As shown in (Katsirelos, Narodytska, and Walsh 2011), a weighted automaton can encode the Hamming and Edit distances to the language of a classical automaton. Contrarily to the ALLDIFF example, we will see that WEIGHTEDREGULAR decomposition can be handled efficiently and effectively by soft local consistencies.

Local Consistency and Decompositions

The use of decompositions instead of their monolithic variant has both advantages and drawbacks. Thanks to local reasoning, a decomposition may be filtered more efficiently but this may also hinder the level of filtering achieved. In classical CSP, it is known that if the decomposition is Berge-acyclic, then enforcing GAC on the decomposition enforces GAC on the global constraint itself (Beeri et al. 1983). We show that a similar result can be obtained for cost functions using either DAC or VAC.

DAC has been originally introduced on binary cost functions using the notion of full support (Cooper et al. 2010). For a cost function w_S, a tuple $t \in D^S$ is a full support for a value (i, a) of $i \in S$ iff $w_i(a) = w_S(t) \bigoplus_{j \in S} w_j(t[j])$. Notice that either $w_i(a) = k$ and (i, a) does not participate in any solution or $w_i(a) < k$ and therefore $w_S(t) \bigoplus_{j \in S, j \neq i} w_j(t[j]) = 0$. DAC has been extended to non binary cost functions in (Sánchez, de Givry, and Schiex 2008) and (Lee and Leung 2009) with different definitions that coincide on binary cost functions. In this paper, we use a simple extension called T-DAC (for terminal DAC). Given a total order $\prec$ on variables, a CFN is said to be T-DAC w.r.t. $\prec$ iff for any cost function w_S, any value (i, a) of the maximum variable $i \in S$ according to $\prec$ has a full support on w_S.

VAC is a more recent local consistency property that establishes a link between a CFN $P = (X, W)$ and a constraint network denoted as $Bool(P)$ with the same set X of domain variables and which contains, for every cost function $w_S \in W, |S| > 0$, a constraint c_S with the same scope which forbids any tuple $t \in D^S$ such that $w_S(t) \neq 0$. A CFN P is said to be VAC iff the arc consistent closure of the constraint network $Bool(P)$ is non empty (Cooper et al. 2010).

Enforcing soft local consistencies

Enforcing such soft local consistencies relies on arc level *Equivalence Preserving Transformations* (EPTs) which apply to one cost function w_S (Cooper and Schiex 2004). Instead of deleting domain values, EPTs shift costs between w_S and the unary constraints $w_i, i \in S$ and therefore operate on a sub-network of P defined by w_S and denoted as $N_P(w_S) = (S, \{w_S\} \cup \{w_i\}_{i \in S})$. The main EPT is described as Algorithm 1. This EPT shifts an amount of cost $|\alpha|$ between the unary cost function w_i and the cost function w_S. The direction of the cost move is given by the sign of α. The precondition guarantees that costs remain non negative in the resulting equivalent network.

Algorithm 1: A cost shifting EPT used to enforce soft arc consistencies. The $\oplus, \ominus$ operations are extended to handle possibly negative costs as follows: for non negative costs α, β, we have $\alpha \ominus (-\beta) = \alpha \oplus \beta$ and for $\beta \leq \alpha$, $\alpha \oplus (-\beta) = \alpha \ominus \beta$.

```
1 Precondition: −w_i(a) ≤ α ≤ min_{t∈D^S, t[i]=a} w_S(t);
2 Procedure Project(w_S, i, a, α)
3     w_i(a) ← w_i(a) ⊕ α;
4     foreach (t ∈ D^S such that t[i] = a) do
5         w_S(t) ← w_S(t) ⊖ α;
```

To enforce T-DAC on a cost function w_S, it suffices to first shift the cost of every unary cost function $w_i, i \in S$ inside w_S by applying Project$(w_S, i, a, -w_i(a))$ for every

value $a \in D_i$. Let j be the maximum variable in S according to $\prec$, one can then apply Project(w_S, j, b, α) for every value (j, b) and $\alpha = \min_{t \in D^S, t[j]=b} w_S(t)$. Let t be a tuple where this minimum is reached. t is then a full support for (j, b): $w_j(b) = w_S(t) \bigoplus_{i \in S} w_i(t[i])$. This support can only be broken if for some unary cost functions $w_i, i \in S, i \neq j$, $w_i(a)$ increases for some value (i, a).

