Improving Tree Decomposition Methods With Function Filtering

Martí Sánchez[1], **Javier Larrosa**[2] **and Pedro Meseguer**[1]
[1]IIIA-CSIC, Campus UAB, 08193 Bellaterra, Spain
[2]Dep. LSI, UPC, Jordi Girona 1-3, 08034 Barcelona, Spain
marti@iiia.csic.es, larrosa@lsi.upc.es, pedro@iiia.csic.es

Abstract

Tree decomposition can solve weighted CSP, but with a high spatial complexity. To improve its practical usage, we present function filtering, a strategy to decrease memory consumption. Function filtering detects and removes some tuples that appear to be consistent but that will become inconsistent when extended to other variables. We show empirically the benefits of our approach.

1 Introduction

A weigthed CSP (WCSP) is defined as $\langle X, D, C, S(k)\rangle$ where X and D are variables and domains as in CSP. C is a finite set of constraints as cost functions; $f \in C$ relates some variables $var(f) = \{x_{i_1}, \ldots x_{i_r}\}$ called its scope, and assigns costs to tuples $t \in \prod_{x_i \in var(f)} D_i$ such that,

$$f(t) = \begin{cases} 0 & \text{if } t \text{ is allowed} \\ 1 \ldots k-1 & \text{if } t \text{ is partially allowed} \\ k & \text{if } t \text{ is totally forbidden} \end{cases}$$

$S(k) = \langle [0, 1, ..., k], \oplus, \geq \rangle$ is a valuation structure such that $a \oplus b = min\{k, a + b\}$, $\top = k$, $\bot = 0$ [Larrosa, 2002]. We assume that the reader is familiar with assignments or value tuples t_S with scope S, complete tuples ($S = X$), projections over $S' \subset S$, $t_S[S']$, and concatenation of two tuples $t_S \cdot t'_T$, defined only if common variables coincide in their corresponding values. We assume that $f(t_S)$ (with $var(f) \subset S$) always means $f(t_S[var(f)])$. A complete tuple t_S is *consistent* if $\bigoplus_{f \in C} f(t) < k$, else t_S is *inconsistent*. A *solution* is a complete consistent assignment with minimum cost. Finding a solution is NP-hard. With $k = 1$ WCSP reduces to CSP.

We define two operations on functions. *Projecting out* a variable $x \in var(f)$ from f, denoted $f_{\Downarrow x}$, is a new function with scope $var(f) - x$ defined as $f_{\Downarrow x}(t) = \min_{a \in D_x}(f(a.t))$. *Summing* two functions f and g is a new function $f + g$ with scope $var(f) \cup var(g)$ and $\forall t \in \prod_{x_i \in var(f)} D_i$, $\forall t' \in \prod_{x_j \in var(g)} D_j$ such that $t.t'$ is defined, $(f+g)(t.t') = f(t) \oplus g(t')$. We say that function g is a *lower bound* of f, denoted $g \leq f$, if $var(g) \subseteq var(f)$ and for all possible tuples t of f, $g(t) \leq f(t)$. A set of functions G is a *lower bound* of f iff $(\sum_{g \in G} g) \leq f$. It is easy to check that for any f, $(f_{\Downarrow x})$ is a lower bound of f, and $\sum_{f \in F}(f_{\Downarrow x}) \leq (\sum_{f \in F} f)_{\Downarrow x}$.

procedure CTE($\langle X, D, C, k\rangle$, $\langle\langle V, E\rangle, \chi, \psi\rangle$)
1 **for each** $(u, v) \in E$ s.t. all $m_{(i,u)}, i \neq v$ have arrived **do**
2 $\quad B \leftarrow \psi(u) \cup \{m_{(i,u)} \mid (i, u) \in E, i \neq v\}$;
3 $\quad m_{(u,v)} \leftarrow (\sum_{f \in B} f) \Downarrow_{elim(u,v)}$;
4 $\quad$ send $m_{(u,v)}$;

Figure 1: The CTE algorithm.

A *tree decomposition* of a WCSP $\langle X, D, C, S(k)\rangle$ is a triplet $\langle T, \chi, \psi\rangle$, where $T = \langle V, E\rangle$ is a tree, χ and ψ are labeling functions which associate with each vertex $v \in V$ two sets, $\chi(v) \subseteq X$ and $\psi(v) \subseteq C$ such that: (i) for each function $f \in C$, there is exactly one vertex $v \in V$ such that $f \in \psi(v)$ and $var(f) \subseteq \chi(v)$; (ii) for each variable $x \in X$, the set $\{v \in V | x \in \chi(v)\}$ induces a connected subtree of T. The *tree-width* of a tree decomposition is $tw = max_{v \in V}|\chi(v)| - 1$. If $(u, v) \in E$ the *separator* is $sep(u, v) = \chi(u) \cap \chi(v)$, and the *eliminator* is $elim(u, v) = \chi(u) - sep(u, v)$ [Dechter, 2003].

