

Exploiting Tree Decomposition and Soft Local Consistency in Weighted CSP

Simon de Givry and Thomas Schiex

INRA, Toulouse, France
{degivry,tschiex}@toulouse.inra.fr

Gerard Verfaillie

ONERA - DCSD, Toulouse, France
verfaillie@cert.fr

Abstract

Several recent approaches for processing graphical models (constraint and Bayesian networks) simultaneously exploit graph decomposition and local consistency enforcing. Graph decomposition exploits the problem structure and offers space and time complexity bounds while hard information propagation provides practical improvements of space and time behavior inside these theoretical bounds.

Concurrently, the extension of local consistency to weighted constraint networks has led to important improvements in branch and bound based solvers. Indeed, soft local consistencies give incrementally computed strong lower bounds providing inexpensive yet powerful pruning and better informed heuristics.

In this paper, we consider combinations of tree decomposition based approaches and soft local consistency enforcing for solving weighted constraint problems. The intricacy of weighted information processing leads to different approaches, with different theoretical properties. It appears that the most promising combination sacrifices a bit of theory for improved practical efficiency.

Introduction

Graphical model processing is a central problem in AI. In the last years, in order to solve satisfaction, optimization or counting problems, several algorithms have been proposed that simultaneously exploit a decomposition of the graph of the problem and the propagation of hard information using local consistency enforcing. This includes algorithms such as Recursive Conditioning (RC) (Darwiche 2001), Backtrack bounded by Tree Decomposition (BTD) (Terrioux & Jegou 2003), AND-OR tree and graph search (Marinescu & Dechter 2005b; 2005a), all related to Pseudo-tree search (Freuder & Quinn 1985).

Implicitly or not, all these algorithms exploit the properties of Tree Decompositions (Bodlaender 2005) which capture structural independence information, in order to obtain theoretical bounds on the time and space complexity of the algorithms. The combination with hard information propagation provides additional pruning leading to im-

proved practical time and space complexity inside the original bounds. Note however that in the context of tree-search, these bounds are obtained at the cost of restricted freedom in variable assignment ordering (see (Bacchus, Dalmao, & Pitassi 2003) on this topic).

In this paper, we are specifically interested in optimization problems in the framework of valued and weighted constraint networks (Schiex, Fargier, & Verfaillie 1995). Weighted constraint networks (WCN) provide a very general model with several applications in domains such as *resource allocation*, *combinatorial auctions* and *bioinformatics*.

In a first part, we show that, with a limited weakening of existing theoretical complexity bounds, an alternative combination of tree decomposition and branch and bound can be defined. In a second part, we exploit the extension of local consistency to WCN. This extension has led to increasingly efficient branch and bound algorithms using increasingly strong local consistency properties (Cooper & Schiex 2004; Larrosa & Schiex 2004; 2003). They offer incrementally maintained strong bounds and also contribute to better informed variable and value ordering. Introduced in tree decomposition based approaches, they should enhance their practical time and space complexities and also provide better guidance.

Taken together, these two parts show the increased intricacy of weighted information processing, leading to different possible combinations, each having different theoretical properties and practical efficiencies.

Preliminaries

A weighted binary CSP (WCSP) is a triplet (X, D, W) . $X = \{1, \dots, n\}$ is a set of n variables. Each variable $i \in X$ has a finite domain $D_i \in D$ of values that can be assigned to it. The maximum domain size is d . W is a set of soft constraints (or cost functions). A binary soft constraint $W_{ij} \in W$ is a function $W_{ij} : D_i \times D_j \mapsto [0, k]$ where k is a given maximum integer cost corresponding to a completely forbidden assignment (expressing hard constraints). If they do not exist, we add to W one unary cost function for every variable such that $W_i : D_i \mapsto [0, k]$ and a zero arity constraint W_\emptyset (a constant cost paid by any assignment). All these additional cost functions have initial value 0, leaving the semantics of the problem unchanged.