To enforce T-DAC on a complete CFN (X, W), one can simply sort W according to $\prec$ and apply the previous process on each cost function, successively. When a cost function w_S is processed, all the cost functions whose maximum variable appears before the maximum variable of S have already been processed which guarantees that none of the established full supports will be broken in the future. Enforcing T-DAC is therefore in $O(ed^r)$ in time, where $e = |W|$ and $r = \max_{w_S \in W} |S|$. Using the Δ data-structures introduced in (Cooper et al. 2010), space can be reduced to $O(edr)$.

The most efficient algorithms for enforcing VAC enforces an approximation of VAC called VAC$_\varepsilon$ with a time complexity in $O(\frac{ekd^r}{\varepsilon})$ and a space complexity in $O(edr)$. Alternatively, optimal soft arc consistency can be used to enforce VAC in $O(e^{6.5}d^{(3r+3.5)} \log M)$ time (where M is the maximum finite cost in the network).

Berge acyclicity and directional arc consistency

In this section, we show that enforcing T-DAC on a Berge-acyclic decomposition of a cost function or on the original global cost function yields the same cost distribution on the last variable and therefore the same lower bound (obtained by node consistency (Larrosa and Schiex 2003)).

Theorem 2 *If a global cost function $z(T, \theta)$ decomposes into a Berge-acyclic CFN $N = (T \cup E, F)$ then there is an ordering on $T \cup E$ such that the unary cost function w_{i_n} on the last variable i_n produced by enforcing T-DAC on the sub-network $(T, \{z(T, \theta)\} \cup \{w_i\}_{i \in T})$ is identical to the unary cost function w'_{i_n} produced by enforcing T-DAC on the decomposition $N = (T \cup E, F \cup \{w_i\}_{i \in T})$.*

Proof Consider the decomposed network N and $I_N = (T \cup E \cup F, E_I)$ its incidence graph. We know that I_N is a tree whose vertices are the variables and the cost functions of N. We root I_N in a variable of T. The neighbors (parent and sons, if any) of a cost functions w_S are the variables in S. The neighbors of a variable i are the cost functions involving i. Consider any topological ordering of the vertices of I_N. This ordering induces a variable ordering $(i_1, \ldots, i_n), i_n \in T$ which is used to enforce T-DAC on N. Notice that for any cost function $w_S \in F$, the parent variable of w_S in I_N appears after all the other variables of S.

Consider a value (i_n, a) of the root. If $w_{i_n}(a) = k$, then any complete assignment extending this value has cost $w_{i_n}(a)$. Otherwise, $w_{i_n}(a) < k$. Let w_S, be any son of i_n and t_S a full support of (i_n, a) on w_S. We have $w_{i_n}(a) = w_S(t) \bigoplus_{i \in S} w_i(t[i])$ which proves that $w_S(t) = 0$ and $\forall i \in S, i \neq i_n, w_i(t[i]) = 0$. I_N being a tree, we can inductively apply the same argument on all the descendants of i_n until leaves are reached, proving that the assignment (i_n, a) can be extended to a complete assignment with cost $w_{i_n}(a)$ in N. In either cases, $w_{i_n}(a)$ is the cost of an optimal extension of (i_n, a) in N.

Suppose now that we enforce T-DAC using the previous variable ordering on the undecomposed sub-network $(T, \{z(T, \theta)\} \cup \{w_i\}_{i \in T})$. Let t be a full support of (i_n, a) on $z(T, \theta)$. By definition $w_{i_n}(a) = z(T, \theta) \bigoplus_{i \in T} w_i(t[i])$ which proves that $w_{i_n}(a)$ is the cost of an optimal extension of (i_n, a) on $(T, \{z(T, \theta)\} \cup \{w_i\}_{i \in T})$. By definition of decomposition, and since $i_n \notin E$, this is equal to the cost of an optimal extension of (i_n, a) in N. □

T-DAC has therefore enough power to handle Berge-acyclic decompositions without losing any filtering strength, provided a correct order is used for applying EPTs.

Berge acyclicity and virtual arc consistency

Virtual Arc Consistency offers a simple and direct link between CNs and CFNs which allows to directly lift classical CNs properties to CFNs, under simple conditions.