Cluster-Tree Elimination (CTE) is an algorithm that solves WCSP by sending messages along tree decomposition edges [Dechter and Pearl, 1989]. Edge $(u, v) \in E$ has associated two CTE messages $m_{(u,v)}$, from u to v, and $m_{(v,u)}$, from v to u. $m_{(u,v)}$ is a function computed summing all functions in $\psi(v)$ with all incoming CTE messages except from $m_{(v,u)}$ and projecting out variables in $elim(u, v)$. $m_{(u,v)}$ has scope $sep(u, v)$. The CTE algorithm appears in Figure 1. Its complexity is time $O(d^{tw+1})$ and space $O(d^s)$, where d is the largest domain size and s is the maximum separator size.

Mini-Cluster-Tree Elimination (MCTE(r)) approximates CTE. If the number of variables in a cluster is high, it may be impossible to compute $m_{(u,v)}$ due to memory limitations. MCTE(r) computes a lower bound by limiting by r the arity of the functions sent in the messages. A MCTE(r) message, $M_{(u,v)}$, is a set of functions that approximate the corresponding CTE message $m_{(u,v)}$ ($M_{(u,v)} \leq m_{(u,v)}$). It is computed as $m_{(u,v)}$ but instead of summing all functions of set B, it computes a partition $P = \{B_1, B_2, \ldots, B_p\}$ of B such that the sum of the functions in every B_i does not exceed arity r. The MCTE(r) algorithm, with time and space complexity $O(d^r)$, is obtained replacing line 3 of CTE by,

3.1 $\quad P \leftarrow partitioning(B, r)$;
3.2 $\quad M_{(u,v)} \leftarrow \{(\sum_{f \in B_i} f) \Downarrow elim(u, v) | B_i \in P\}$;

2 Function Filtering

A *nogood* is a tuple t that cannot be extended into a complete consistent assignment. Nogoods are useless for solution generation, so they can be eliminated as soon as are detected. We store function f as a set S_f containing all pairs $(t, f(t))$ with cost less than k. The *size* of f, denoted $|f|$, is the number of tuples with cost less than k. Computing $f \Downarrow_x$ has time complexity $O(|f|)$. For summing $f + g$, we iterate over all the combinations $(t, f(t)) \in S_f$ and $(t', g(t')) \in S_g$ and, if they match, compute $(t \cdot t', f(t) \oplus g(t'))$, which has complexity $O(|f||g|)$. Efficiency of operations depends on function size. To anticipate the detection of nogoods, we propose *function filtering* on f from a set of functions H, noted $\overline{f}^H$, as

$$\overline{f}^H(t) = \begin{cases} f(t) & \text{if } (\bigoplus_{h \in H} h(t)) \oplus f(t) < k \\ k & \text{otherwise} \end{cases}$$

Tuples reaching the upper bound k are removed, which causes to reduce $|f|$ before operating with it. Therefore, let f (resp. g) be a function and F (resp. G) be a lower bound. When summing f and g, if previously filtered with the lower bound of the other function, the result is preserved,

$$\overline{f}^G + \overline{g}^F = f + g$$

and the sum is done with functions of smaller size, so it is presumably done more efficiently. If $f \notin H, g \notin H$,

$$\overline{f + g}^H = \overline{\overline{f}^H + \overline{g}^H}^H$$

Function filtering easily integrates into CTE. We define a *filtering tree-decomposition* $\langle T, \chi, \psi, \phi \rangle$, where $\phi(u, v)$ is a set of functions associated to edge $(u, v) \in E$ with scope included in $sep(u, v)$. $\phi(u, v)$ must be a lower bound of the corresponding $m_{(u,v)}$ (namely, $\phi(u, v) \leq m_{(u,v)}$). The new algorithms CTEf and MCTEf(r) use a filtering tree decomposition. They are equivalent to CTE and MCTE(r) except in that they use $\phi(u, v)$ for filtering functions before computing $m_{(u,v)}$ or $M_{(u,v)}$. For CTEf, we replace line 3 by,

3 $\quad m_{(u,v)} \leftarrow \overline{\sum_{f \in B} f}^{\phi(u,v)} \Downarrow_{elim(u,v)}$;

Similarly for MCTEf(r) we replace line 3 by two lines,

3.1 $\quad P \leftarrow \overline{partitioning(B, r)}^{\phi(u,v)}$;