The problem is then to find a complete assignment

with a minimum cost: $\min_{(a_1, a_2, \dots, a_n) \in \prod_i D_i} \{W_\emptyset + \sum_{i=1}^n W_i(a_i) + \sum_{W_{ij} \in W} W_{ij}(a_i, a_j)\}$, an optimization problem with an associated NP-complete decision problem.

The constraint graph of a WCSP is a graph $G = (X, E)$ with one vertex for each variable and one edge (i, j) for every binary constraint $W_{ij} \in W$. A tree decomposition of this graph is defined by a tree (C, T) . The set of nodes of the tree is $C = \{C_1, \dots, C_m\}$ where C_e is a set of variables ($C_e \subset X$) called a cluster. T is a set of edges connecting clusters and forming a tree (a connected acyclic graph). The set of clusters C must cover all the variables ($\bigcup_{C_e \in C} C_e = X$) and all the constraints ($\forall (i, j) \in E, \exists C_e \in C \text{ s.t. } i, j \in C_e$). Furthermore, if a variable i appears in two clusters C_e and C_g , i must also appear in all the clusters C_f on the unique path from C_e to C_g in (C, T) .

The rationale behind this definition is that acyclic problems can be solved with a good theoretical complexity. Thus, a tree decomposition decomposes a problem in sub-problems (clusters) organized in an acyclic graph and in such a way that the variables that relate two adjacent clusters (variables whose removal disconnects the subproblems) are obtained by intersecting the two clusters.

This is illustrated on Figure 1 where the graph of a frequency assignment problem is covered by clusters defining a tree decomposition. Because of the usual emphasis on purely random problems, one may think that such nice decompositions are exceptional but the emphasis on modularity in design problems and the (spatial or temporal) locality of many problems easily induce such structures.

The tree-width of a tree decomposition (C, T) is equal to $\max_{C_e \in C} \{|C_e|\} - 1$, denoted w in the sequel. The tree-width w^* of G is the minimum tree-width over all tree decompositions of G (also called the induced-width of G). If a root $C_r \in C$ is chosen, the maximum number of variables appearing in a path starting from C_r is called the tree-height of the decomposition. Finding a minimum tree-width or a minimum tree-height decomposition is NP-hard.

Exploiting tree decompositions

For a given WCSP, we consider a rooted tree decomposition (C, T) with an arbitrary root C_1 . We denote by $Father(C_e)$ (resp. $Sons(C_e)$) the parent (resp. set of sons) of C_e in T . The separator of C_e is the set $S_e = C_e \cap Father(C_e)$. The set of proper variables of C_e is $V_e = C_e \setminus S_e$. Note that the V_e family defines a partition of X . For a given variable $i \in X$, we denote by $[i]$ the index of the cluster such that $i \in V_{[i]}$.

The essential property of tree decomposition is that assigning S_e separates the initial problem in two subproblems which can then be solved independently. The first subproblem, denoted P_e , is defined by the variables of C_e and all its descendant clusters in T and by all the cost functions involving at least one proper variable of these clusters. The remaining constraints, together with the variables they involve, define the remaining subproblem.

This property has been exploited by many related algorithms. In a branch and bound framework, this property may be exploited by restricting the variable ordering. Imagine all