Theorem 3 *In a CFN, if a global cost function $z(T, \theta)$ decomposes into a Berge-acyclic CFN $N = (T \cup E, F)$ then enforcing VAC on either $(T, \{z(T, \theta)\} \cup \{w_i\}_{i \in T})$ or on $(T \cup E, F \cup \{w_i\}_{i \in T})$ yields the same lower bound $w_\varnothing$.*

Proof Enforcing VAC on the CFN $P = (T \cup E, F \cup \{w_i\}_{i \in T})$ does not modify the set of scopes and yields an equivalent problem P' such that $Bool(P')$ is Berge-acyclic, a situation where arc consistency is a decision procedure. We can directly make use of Proposition 10.5 of (Cooper et al. 2010) which states that if a CFN P is VAC and if $Bool(P)$ is in a class of CSPs for which arc consistency is a decision procedure, then P has an optimal solution of cost $w_\varnothing$.

Similarly, the network $Q = (T, \{z(T, \theta)\} \cup \{w_i\}_{i \in T})$ contains just one cost function with arity strictly above 1 and $Bool(Q)$ will be decided by arc consistency. Enforcing VAC will therefore provide a CFN which also has an optimal solution of cost $w_\varnothing$. The networks P and Q having the same optimal cost by definition of a decomposition. □

Experimental Results

In this section, we intend to evaluate the practical interest of global cost function decompositions. Compared to the monolithic cost function filtering algorithm, these decompositions allow for a simple implementation and will provide effective filtering. But their actual performance needs to be evaluated.

All problems were solved using the CFN solver `toulbar2 0.9.5`[2] with pre-processing off (option line `-o -e: -f: -dec: -h: -c: -d: -q:`), and a variable assignment and DAC ordering compatible with the Berge-acyclic structure of the decompositions. The dynamic value ordering chooses the existential EAC value first (Larrosa et al. 2005). No initial upper bound is used. The same level of local consistency (namely (weak) EDGAC*, stronger than T-DAC and which therefore will produce an optimal $w_\varnothing$ for every global cost function) was used in all cases. All the experiments were run using several 2.66 Ghz Intel Xeon CPU cores with 64GB RAM.

[2] https://mulcyber.toulouse.inra.fr/projects/toulbar2.

Random WEIGHTEDREGULAR

Following (Pesant 2004), we generated random automata with $|Q|$ states and $|\Sigma|$ symbols. We randomly selected 30% of all possible pairs $(s, q_i) \in \Sigma \times Q$ and randomly chose a state $q_j \in Q$ to form a transition $\delta(s, q_i) = q_j$ for each such pair. The set of final states F is obtained by randomly selecting 50% of states in Q. Random sampling uses a uniform distribution.

From each automaton, we built two CFNs: one using a monolithic SOFTREGULAR cost function using Hamming distance (Lee and Leung 2009) and another using the Berge-acyclic decomposition of an equivalent WEIGHTEDREGULAR global cost functions. To make the situation more realistic, we added to each of these problems the same set of random unary constraints (one per non-extra variable, unary costs randomly chosen between 0 and 9). We measured two times: (1) time for loading and filtering the initial problem and (2) total time for solving the CFN (including the previous time). The first time is informative on the filtering complexity while the second emphasizes the incrementality of the filtering algorithms. Times were averaged on 100 runs and samples reaching the time limit of one hour were counted as such.

n	$\|\Sigma\|$	$\|Q\|$	Monolithic		Decomposed	
			filter	solve	filter	solve
25	5	10	0.12	0.51	0.00	0.00
		80	2.03	9.10	0.08	0.08
25	10	10	0.64	2.56	0.01	0.01
		80	10.64	43.52	0.54	0.56
25	20	10	3.60	13.06	0.03	0.03
		80	45.94	177.5	1.51	1.55
50	5	10	0.45	3.54	0.00	0.00
		80	11.85	101.2	0.17	0.17
50	10	10	3.22	20.97	0.02	0.02
		80	51.07	380.5	1.27	1.31
50	20	10	15.91	100.7	0.06	0.07
		80	186.2	1,339	3.38	3.47

Looking just to filtering time, it is clear that decomposition offers impressive improvements despite a much simpler implementation. Solving times show that it also inherits the excellent incrementality of usual consistency enforcing algorithms for free.

Nonograms

(prob012 in the CSPLib) are NP-complete logic puzzles in which cells in a grid have to be colored in such a way that a given description for each row and column, giving the lengths of distinct colored segments, is adhered to.

A $n \times n$ nonogram can be represented using n^2 Boolean variables x_{ij} specifying the color of the square at position (i, j). The restrictions on the lengths of segments in each row or column can be captured by a REGULAR constraint. In order to evaluate the interest of filtering decomposable cost functions, we have performed two types of experiments on nonograms.

