Decomposing Global Cost Functions

C. Bessiere[1], P. Boizumault[2], S. de Givry[3], P. Gutierrez[4], S. Loudni[2], JP. Métivier[2], T. Schiex[3]

[1] LIRMM, Université de Montpellier, France
[2] GREYC, Université de Caen, France
[3] UBIA, UR 875, INRA, F-31320 Castanet Tolosan, France
[4] IIIA-CSIC, Campus Universitat Autònoma de Barcelona, 08193, Bellaterra, Spain

Abstract. Similarly to what has been done with Global Constraints in Constraint Programming, different results have been recently published on Global Cost Functions in weighted CSPs, defining the premises of a Cost Function Programming paradigm. In this paper, in the spirit of Berge-acyclic decompositions of global constraints such as REGULAR, we explore the possibility of decomposing Global Cost Functions in such a way that enforcing soft local consistencies on the decomposed cost function offers guarantees on the level of consistency enforced on the original global cost function. We show that an extension of Directional Arc Consistency to arbitrary arities and Virtual Arc Consistency offer specific guarantees. We conclude by preliminary experiments on WEIGHTEDREGULAR decompositions that show that decompositions may be very useful to easily integrate global cost functions in existing solvers with good efficiency.

.

Introduction

Graphical model processing is a central problem in AI. The optimization of the combined cost of local cost functions, central in the valued/weighted CSP frameworks [25], captures problems such as weighted MaxSAT, Weighted CSP or Maximum Probability Explanation in probabilistic networks. It has applications in *resource allocation*, *combinatorial auctions*, *bioinformatics*...

The main approach to solve such problems in the most general situation relies on Branch and Bound combined with dedicated lower bounds. Such lower bounds can be provided by enforcing soft local consistencies [5], leading to pruning as in Constraint Programming solvers. CP solvers are also equipped with global constraints which are often considered as crucial for solving large difficult problems. Dedicated algorithms for filtering such constraints have been introduced. For some classes of global constraints, among which the famous REGULAR constraint, it has been shown that using a direct decomposition of the constraint into

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

a Berge-acyclic network of fixed arities constraints could lead to better efficiency, with a simpler implementation and without losing effectiveness in filtering.

The notion of global constraints has been recently extended to weighted CSP, defining Global Cost functions [26,19,18] with associated efficient filtering algorithms. In this paper, we consider the possible decomposition of global cost functions into Berge-acyclic networks and see if enforcing local consistency on the decomposition can lead to a filtering which is comparable to the filtering obtained by directly enforcing the same consistency on the original global cost function.

To give body to this notion of Berge-acyclic decomposable cost functions, we use the WeightedRegular cost function, which relies on a weighted finite automaton to define a cost function on assignments, considered as a regular language.

After some preliminaries introducing Cost Function Networks and Soft Local Consistencies, we define Cost function decomposition and show how it can be applied to the WeightedRegular cost function in Section 2. Section 3 then shows that enforcing soft local consistencies such as Directional Arc Consistency or Virtual Arc Consistency on Berge-acyclic decompositions is essentially equivalent to a direct application on the original global cost function. Finally, Section 4 reports preliminary experiments comparing the efficiency of decomposed vs. monolithic version of the WeightedRegular cost function used to model SoftRegular cost functions.

1 Preliminaries

A Cost Function Network (CFN) or weighted CSP 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 than 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 domain 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 (or ∞) used to represent forbidden assignments (expressing hard constraints). To faithfully capture hard constraints, costs are combined using the bounded addition defined by $\alpha \oplus \beta = \max(k, \alpha + \beta)$. Observe that the intolerable cost k may be either finite or infinite. A cost β may also 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 $w_\varnothing$. A tuple t_S is said to be valid iff $\forall i \in S, w_i(t[i]) < k$.

The associated hyper-graph of a CFN (X, W) is an hypergraph with one vertex per variable $i \in X$ and one hyperedge per scope S such that $\exists w_S \in W$. We consider CFN with connected hypergraphs. The intersection graph of an hypergraph has one vertex by hyperedge and an edge connects two vertices iff their associated hyperedges intersect. An hyper-graph is Berge acyclic iff hyperedges intersect by at most one vertex and its intersection graph is acyclic [1].

The central problem in CFN is to find an optimal *solution*: a complete assignment t minimizing the combined cost function $\bigoplus_{w_S \in W} w_S(t[S])$, with a cost strictly lower than k. This optimization problem has an associated NP-complete decision problem and restrictions to boolean variables and binary constraints are known to be APX-hard [22]. It 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 [14] and quadratic pseudo-boolean optimization [3].

General exact methods for solving this minimization problem usually rely on branch and bound algorithms equipped with dedicated lower bounds. We focus in this paper on the incremental lower bounds provided by maintaining soft local consistencies at the arc level such as Directed Arc Consistency (DAC [6,17]) and Virtual Arc Consistency (VAC [5]).