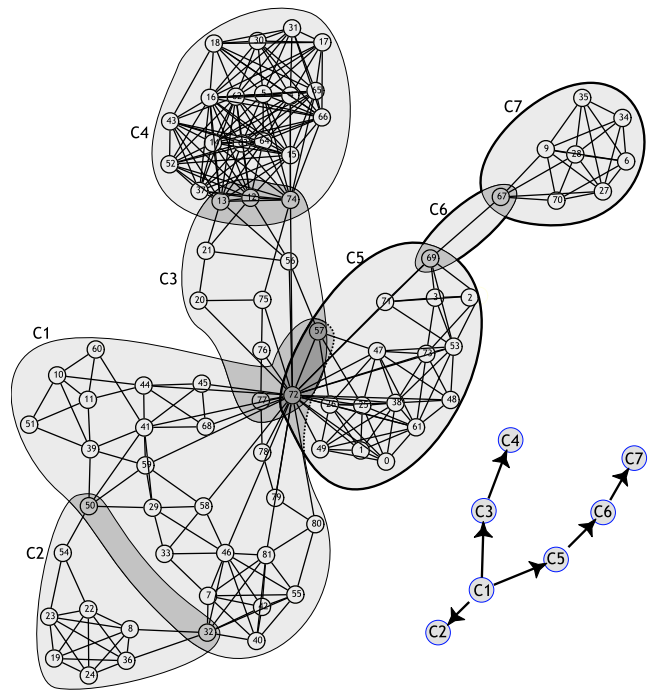


Figure 1: The constraint graph of preprocessed RLFAP SCEN-06 problem covered by clusters and the associated tree decomposition using C_1 as root. Problem P_5 is outlined with separator $S_5 = \{57, 72\}$. The constraint $W_{57,72}$ inside S_5 is not part of P_5 . P_6 has a separator $S_6 = \{69\}$.

the variables of a cluster C_e are assigned before any of the remaining variables in its son clusters and consider a current assignment A . Then, for any cluster $C_f \in Sons(C_e)$, and for the current assignment A_f of the separator S_f , the subproblem P_f under assignment A_f (denoted P_f/A_f) can be solved independently from the rest of the problem. If memory allows, the *optimal cost* of P_f/A_f may be recorded which means it will never be solved again for the same assignment of S_f .

If d is the maximum domain size, h the decomposition tree-height and w the tree-width, this idea applied recursively with no recording guarantees that a cluster C_e will never be visited more than d to the power of the number of variables in the path from the root to C_e (proper variables excluded). This gives a guaranteed bound on the number of visited nodes which is dominated by $O(d^h)$. This covers Pseudo-Tree or AND-OR tree search, also applied to WCSP in (Larrosa, Meseguer, & Sanchez 2002; Marinescu & Dechter 2005b).

With recording, a cluster C_e will never be visited more than the number of assignments of S_e . Given the number of variables in V_e , this means that the number of nodes is dominated by $O(d^{w+1})$, $w + 1$ being the size of the largest cluster. This gives algorithms such as RC with full recording, BTM and AND-OR graph search, which can be considered as structural learning algorithms. The side effect is that the memory required may be exponential in the separator size.

Exploiting better upper bounds

Traditional depth-first branch and bound algorithms are tree-search algorithms exploiting two bounds. In the WCSP case, where a minimum cost solution is sought, the best known solution so far provides an upper bound on the optimum. During search, at a given node, an extra dedicated lower bounding mechanism provides a lower bound on the cost of any complete assignment below the current node. If the lower bound exceeds the upper bound, one can backtrack. Typically, when search starts, the upper bound is either set to maximum cost k or to the cost of a solution found e.g. by local search.

With the basic combination of tree decomposition with branch and bound, as in BTD applied to optimization problems (Jegou & Terrioux 2004), once a separator S_e is assigned, the corresponding problem P_e/A_e is solved as an independent problem. This means that its initial upper bound is set to k .

So, existing algorithms such as BTD or AND-OR search do not impose any initial upper bound when solving a subproblem. Although this effectively guarantees that the optimum of P_e/A_e will be computed, it provides limited pruning. Furthermore, this optimal cost, once combined with the cost of already totally assigned clusters and with existing lower bounds on the remaining unexplored clusters may well exceed the cost of the best solution found so far: useless optimization has been performed.