3.2 $\quad M_{(u,v)} \leftarrow \{(\sum_{f \in B_i} f) \Downarrow_{elim(u,v)} | B_i \in P\}$;

The effectiveness of the new algorithms will depend on the quality of the lower bound $\phi(u, v)$. It is worth noting that the algorithms will be correct as long as $\phi(u, v)$ is a true lower bound which can be computed using any technique. An option is to include in $\phi(u, v)$ all the original functions used to compute $m_{(v,u)}$ properly projected,

$$\phi(u, v) = \{f \Downarrow_S | f \in \psi(w), w \in T(u, v), S = var(f) - \chi(u)\}$$

Our CTEf and MCTEf implementations use this lower bound. $T(u, v)$ denote the nodes of the tree decomposition reachable from node v after the eliminaiton of edge (u, v). Another option for CTEf is to include in $\phi(u, v)$ a message $M_{(v,u)}$ from a previous execution of MCTE(r). Applying this idea to MCTEf, we obtain a recursive algorithm which naturally produces an elegant iterative approximating method called IMCTEf (Figure 2). It executes MCTEf(r) using as lower bounds $\phi(u, v)$ the messages $M^{r-1}_{(v,u)}$ computed by MCTEf($r - 1$) which, recursively, uses the messages $M^{r-2}_{(v,u)}$ computed by MCTEf($r - 2$), an so on.

procedure IMCTE($\langle X, D, C, k \rangle$, $\langle \langle V, E \rangle, \chi, \psi \rangle$)
  for each $(u, v) \in E$ **do** $\phi(u, v) := \{\emptyset\}$; $r := 1$;
  repeat
    MCTEf(r); $r := r + 1$;
    for each $(u, v) \in E$ **do** $\phi(u, v) := M_{(u,v)}$;
  until exact solution or exhausted resources

Figure 2: The IMCTE algorithm.

State of the art CTE based algorithms assume to consume all d^s memory. Here we show that storing only consistent tuples and applying filtering techniques we can greatly reduce memory usage. Experiments focus on (i) showing that CTEf uses less memory than CTE to find the exact solution, and (ii) showing that MCTEf(r), exhausts resources at a smaller arity r and finds worst LB than the iterative version IMCTE. We have tested CTE, CTEf, MCTEf(r) and IMCTE on DIMACS dubois Max-Sat instances, Borchers Weigthed Max-Sat instances and SPOT instances. When d^s is small CTE is feasible. When d^s is large, CTE is unfeasible and filtering is very useful as shown by the gain in one order of magnitude in the used memory (see the first Borchers and the firsts SPOT instances). In unfeasible instances for both CTE and CTEf, IMCTE finds better quality lower bounds than MCTEf(r).

References

[Dechter and Pearl, 1989] R. Dechter and J. Pearl. Tree clustering for constraints networks. *Artifical Intelligence*, 38, 1989.

[Dechter, 2003] R Dechter. *Constraint Processing*. Elsevier, 2003.

[Larrosa, 2002] J. Larrosa. Node and arc consistency in weighted csp. In *Proc. AAAI*, 2002.

							MCTEf(r)		IMCTE		
	$\|X\|$	$\|C\|$	d	sep	CTE	CTEf	r	LB	r	LB	UB
dubois100	75	200	2	3	3k	2k					1
wp2100	50	95	2	9	6k	1k					16
wp2150	50	138	2	15	302k	40k					34
wp2200	50	186	2	19	-	733k					69
wp2250	50	233	2	24	-	-	21	69	25	96	96
wp2300	50	261	2	26	-	-	21	76	21	132	132
wp2350	50	302	2	30	-	-	21	129	21	155	212
wp2400	50	340	2	30	-	-	20	70	20	137	212
wp2450	50	378	2	31	-	-	20	130	20	187	257
wp2500	50	418	2	34	-	-	20	168	20	251	318
spot54	67	271	4	11	754k	16k					37
spot29	82	462	4	14	-	63k					8059
spot503	143	635	4	8	-	26k					11113
spot505	240	2242	4	22	-	-	12	5020	12	14188	21254
spot42	190	1394	4	26	-	-	12	109000	12	123050	155051

Table 1: Columns are: instance, number of variables, number of constraints, maximum domain size, maximum separator size, tuples consumed by CTE algorithm, tuples consumed by CTEf algorithm (- denotes exhausted memory), arity r reached by MCTE(r), LB computed by MCTE(r), arity r reached by IMCTE, LB computed by IMCTE (before resources exhausted), optimal UB of the problem.