DAC has been originally introduced on binary cost functions using the notion of strong support [5] and later extended to non binary cost functions in [24] and [19] with different definitions. These two definitions coincide on binary cost functions. In this paper, we use a simpler extension of DAC, and to avoid confusion, we call this variant T-DAC (for terminal DAC). Given a total order $\prec$ on variables, a binary CFN is said to be Terminal Directional Arc Consistent (T-DAC) w.r.t. $\prec$ iff for any cost function w_S, and for any value (i, a) of the maximum variable $i \in S$ according to $\prec$, there exists $t \in D_S, t[i] = a$ such that $w_i(a) = w_S(t) \bigoplus_{j \in S} w_j(t[j])$. The tuple t is a full support of (i, a) on w_S w.r.t. $\prec$. Note that either $w_i(a) = k$ and (i, a) does not participate in any solution or $w_i(a) < k$ and this implies that $w_S(t) \bigoplus_{j \in S, j \neq i} w_j(t[j]) = 0$.

Virtual Arc Consistency is a more recent local consistency property that establishes a link between a Cost Function Network $P = (X, W)$ and a Constraint Network denoted as $Bool(P)$ defined by the same set X of domain variables and such that every constraint in $Bool(P)$ is the result of the transformation of a cost function $w_S \in W$ into 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 Virtually Arc Consistent iff the arc consistent closure of the constraint network $Bool(P)$ is non empty [5].