It may therefore be better to start solving P_e/A_e with a non trivial upper bound obtained by taking the difference between the upper bound associated with the parent problem $P_{Father(C_e)}$ and a lower bound on $P_{Father(C_e)}$ but P_e . However, this has a bad side-effect: after solving a subproblem, if a solution with a cost strictly lower than the initial upper bound is found, this solution is optimal and can be recorded as such. But if no solution is found, we just have a new lower bound on the optimum of P_e/A_e . The lower bound and the fact it is optimal can be recorded in LB_{P_e/A_e} and Opt_{P_e/A_e} respectively, initially set to 0 and *false*.

If the search comes back later to the problem P_e/A_e , and even if this lower bound has been recorded, the subproblem may need to be solved again. However, since the lower bound recorded will necessarily be improved at each search, the number of successive resolutions is bounded by k (and more tightly by the optimum of P_e/A_e).

The resulting algorithm has the same space complexity as BTD and the number of visited nodes is bounded by $O(k \cdot d^{w+1})$ where w is the tree-width of the decomposition used. Note that because it still exploits problem decomposition, the number of nodes is also dominated by $O(d^h)$ where h is the decomposition tree-height (thus still better than branch and bound in $O(d^n)$).

Local consistency and tree decomposition

Independently of tree decomposition approaches, the algorithmics of WCSP has been widely improved in the last years by exploiting the extension of local consistency to cost functions. Several increasingly stronger local consistencies have been defined since then (soft node and arc con-

sistency (Larrosa & Schiex 2004), full directional arc consistency (DAC/FDAC) (Larrosa & Schiex 2003; Cooper & Schiex 2004)), leading to increasingly efficient branch and bound algorithms.

As in the classical case, enforcing local consistency on the initial problem provides in polynomial time an *equivalent* problem – defining the same cost distribution on complete assignments – with possible smaller domains. It may also increase the value of W_\emptyset and the costs $W_i(a)$ associated with domain values. W_\emptyset defines a strong lower bound which can be exploited by branch and bound algorithms while the updated $W_i(a)$ can inform variable and value orderings.

For our purpose, we point out that enforcing such local consistencies is done by the repeated application of atomic operations called *arc equivalence preserving transformations* (Cooper & Schiex 2004). If we consider two cost functions W_{ij} and W_i , a value $a \in D_i$ and a cost α , we can add α to $W_i(a)$ and subtract α from every $W_{ij}(a, b)$ for all $b \in D_j$. Simple arithmetics shows that the global cost distribution is unchanged while costs may have moved from the binary to the unary level (if $\alpha > 0$, this is called a Projection) or from the unary to the binary level (if $\alpha < 0$, this is called an Extension). In these operations, any cost above k , the maximum allowed cost, can be considered as infinite and is thus unaffected by subtraction. If no negative cost appears and if all costs above k are set to k , the remaining problem is always a valid and equivalent WCSP.

The same mechanism, at the unary level, can be used to move costs from the W_i to W_\emptyset . Finally, any value a such that $W_i(a) + W_\emptyset$ is equal to k can be deleted.

From this description, one may already see that, even if it preserves the cost of complete assignments, local consistency may move costs between clusters, thereby invalidating previously recorded information.

Binary projection and extension Moving costs between cost functions W_{ij} and $W_i(a)$ may modify the cost distributions (and therefore the recorded optima) of subproblems. This happens only when W_i and W_{ij} are associated to different subproblems, such as $W_{72} \in C_1$ and $W_{72,69} \in C_5$ in Fig. 1, and thus when the first cluster from the root containing i , $C_{[i]}$, is an ascendant of $C_{[j]}$. In this case, the cost distribution of $P_{[j]}$ and of all ascendant subproblems up to (but not including) $P_{[i]}$ is decreased or increased (according to α 's sign).

We store these cost modifications in a specific *backtrackable* data structure $\Delta W_i^e(a)$. There is one count for every value of every variable in every separator, resulting in a reasonable $O(mnd)$ space complexity.