Softened nonograms: can be built from classical nonograms by relaxing the strict adherence to the indicated lengths of colored segments. For this, we relax the REGULAR constraints on each row and column in the softened version using the Hamming distance. The associated cost indicates how many cells need to be modified to satisfy the attached description. This problem contains $2n$ WEIGHTEDREGULAR cost functions, with intersecting scopes. In order to be able to apply Theorem 2 on each of these global cost functions, one must build a global variable order which is a topological ordering for each of these cost functions. Although this requirement seems hard to meet in general, it is easy to produce in this specific case. The x_{ij} variables can, for example, be ordered in lexicographic order, from top left to bottom right and extra-variables inserted anywhere between their flanking original variables. Global cost function scopes are usually expressed to capture properties defined on time (as in rostering problems) or space (as in nonograms, or text processing problems). In those cases, the global order defined by time or space defines a global variable ordering that will often satisfy the conditions of Theorem 2.

Random $n \times n$ nonogram instances are generated by uniformly sampling the number of segments in each row/column between 1 and $\lfloor \frac{n}{3} \rfloor$. The length of each segment is uniformly and iteratively sampled from 1 to the maximum length that allows remaining segments to be placed (considering a minimum length of 1).

We solved these problems with `toulbar2` as before and measured the percentage of problems solved as well as the mean cpu-time (unsolved problems are counted for one hour) on samples of 100 problems.

Size	Monolithic		Decomposed	
	Solved	Time	Solved	Time
6×6	100%	1.98	100%	0.00
8×8	96%	358	100%	0.52
10×10	44%	2,941	100%	30.2
12×12	2%	3,556	82%	1,228
14×14	0%	3,600	14%	3,316

In this more realistic setting, involving different interacting global cost functions, decomposition is again the most efficient approach with orders of magnitude speedups.

White noise images: a random solution grid, with each cell colored with probability 0.5, is generated. A nonogram problem instance is created from the lengths of the segments observed in this random grid. These problems usually have several solutions, among which the original grid. We associate random unary costs, uniformly sampled betwen 0 and 99, with each cell. These costs represent the price to color the cell. A solution with minimum cost is sought. This problem has been modeled in `choco` (rel. 2.1.3, default options) and `toulbar2` (`-h:` option) using $2n$ REGULAR global constraints. In the `choco` model, a SCALAR constraint involving all variables is used to define the criteria to optimize. In `toulbar2`, coloring costs are captured by unary cost functions and the REGULAR constraints are represented by WEIGHTEDREGULAR cost functions with weights in $\{0, k\}$. The monolithic version has been tried but gave very poor results.

We measured the percentage of problems solved as well

as the mean cpu-time (unsolved problems are counted for $\frac{1}{2}$ hour, the time-limit used) on samples of 50 problems.

Size	choco		toulbar2	
	Solved	Time	Solved	Time
20×20	100%	1.88	100%	0.93
25×25	100%	14.78	100%	3.84
30×30	96%	143.6	96%	99.01
35×35	80%	459.9	94%	218.2
40×40	46%	1,148	66%	760.8
45×45	14%	1,627	32%	1.321

On this problem, enforcing soft filtering on decomposed global cost functions is preferable to traditional bound/GAC filtering of a pure CP model with cost variables. Using decomposition, the direct use of soft filtering such as EDAC, which subsumes T-DAC, provides a better exploitation of costs, with minimal implementation efforts.