Enforcing soft local consistencies

Enforcing such soft local consistencies relies on so-called arc level *Equivalence Preserving Transformations* (EPTs) which apply to one cost function w_S [7]. Instead of just deleting domain values, EPTs may shift cost 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 $N_P(w_S) = (S, \{w_S\} \cup_{i \in S} w_i)$. The main EPT used by arc level soft local consistencies 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: The main 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: $-w_i(a) \leq \alpha \leq \min_{t \in D_S, t[\{i\}]=a}\{w_S(t)\}$;
2 Procedure Project($w_S, i, a, \alpha$)
3     $w_i(a) \leftarrow w_i(a) \oplus \alpha$;
4     foreach ($t \in D_S$ such that $t[\{i\}] = a$) do
5         $w_S(t) \leftarrow w_S(t) \ominus \alpha$;
```

To enforce T-DAC on a single 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 strong 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 $w_i, i \in S, i \neq j$ and $w_i(a)$ increases for some value (i, a).

To enforce T-DAC on a the complete CFN (X, W), one can simply sort W according to the order of the maximum variable of every cost function 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 support will be broken. Enforcing T-DAC is therefore in $O(ed^r)$ in time. Using the Δ data-structures introduced in [5], space can be reduced to $O(edr)$.

The most efficient algorithms for enforcing VAC [5] actually enforce 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, and considering the worst case where the intolerable cost k is finite, 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).

2 Decomposing Global Cost Functions

Global constraints are usually described as families of constraints with a precise semantics parametrized by the number of variables they involve. Most of the usually considered global constraints allow for efficient local consistency enforcing (compared to the default GAC algorithm). The notion of global constraints has been extended to define Soft Global Constraints such as SoftAllDiff or SoftRegular [11]. These "soft" global constraints are not cost functions but classical global constraints defined over a set of variables which includes a dedicated "cost" variable representing the cost of the assignment of the remaining variables under the precise softened global constraint semantics. For several such constraints, efficient dedicated algorithm for enforcing Generalized Arc Consistency have been introduced [11].

Recently, different papers [26,19,18] have shown that it is possible to define Global Cost Functions as cost functions with a precise semantics parametrized by the number of variables they involve, together with efficient soft local consistency enforcing algorithms. Compared to the previous cost variable based approach, this new approach offers improved propagation thanks to the enhanced communication between cost functions enabled by the arc level EPTs used to enforce Soft AC [7], DAC and FDAC [6,17], EDAC [16], OSAC and VAC [5].

2.1 Decomposing Cost Functions

Similarly to constraints, cost functions may possibly decompose into a set of cost functions of smaller arities.

Definition 1. *A cost function z_T decomposes into a cost function network $(T \cup E, F)$ iff $\forall t \in D_T, z_T(t) = \min_{t' \in D_{T \cup E}, t'[T]=t} \bigoplus_{w_S \in F} w_S(t'[S])$.*

Clearly, if z_T appears in a CFN $P = (X, W)$ and decomposes into $(T \cup E, F)$, then the optimal solutions of P can be directly obtained by projecting the optimal solutions of the CFN $P' = (X \cup E, W \setminus \{z_T\} \cup F)$ on X.

For technical reasons, we introduce the notion of extra-minimal decompositions.

Definition 2. *A decomposition $(T \cup E, F)$ of z_T is said to be extra-minimal iff all variables in E are involved in at least two cost functions in F.*

This is done without loss of generality since from any decomposition, one can easily produce an extra-minimal decomposition: for any extra variable $i \in T$ which is involved in just one cost function $w_S \in F$, we can eliminate i from E, replace w_S by the cost function $f = \min_i w_S$ on $S \setminus \{i\}$ and get a network $(T \cup E \setminus \{i\}, F \cup \{f\} \setminus \{w_S\})$ which is an extra-minimal decomposition. This process removes extra variables and reduces scopes and therefore preserves Berge-acyclicity.

Example 1. Consider the soft All-different cost function with the so-called decomposition measure [11]: the cost of an assignment is equal to the number of pairs of variables taking the same value. This global cost function can be decomposed in a set of $\frac{n.(n-1)}{2}$ binary soft difference cost functions, each involving a different pair of variables. A soft difference cost function takes cost 1 iff the two involved variables are equal and 0 otherwise. In this case, no extra variable is required and the decomposition is therefore already extra-minimal.

As for global constraints, using decomposition may lead to more local reasoning which may be less effective. In all cases, it facilitates the implementation of the global cost function in solvers without requiring the cost function to be "projection safe" (ability to perform EPTs on the internal representation of global cost functions directly, as introduced in [19]).

2.2 From Regular to Weighted Regular