Beyond decomposable cost functions

In some cases, problems may contain global cost functions which are not decomposable just because the bounded arity cost function decomposition is not polynomial in size. However, if the network is Berge-acyclic, Theorem 2 still applies. With exponential size networks, filtering will take exponential time but may yield strong lower bounds. The linear equation global constraint $\sum_{i=1}^{n} a_i x_i = b$ (a and b being small integer coefficients) can be easily decomposed introducing $n-3$ intermediate sum variables q_i and ternary sum constraints of the form $q_{i-1} + a_i x_i = q_i$ with $i \in [3, n-2]$ and $a_1 x_1 + a_2 x_2 = q_2$, $q_{n-2} + a_{n-1} x_{n-1} + a_n x_n = b$. The extra variables q_i have b values which is exponential in the representation of b. We consider the Market Split problem defined in (Cornuéjols and Dawande 1998; Trick 2003). The goal is to minimize $\sum_{i=1}^{n} o_i x_i$ such that $\sum_{i=1}^{n} a_{i,j} x_i = b_j$ for each $j \in [1, m]$ and x_i are Boolean variables in $\{0, 1\}$ (o, a and b being positive integer coefficients). We compared the Berge-acyclic decomposition in `toulbar2` with a direct application of the Integer Linear Programming solver `cplex` (version 12.2.0.0). We generated random instances with random integer coefficients in $[0, 99]$ for o and a, and $b_j = \lfloor \frac{1}{2} \sum_{i=1}^{n} a_{i,j} \rfloor$. We used a sample of 50 problems with $m = 4, n = 30$ leading to $\max b_j = 918$. The mean number of nodes developed in `cplex` is 50% higher than in `toulbar2`. But `cplex` was on average 6 times faster than `toulbar2` on these problems. 0/1 knapsack problems probably represent a worst case situation for `toulbar2` given that `cplex` embeds much of what is known about 0/1 knapsacks (and only part of these extend to more complicated domains). Possible avenues to improve `toulbar2` results in this unfavorable situation would be to use a combination of the m knapsack constraints into one as suggested in (Trick 2003) and a direct exploitation of the properties of the ternary linear constraints for more compact representation and more efficient filtering.

Related works

It should be pointed out that T-DAC is closely related to mini-buckets (Dechter 1997) and Theorem 2 can easily be adapted to this scheme. Mini-buckets perform a weakened form of variable elimination: when a variable x is eliminated, the cost functions linking x to the remaining variables are partitioned into sets containing at most i variables in their scopes and at most m functions. If we compute mini-buckets using the same variable ordering, with $m = 1$ and unbounded i, we will obtain the same marginal cost function as T-DAC on the root variable r, with the same time complexity. Mini-buckets can be used along two main recipes: precomputed (static) mini-buckets do not require update during search but restrict search to one static variable ordering; dynamic mini-buckets allow for dynamic variable ordering (DVO) but suffer from a lack of incrementality. Soft local consistencies, being based on EPTs, always yield equivalent problems, providing incrementality during search and are compatible with DVO. Soft arc consistencies also offer a space complexity in $O(edr)$ while mini-bucket may require space exponential in i.

Conclusion

In this paper, we have extended constraint decomposition to cost functions occurring in CFNs. For cost functions having a Berge-acyclic decomposition, we have shown that a simple filtering, at the directed arc consistency level, provides a comparable filtering on the decomposition or on the global cost function itself, provided a suitable variable ordering is used for DAC enforcing. For the stronger Virtual AC filtering, the same result is obtained, without any requirement.

The application of this result on the trivial class of Berge-acyclic global cost functions defined by Berge-acyclic decomposable global *constraints* is already significant since it allows to enforce *soft* local consistencies on networks containing Berge-acyclic decomposable global constraints such as REGULAR, GRAMMAR, AMONG,...

We have shown that these Berge-acyclic global constraints can also be relaxed into a Berge-acyclic global cost function using a generalization of the usual "decomposition" measure. This immediately provides a long list of Berge-acyclic decomposable global *cost functions*. Our experimental results based on the application of DAC on the relaxation of the REGULAR constraint into the WEIGHTEDREGULAR cost function show that the decomposition approach offers impressive speedups and cheap implementation compared to the monolithic cost function algorithms.

To experimentally evaluate the practical interest of the stronger result on VAC, a technically involved implementation of VAC on non binary constraints would be needed.

Although it is currently restricted to Berge-acyclic decompositions, this work paves the way for a more general form of "structural decompositions" of global cost functions where global cost functions decompose into an acyclic structure of local cost functions, with bounded separator sizes (but not necessarily of cardinality 1). These global structurally decomposed cost functions could then be filtered efficiently through dedicated incremental equivalence preserving transformations capturing non serial dynamic programming algorithms.

References

Beeri, C.; Fagin, R.; Maier, D.; and M.Yannakakis. 1983. On the desirability of acyclic database schemes. *Journal of the ACM* 30:479–513.

Beldiceanu, N.; Carlsson, M.; Debruyne, R.; and Petit, T. 2005. Reformulation of global constraints based on constraints checkers. *Constraints* 10(4):339–362.

Bessiere, C., and Van Hentenryck, P. 2003. To be or not to be ... a global constraint. In *Proc. CP'03*, 789–794.