Initially introduced in [23], the $\textsc{Regular}_{\mathcal{A}}(i_1, \ldots, i_n)$ global constraint authorizes a tuple $(v_1, ..., v_n)$ iff it is a string of the language defined by the Finite Automaton $\mathcal{A}$. A deterministic finite automaton (DFA) is defined by (Q,Σ,θ,q_0,F), where Q is is a finite set of states, Σ is a finite set of symbols (the alphabet), $\theta : Q \times \Sigma \to Q$ is the transition function, $q_0 \in Q$ is the initial state and $F \subseteq Q$ is a set of final (or accepting) states. A non-deterministic finite automaton (NFA) is defined by (Q,Σ,δ,q_0,F) where δ is a transition function from $\Sigma \times Q \to 2^Q$. The automaton starts in the initial state q_0. In state q, the automaton inputs the next symbol $s \in \Sigma$ and moves to a state in $\delta(s, q)$. The string is accepted iff there is a path that ends in a state $q \in F$. The whole set of strings accepted by a N/DFA $\mathcal{A}$ is the language recognized by it, noted $L(\text{A})$. A regular language is a language which can be recognized by a DFA or a NFA.

The $\textsc{SoftRegular}^{d}_{\mathcal{A}}(i_1, \ldots, i_n, z)$ global constraint can be directly softened using any distance d between strings. This global constraint authorizes a tuple $(v_1, ..., v_n, c)$ iff the minimum distance according to d between $(v_1, \ldots, v_n)$ and a string of the language of $\mathcal{A}$ is c. Traditional distances between strings of the same length such as the Hamming distance (number of positions at which string differs) or the Edit distance (minimum number of substitutions, insertions and deletions needed to edit one string into the other) have been considered and dedicated global constraints with cost variables proposed in [11].

More recently, [8] has been considering using weighted automata as a more general way of expressing soft regular constraints. This has also been extended to CFG (Context Free Grammars) in [13]. We follow the same idea and consider the $\textsc{WeightedRegular}$ global *cost function*, defined through a weighted automaton.

Weighted Automata and Language A weighted version of a N/DFA was introduced in [12]. A weighted finite automaton (WFA) is a FA where the transition function δ is replaced by a transition cost function σ. In our WCSP context, this function will output cost in $[0, k]$: $\sigma : Q \times \Sigma \times Q \to [0, k]$. An additional "exit" cost function ρ encodes cost for exiting the automaton $\rho : F \to [0, k]$[5].

The weighted automaton starts in the initial state q_0. In state q, the automaton inputs the next symbol $s \in \Sigma$ and moves to a state q', paying a cost $\sigma(q, s, q')$. The string is accepted iff a state $q \in F$ is ultimately reached. The cost of such an accepting path is the sum of the cost of all the transitions used (including the final step reaching an element of F, defined by ρ). The cost associated with a string ℓ is defined as the minimum cost over all possible accepting paths. If a string cannot be derived, then its associated cost is just k (the intolerable cost).

The $\textsc{WeightedRegular}_{\mathcal{A}}(i_1, \ldots, i_n)$ cost function is defined on a sequence of variables T from a weighted automaton $\mathcal{A}$ using domain values as symbols in the set Σ. The cost of an assignment is just the cost of the string defined by the assignment of T according to $\mathcal{A}$.

[5] The usual definition of weighted automata includes also an "entry" cost function [21]. We don't use it in this paper.

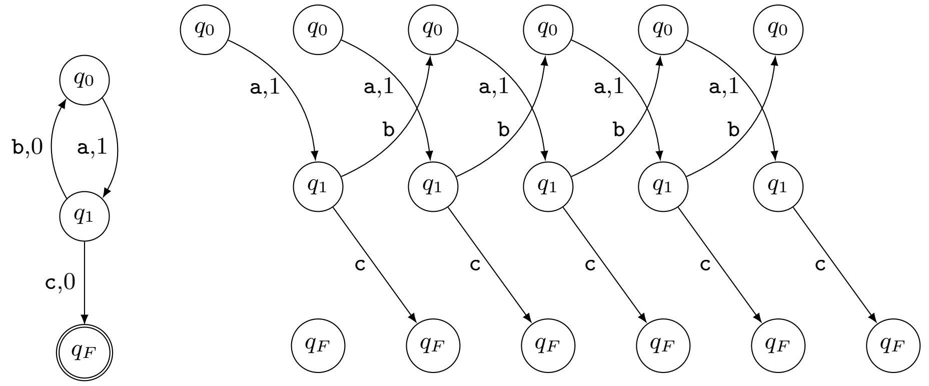
2.3 Decomposing Weighted Regular

We decompose the WeightedRegular using ternary cost functions encoding the weighted automaton and a sequence of extra state variables. This decomposition is similar in essence to the original decomposition of the Regular constraint but relies on cost functions. The decomposition is defined by:

- the original domain variables $i_1, \ldots, i_n \in T$,
- extra domain variables $s_0, \ldots, s_n$ representing automaton states. The domain of s_0 is just $\{q_0\}$, the domain of s_n is F. All other s_* variables have a domain equal to Q, the set of possible states.
- the set of ternary cost functions w_{s_{j-1},i_j,s_j} which returns $\sigma(s_{j-1}, i_j, s_j)$.
- a unary cost function w_{s_n} on s_n directly defined by ρ.

By construction, the minimum cost that $\bigoplus_{j=1}^{n} w_{s_{j-1},i_j,s_j} \oplus w_{s_n}$ can take is precisely the cost defined by WeightedRegular$_{\mathcal{A}}(i_1, \ldots, i_n)$.

This construction can be seen as a WCSP representation of the "unrolled" automaton transition graphs over n steps where each variable s_j represents the possible states at step j. The figure below illustrates this on a simple (deterministic) automaton for words $a(ba)^*c$ with each occurrence of `a` having cost 1 and a sequence of 6 variables (this automaton can only emit even length words whose cost is half the length). The unrolled automaton on the right does not mention costs for clarity (cost 0 everywhere except for edges emitting `a` with cost 1). Omitted edges represent intolerable k costs. It should be clear that one can associate one variable with each "column" of states, from the first to the last column. The additional variable i_j capture the possible characters emitted, and the triple has the associated emitting cost.



Similarly to what has been shown in cost variable decomposition using context free grammars [13], it is possible to encode Hamming and Edit distance based soft constraints using a weighted automaton and therefore a WeightedRegular cost function.

Considering the Hamming distance, from an original DFA $\mathcal{A}$, we just derive a weighted automata with the same alphabet and states, with a constant zero exit function ρ and a transition cost function $\sigma(q, s, q')$ defined as:

- 0 whenever $q' \in \delta(s, q)$ in $\mathcal{A}$,
- 1 whenever $q' \notin \delta(s, q)$ in $\mathcal{A}$, but $\exists t \in \Sigma, t \neq s$, such that $q' \in \delta(t, q)$,
- the intolerable cost k otherwise.

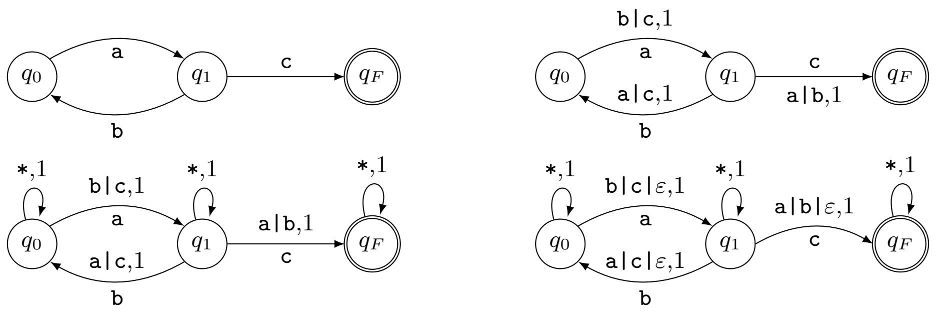


Fig. 1. From top-left to bottom-right, in reading order: (tl) the DFA for the language $a(ba)^*c$ (tr) the WFA encoding the Hamming distance to the previous automaton, (bl) the WFA allowing both for substitution and insertion (br) the WFA for the Edit distance. An arc is labelled by symbols,cost pairs where 0 costs are omitted and a '|' separated list of symbols is used to factorize several transitions with same source, destination and cost.

The Edit distance $d(s_1, s_2)$ of two words s_1 and s_2 is the smallest number of insertions, deletions, and substitutions required to change one word into another. It captures the fact that two words that are identical except for one extra or missing symbol should be considered close to one another. Considering the Edit distance from an original DFA $\mathcal{A}$, we derive a WFA with alphabet $\Sigma \cup \{\varepsilon\}$ and states Q with four kinds of transition cost functions:

1. copy of automaton $\mathcal{A}$: $\sigma(q, s, q')$=0 for each transition $q' \in \delta(s, q)$ in $\mathcal{A}$,
2. substitution: transition cost function is the same as for Hamming,
3. insertion: for each $q \in Q$ *s.t.* $q \notin \delta(s, q)$, $\sigma(q, s, q)$=1
4. deletion: to each transition $q' \in \delta(s, q)$ in $\mathcal{A}$ is associated an ε-transition[6] $q' \in \delta(\varepsilon, q)$ *s.t.* $\sigma(q, \varepsilon, q')$=1

Consider the DFA for the language $a(ba)^*c$ (Fig. 1-top-left). The WFA for substitution is depicted Fig. 1 (top-right), for substitution/insertion Fig. 1 (bottom-left) and for substitution/insertion/deletion Fig. 1 (bottom-right).

[6] An ε-transition is a transition using an empty symbol ε which does not emit anything. In practice, to derive a given word, a transition from state q to q' emitting ε means that we can directly go from q to q' without considering available symbols.

Unfortunately, the decomposition we presented in the beginning of this Section for WeightedRegular based on WFA without ε-transitions does not extend directly to WFA with ε-transitions, which are used for deletions. However, ε-transitions can be removed by computing the ε-closure of the WFA (see [21]). Then, the resulting WFA can be directly decomposed.

3 Local Consistency and Decompositions

Decomposing large arity constraints or cost functions into an equivalent combination of smaller arity components may sometimes be practically useful by itself: small arity may weaken propagation but may improve efficiency. Ultimately, this is a matter of compromise (especially in unstructured domains where there is no dedicated propagator for the considered cost function [9]).

However, for global cost functions, which have a precise semantics, it may be possible to define decompositions such that enforcing a given level of consistency on the decomposition offers a guarantee on the strength of the filtering compared to what would have been done using the original (non decomposed) cost function.

On classical constraint networks, different global constraints, such as the Regular global constraint, can be decomposed into a Berge-acyclic set of small arity constraints. By virtue of arc consistency, it is known that enforcing GAC on the set of Berge-acyclic constraint enforces GAC on the global constraint itself [1]. We show that a similar result can be obtained for cost functions and we illustrate this on the WeightedRegular cost function.

3.1 Directional AC

In this section, we will show that enforcing T-DAC on a decomposition of a cost function is equivalent to enforcing it on the original cost function itself, as far as the decomposition is Berge-acyclic.

We consider a decomposable global cost function z_T on the variables $T =$ with possible associated unary cost function $w_i, i \in T$. We assume that z_T can be decomposed in an extra-minimal Berge-acyclic cost function network $N = (T \cup E, F)$.

We now show that there exists a variable ordering on $T \cup E$ such that enforcing T-DAC on the decomposition is as strong as enforcing T-DAC on the original global cost function z_T.

Theorem 1. *In a CFN, if a global cost function z_T decomposes into an extra-minimal Berge-acyclic cost function network $N = (T \cup E, F)$ 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 subnetwork $(T, \{z_T\} \cup_{i \in T} w_i)$ is identical to the unary cost function w'_{i_n} produced by enforcing T-DAC on the decomposition $N = (T \cup E, F \cup_{i \in T} w_i)$.*

Proof. We first show that for any ordering of the variables, and any value (i_n, a) of the last variable i_n of T according to the ordering, enforcing T-DAC on the subnetwork $(T, \{z_T\} \cup_{i \in T} w_i)$ guarantees that the single value assignment (i_n, a) can be extended to a complete assignment t of T with cost $w_{i_n}(a)$. Let us consider t a full support of (i_n, a) on z_T. By definition $w_{i_n}(a) = z_T \bigoplus_{i \in T} w_i(t[i])$ which is precisely the cost of the complete assignment t in the subnetwork $(T, \{z_T\} \cup_{i \in T} w_i)$.

Conversely, we will now prove that the same property holds after enforcing T-DAC on the decomposition of z_T. The proof is not difficult but technical. Let $L = (F, I)$ be the intersection graph associated with the decomposition. L has one vertex per cost function in F and an edge $e \in I$ connects any two cost functions sharing a variable. Because of the Berge-acyclicity of N, L is acyclic (a forest). We can assume without loss of generality that L is connected (a tree). Indeed, if it is not connected, we can select two connected components and link them by adding one dummy constant zero binary cost function in F and just repeat this until L is connected. This preserves Berge-acyclicity and does not change the maximum arity in F.