Bessiere, C.; Hebrard, E.; Hnich, B.; Kiziltan, Z.; and Walsh, T. 2008. Slide: A useful special case of the cardpath constraint. In *Proc. of ECAI'08*, 475–479.

Bessiere, C. 2006. Constraint propagation. In Rossi, F.; van Beek, P.; and Walsh, T., eds., *Handbook of Constraint Programming*. Elsevier. chapter 3.

Boros, E., and Hammer, P. 2002. Pseudo-Boolean Optimization. *Discrete Appl. Math.* 123:155–225.

Cooper, M. C., and Schiex, T. 2004. Arc consistency for soft constraints. *Artificial Intelligence* 154(1-2):199–227.

Cooper, M.; de Givry, S.; Sanchez, M.; Schiex, T.; Zytnicki, M.; and Werner, T. 2010. Soft arc consistency revisited. *Artificial Intelligence* 174:449–478.

Cooper, M. C. 2003. Reduction operations in fuzzy or valued constraint satisfaction. *Fuzzy Sets and Systems* 134(3):311–342.

Cornuéjols, G., and Dawande, M. 1998. A class of hard small 0-1 programs. In *Integer Programming and Combinatorial Optimization, 6th International IPCO Conference, Houston, Texas, USA, June 22-24, 1998, Proceedings*, 284–293.

Culik II, K., and Kari, J. 1993. Image compression using weighted finite automata. In Borzyszkowski, A. M., and Sokolowski, S., eds., *MFCS*, volume 711 of *Lecture Notes in Computer Science*, 392–402. Springer.

Dechter, R. 1997. Mini-buckets: A general scheme for generating approximations in automated reasoning. In *Proc. of the* 16^{th} *IJCAI*, 1297–1303.

Katsirelos, G.; Narodytska, N.; and Walsh, T. 2011. The weighted grammar constraint. *Annals OR* 184(1):179–207.

Koller, D., and Friedman, N. 2009. *Probabilistic graphical models*. MIT press.

Larrosa, J., and Schiex, T. 2003. In the quest of the best form of local consistency for weighted CSP. In *Proc. of the* 18^{th} *IJCAI*, 239–244.

Larrosa, J., and Schiex, T. 2004. Solving weighted CSP by maintaining arc consistency. *Artif. Intell.* 159(1-2):1–26.

Larrosa, J.; de Givry, S.; Heras, F.; and Zytnicki, M. 2005. Existential arc consistency: getting closer to full arc consistency in weighted CSPs. In *Proc. of the* 19^{th} *IJCAI*, 84–89.

Lee, J. H.-M., and Leung, K. L. 2009. Towards efficient consistency enforcement for global constraints in weighted constraint satisfaction. In Boutilier, C., ed., *Proc of the* 21^{th} *IJCAI*, 559–565.

Lee, J., and Leung, K. 2012. Consistency techniques for flow-based projection-safe global cost functions in weighted constraint satisfaction. *Journal of Artificial Intelligence Research* 43:257–292.

Papadimitriou, C., and Yannakakis, M. 1991. Optimization, approximation, and complexity classes. *Journal of Computer and System Sciences* 43(3):425–440.

Pesant, G. 2004. A regular language membership constraint for finite sequences of variables. In Wallace, M., ed., *CP*, volume 3258 of *Lecture Notes in Computer Science*, 482–495. Springer.

Petit, T.; Régin, J.-C.; and Bessiere, C. 2001. Specific filtering algorithms for over-constrained problems. In *CP*, 451–463.

Sánchez, M.; de Givry, S.; and Schiex, T. 2008. Mendelian error detection in complex pedigrees using weighted constraint satisfaction techniques. *Constraints* 13(1-2):130–154.

Schiex, T.; Fargier, H.; and Verfaillie, G. 1995. Valued constraint satisfaction problems: hard and easy problems. In *Proc. of the* 14^{th} *IJCAI*, 631–637.

Trick, M. A. 2003. A dynamic programming approach for consistency and propagation for knapsack constraints. *Annals of Operations Research* 118(1-4):73–84.

van Hoeve, W. J.; Pesant, G.; and Rousseau, L.-M. 2006. On global warming: Flow-based soft global constraints. *J. Heuristics* 12(4-5):347–373.

Zytnicki, M.; Gaspin, C.; de Givry, S.; and Schiex, T. 2009. Bounds arc consistency for weighted CSPs. *Journal of Artificial Intelligence Research* 35(2):593–621.