We root L in any cost function that involves at least one original variable in T. We then perform a topological sort on L which successively selects a cost function with just one non selected neighbor in L (a leaf), choosing first cost functions whose intersection with this neighbor is an extra variable from E. Let $f_{S_1}, \ldots, f_{S_e}$ be the sequence of cost functions that are successively removed by the topological sort. Note that f_{S_i} intersects with $f_{S_{i+1}}$ by at most one variable (Berge acyclicity). This intersecting variable will be denoted as $\mathit{Root}(f_{S_i})$.

This ordering on cost functions can be extended to an order on variables as follows: we take each cost function f_{S_i} in the previous order and replace it by the sequence of all the variables in S_i which have not already been ordered. This sequence itself is ordered in such a way that the variable $\mathit{Root}(f_{S_i})$ that intersects with $f_{S_{i+1}}$ (if any) is ordered last. This produces an order $j_1, \ldots, j_m$ over variables in $T \cup E$.

First note that the variable j_m must be a variable of T. Indeed, since the decomposition is network minimal, all extra variables in f_{S_e} must appear in the intersection between scopes with previous cost functions. Because of the topological ordering chosen, if j_m is en extra variable, this means that f_{S_e} involves only extra variables which contradicts the choice of a root with a variable from T. Thus, $j_m \in T$ and $j_m = i_n$.

Assume now that $(T \cup E, F \cup_{i \in T} w_i)$ is made T-DAC consistent (which does not change scopes). Consider a value (i_n, a). If $w_{i_n}(a) = k$, then clearly any complete assignment extending this value has a cost of k and the property is proved. Otherwise, $w_{i_n}(a) < k$. Let t_e be a strong support of this value. We have $w_{i_n}(a) = w_{S_e}(t_e) \bigoplus_{i \in S_e} w_i(t_e[i])$ which proves that t_e is an assignment of S_e with the same cost $w_{i_n}(a)$ and such that $\forall i \in S_e, i \neq i_n), w_i(t_e[i]) = 0$. This is specifically true for any of the variables i in S_e that intersect with scopes of children cost functions, where the value $t_e[i]$ will have a zero unary cost. Since L is a tree, we can inductively use the same argument based on T-DAC to show that the tuple t_e can be extended to more variables with no increase in cost until the leafs of the tree are reached and all variables are assigned. This proves that the assignment (i_n, a) can be extended to complete assignment of $(T \cup E, F \cup_{i \in T} w_i)$ with cost $w_{i_n}(a)$. □

This result shows that directional consistency has enough power to handle Berge-acyclic decompositions in Weighted CSP without losing any propagation strength, provided a correct order is used for cost function propagation.

In practice, a CFN may contain different Berge-acyclic decomposable cost functions sharing variables and there may be no variable order which would be compatible with all the Berge-acyclic structures of these decomposed cost functions. In this case, a possibility would be to use so-called "Propagator groups" [15] which have been recently proposed for Berge-acyclic propagation in CN. Each propagator group is in charge of propagating one decomposition. For CFN, proper ordering between groups will likely be needed to avoid cycling.

3.2 Virtual AC

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

Consider a decomposable global cost function z_T on the variables T with possible associated unary cost functions $w_i, i \in T$. We assume that z_T can be decomposed in an extra-minimal Berge-acyclic cost function network $N = (T \cup E, F)$.

Theorem 2. *In a CFN, if a global cost function z_T decomposes into a Berge-acyclic cost function network $N = (T \cup E, F)$ then enforcing VAC on either $(T, \{z_T\} \cup_{i \in T} w_i)$ or on $(T \cup E, F \cup_{i \in T} w_i)$ yields the same lower bound $w_\varnothing$.*

Proof. Enforcing VAC on the CFN $P = (T \cup E, F \cup_{i \in T} w_i)$ 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 [5] which states that if a CFN P is VAC and $Bool(P)$ 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 $Q = (T, \{z_T\} \cup_{i \in T} w_i)$ 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 has also an optimal solution of cost $w_\varnothing$. The networks P and Q having optimal solutions of the same cost by definition of a decomposition, the result follows. □

4 Experimental Results

In this section, we intend to probe the possible interest of global cost function decompositions. These decompositions allow for a simple implementation, but it is also interesting to check if they can improve filtering performances or, at least, not degrade them too much.

We implemented the ternary encoding of the WeightedRegular cost function allowing to model a SoftRegular with Hamming distance. Since the monolithic propagator for the global SoftRegular cost function is already implemented in the WCSP solver `toulbar2` with associated (weak) E/FDGAC* filtering, this allows to compare the decomposition and monolithic versions.

Following [23], we generated random automata with $|Q|$ states and an alphabet size $|\Sigma|$. 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. All random samples use a uniform distribution.

From each automaton, we built two CFNs: one using a monolithic SoftRegular cost function using Hamming distance and another using the Berge-acyclic decomposition of the WeightedRegular cost function encoding the same cost function. 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 (with unary costs randomly chosen between 0 and 9). These problems have been solved using the CFN solver `toulbar2` (See `https://mulcyber.toulouse.inra.fr/projects/toulbar2`, version 0.9.4).

All preprocessing options of toulbar2 except filtering were turned off (option line `-o -e: -f: -dec: -h: -c: -d:`) and a DAC ordering compatible with the Berge-acyclic structure of the decomposition was used. The value ordering used chooses the existential EAC value first. The adaptive variable ordering heuristic is dom/wdeg. No initial upper bound is used. The same level of local consistency (namely (weak) EDGAC*, stronger than the T-DAC consistency considered in the paper and which therefore will also produce an optimal $w_\varnothing$ at the root) was used in all cases. We measured two times: (1) time for loading the problem and filtering the root node of the search tree 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. All the experiments were run on one 2.66 Ghz Intel Xeon CPU core with 64Gb RAM available with a time-limit of 5'. Times were averaged on 30 runs and samples reaching the time limit are considered as terminating in 5'. Table 1 shows the results.

Clearly, the decomposition brings an impressive progress in terms of efficiency. The results of these experiments should however be considered with care. The current implementation of the monolithic version of SoftRegular in `toulbar2` is probably far from optimized and, clearly, it has very limited incrementality. On the other hand, the core table filtering algorithms of `toulbar2` show excellent incrementality. These results could still be improved however: the ternary decomposition uses the same ternary cost function over and over in the problem. This is currently ignored by the solver. Taking this into account could considerably lower memory usage and file sizes.

It is also interesting but not trivial to compare the asymptotic complexities of the algorithms considered here. Enforcing EDGAC* on the decomposition is in $O(\max(nd, k).nd^3)$ [24].

n	$\|\Sigma\|$	$\|Q\|$	Monolithic		Decomposed		n	$\|\Sigma\|$	$\|Q\|$	Monolithic		Decomposed	
			filter	solve	filter	solve				filter	solve	filter	solve
25	5	10	0.08	0.46	0.00	0.00	50	5	10	0.30	3.39	0.00	0.00
		20	0.20	1.17	0.01	0.01			20	0.78	9.01	0.01	0.01
		40	0.47	2.69	0.02	0.02			40	2.41	28.58	0.04	0.05
		80	1.63	9.55	0.08	0.08			80	8.74	108.3	0.17	0.18
25	10	10	0.42	2.36	0.00	0.00	50	10	10	1.70	18.01	0.01	0.01
		20	0.94	5.37	0.02	0.02			20	3.86	41.58	0.03	0.04
		40	2.31	13.36	0.06	0.06			40	10.78	117.7	0.13	0.14
		80	5.56	32.65	0.23	0.25			80	260.2	260.2	0.48	0.53
25	20	10	2.07	11.30	0.01	0.01	50	20	10	8.44	84.88	0.03	0.03
		20	5.13	27.95	0.04	0.04			20	21.49	213.5	0.08	0.09
		40	12.40	66.40	0.14	0.15			40	300	300	0.29	0.31
		80	30.34	164.7	0.51	0.55			80	300	300	1.06	1.14

Table 1. Time in seconds to *filter* the CFN and to *solve* it to optimality. The CFNs encode a SoftRegular cost function representing the Hamming distance between a set of n variablesdefining a string and the language of a randomly generated DFA with $|\Sigma| = d$ symbols and $|Q|$ states using either the monolitic filtering algorithm of [18] or a ternary encoding of the equivalent WeightedRegular presented in Section 2.3, using table cost functions.

The asymptotic complexity for enforcing (weak) E/FDGAC* on the global SoftRegular cost function is not so simple to bound and it depends on the nature of the string distance and automaton used. For the Hamming distance and a deterministic automata, the underlying network has $O(|Q|.(n+1))$ vertices and $O(n(|\theta| + |Q|d)) = O(n.|Q|d)$ edges. Therefore, the complexity for just applying the successive shortest path minimum cost flow algorithm [4] on this network will be $O(|V|^2.|E|) = O(n^3.|Q|^3 d)$ while the search for a shortest path using Bellman-Ford will be in $O(|V|.|E|) = O(n^2.|Q|^2 d)$. According to [19,18], this means that the sole search for a full support will be in $O(n^3.|Q|^2.d(d+|Q|))$. Enforcing EDGAC* requires to simultaneously have simple, full and existential supports. Just considering full supports, this means that the complexity for enforcing EDGAC* exceeds $O(\max(nd, k).n^5.|Q|^2.d(d + |Q|))$. Assuming $\frac{|Q|}{|\Sigma|}$ is in $O(1)$, this is in $O(\max(nd, k).n^5.d^4)$ compared to the $O(\max(nd, k).nd^3)$ obtained through decomposition. This corroborates our experimental results but the strength of this comparison is largely conditionned by the tightness of the bounds presented in [19,18]. Another reason for the impressive speedups we observe may also lies in the non optimality of the global filtering algorithms for SoftRegular.

Conclusion

In this paper, we have extended the idea of constraint decomposition to cost functions occurring in CFNs. For cost functions having a Berge-acyclic decompositions, we have shown that even relatively simple filtering, at the directed arc consistency level, provide a comparable filtering when they are applied onto the decomposition or on the global cost function itself, provided a suitable variable ordering is used. For the stronger consistency Virtual AC filtering, the same result is obtained, without any requirement on variable ordering.

An example of decomposable cost function that has been considered in the paper is the WeightedRegular cost function. This cost function, based on weighted automata, allows to directly decompose the SoftRegular constraint using either the Hamming or the Edit distance[7].

The results presented in this paper are still preliminary. There are at least three directions that are worth exploring here. First, one should consider other decomposable cost functions beyond just WeightedRegular. Several related constraints, including ContextFreeGrammar, Among, ... have Berge-acyclic decompositions and their cost functions variants likely have related Berge-acyclic decompositions, allowing for a simple extension of WCSP solvers to more extensive Cost Function Programming tools.

Beyond this, more experiments should be performed on more complex problems such as Nurse Rostering problems [20]. Finally, the strength and the simplicity of the result obtained for this local consistency plead for experiments using Virtual AC. This was not possible at the time this paper was written as the only implementations of VAC we know are restricted to binary cost functions.

Although restricted to Berge-acyclic decompositions, this work paves the way for a more general form of "structural decompositions" where global cost functions decompose into an acyclic structure of local cost functions combined with operators that could be different from $\oplus$, with bounded separator sizes (but not necessarily cardinality 1). For various operators, these global structurally decomposed cost functions could then be propagated efficiently through non serial dynamic programming (elimination) approaches.

References

1. Beeri, C., Fagin, R., Maier, D., M.Yannakakis: On the desirability of acyclic database schemes. Journal of the ACM 30, 479–513 (1983)
2. Bessiere, C., Hebrard, E., Hnich, B., Kiziltan, Z., Quimper, C., Walsh, T.: Reformulating global constraints: the slide and regular constraints. Abstraction, Reformulation, and Approximation pp. 80–92 (2007)
3. Boros, E., Hammer, P.: Pseudo-Boolean Optimization. Discrete Appl. Math. 123, 155–225 (2002)
4. Busacker, R., Gowen, P.: A procedure for determining minimal-cost network flow patterns. In: ORO Technical Report 15 (1961)
5. Cooper, M., de Givry, S., Sanchez, M., Schiex, T., Zytnicki, M., Werner, T.: Soft arc consistency revisited. Artificial Intelligence 174, 449–478 (2010)
6. Cooper, M.C.: Reduction operations in fuzzy or valued constraint satisfaction. Fuzzy Sets and Systems 134(3), 311–342 (2003)

[7] A result that could not be achieved in [2]. Note however that [13] also shows how the WeightedCFG soft constraint can encode Edit distance using ε-transitions.

7. Cooper, M.C., Schiex, T.: Arc consistency for soft constraints. Artificial Intelligence 154(1-2), 199–227 (2004)
8. Demassey, S., Pesant, G., Rousseau, L.M.: A cost-regular based hybrid column generation approach. Constraints 11(4), 315–333 (2006)
9. Favier, A., de Givry, S., Legarra, A., Schiex, T.: Pairwise decomposition for combinatorial optimization in graphical models. In: Proc. of IJCAI'11. Barcelona, Spain (2011)
10. Gent, I.P. (ed.): Principles and Practice of Constraint Programming - CP 2009, 15th International Conference, CP 2009, Lisbon, Portugal, September 20-24, 2009, Proceedings, Lecture Notes in Computer Science, vol. 5732. Springer (2009)
11. van Hoeve, W.J., Pesant, G., Rousseau, L.M.: On global warming: Flow-based soft global constraints. J. Heuristics 12(4-5), 347–373 (2006)
12. II, K.C., Kari, J.: Image compression using weighted finite automata. In: Borzyszkowski, A.M., Sokolowski, S. (eds.) MFCS. Lecture Notes in Computer Science, vol. 711, pp. 392–402. Springer (1993)
13. Katsirelos, G., Narodytska, N., Walsh, T.: The weighted grammar constraint. Annals OR 184(1), 179–207 (2011)
14. Koller, D., Friedman, N.: Probabilistic graphical models. MIT press (2009)
15. Lagerkvist, M.Z., Schulte, C.: Propagator groups. In: Gent [10], pp. 524–538
16. Larrosa, J., de Givry, S., Heras, F., Zytnicki, M.: Existential arc consistency: getting closer to full arc consistency in weighted CSPs. In: Proc. of the 19^{th} IJCAI. pp. 84–89. Edinburgh, Scotland (Aug 2005)
17. Larrosa, J., Schiex, T.: Solving weighted CSP by maintaining arc consistency. Artif. Intell. 159(1-2), 1–26 (2004)
18. Lee, J., Leung, K.L.: A stronger consistency for soft global constraints in weighted constraint satisfaction. In: Fox, M., Poole, D. (eds.) AAAI. AAAI Press (2010)
19. Lee, J.H.M., Leung, K.L.: Towards efficient consistency enforcement for global constraints in weighted constraint satisfaction. In: Boutilier, C. (ed.) IJCAI. pp. 559–565 (2009)
20. Métivier, J.P., Boizumault, P., Loudni, S.: Solving nurse rostering problems using soft global constraints. In: Gent [10], pp. 73–87
21. Mohri, M.: Edit-distance of weighted automata: General definitions and algorithms. International Journal of Foundations of Computer Science 14(6), 957–982 (2003)
22. Papadimitriou, C., Yannakakis, M.: Optimization, approximation, and complexity classes. Journal of Computer and System Sciences 43(3), 425–440 (1991)
23. Pesant, G.: A regular language membership constraint for finite sequences of variables. In: Wallace, M. (ed.) CP. Lecture Notes in Computer Science, vol. 3258, pp. 482–495. Springer (2004)
24. Sánchez, M., de Givry, S., Schiex, T.: Mendelian error detection in complex pedigrees using weighted constraint satisfaction techniques. Constraints 13(1-2), 130–154 (2008)
25. Schiex, T., Fargier, H., Verfaillie, G.: Valued constraint satisfaction problems: hard and easy problems. In: Proc. of the 14^{th} IJCAI. pp. 631–637. Montréal, Canada (Aug 1995)
26. Zytnicki, M., Gaspin, C., Schiex, T.: A new local consistency for weighted CSP dedicated to long domains. In: Proceedings of the 2006 ACM Symposium on Applied Computing (SAC). pp. 394–398. Dijon, France (Apr 2007)