Initially, ΔW is set to zero. When an arc equivalence preserving operation is performed as above, all the $\Delta W_i^e(a)$ counters for all separators S_e , from $S_{[j]}$ up to $S_{[i]}$ (excluded) are updated: $\Delta W_i^e(a) := \Delta W_i^e(a) + \alpha$.

During the search, when a separator S_e is assigned by assignment A_e , we can obtain the total cost that has been moved out of the subproblem by summing up all the $\Delta W_i^e(a)$ for all values (i, a) in the separator assignment A_e and correct any recorded information. This summa-

tion is denoted $\overline{\Delta W}_{P_e/A_e}$ in the following. For instance, in Fig. 1, $\overline{\Delta W}_{P_5/A_5} = \Delta W_{57}^5(a) + \Delta W_{72}^5(b)$ with $A_5 = \{(57, a), (72, b)\}$.

Unary projection Local consistency enforcing uses a similar mechanism to move costs from unary cost functions W_i to the problem lower bound W_\emptyset . In order to avoid corresponding subproblem cost modification, we split W_\emptyset into local W_\emptyset^e cost functions, each associated to one cluster. When cost from a unary constraint W_i is moved to the zero arity level, it is moved to $W_\emptyset^{[i]}$. For a given subproblem P_e , its associated lower bound is obtained by summing up all local W_\emptyset^f for all the clusters C_f in P_e defining a local lower bound denoted \overline{W}_\emptyset^e . For the complete problem, this local lower bound is equal to the global lower bound W_\emptyset .

For any strict subproblem $P_e, e \in \{2, \dots, m\}$ and any assignment A_e , we may therefore have two lower bounds, one provided by the lower bound recording mechanism ($LB_{P_e/A_e} - \overline{\Delta W}_{P_e/A_e}$) and the other one provided by local consistency (\overline{W}_\emptyset^e). In the following, we use the maximum of these two bounds, denoted as $lb(P_e/A_e)$. We generalize the previous definition for any subproblem $P_e, e \in \{1, \dots, m\}$ and any partial assignment A : $lb(P_e/A) = W_\emptyset^e + \sum_{C_f \in \text{Sons}(C_e)} lb(P_f/A_f)$, taking into account local consistency and recorded lower bounds as soon as their separator is completely assigned by A .

For instance, in Fig. 1, let $A = \{(57, a), (72, b), (69, c)\}$, then $lb(P_5/A) = W_\emptyset^5 + lb(P_6/A_6) = W_\emptyset^5 + \max\{\overline{W}_\emptyset^6, LB_{P_6/A_6} - \overline{\Delta W}_{P_6/A_6}\} = W_\emptyset^5 + \max\{W_\emptyset^6 + LB_{P_6/A_6} - \Delta W_{69}^6(c)\}$ with $A_6 = \{(69, c)\}$.

Value removal based on local cuts The last mechanism used in local consistency enforcing is value removal. Here, any value $a \in D_i$ such that $W_\emptyset + W_i(a)$ is larger than the global upper bound gub (the best known solution cost) can be removed. For a subproblem P_e , we replace this global condition by a local one $\overline{W}_\emptyset^e + W_i(a) \geq cub$ where \overline{W}_\emptyset^e and cub are the lower and upper bounds of the currently solved problem P_e . A better pruning rule can be deduced by taking into account local consistency and recorded information as in $lb(P_e/A)$ but without using LB_{P_f/A_f} if P_f includes value a (not counting twice the same cost $W_i(a)$). We apply this pruning rule only to variables in the current subproblem P_e .

By doing so, we ensure that any value removed in a given subproblem P_e is also forbidden in every subproblem P_f included in P_e . Thus the set of removed values for one subproblem P_e , is still valid in all the subproblems P_f such that $C_f \in \text{Sons}(C_e)$, and all the propagations made during the exploration of C_e are still valid when exploring $C_f \in \text{Sons}(C_e)$. The proof follows from the fact that the difference between the current upper and lower bounds is monotonically decreasing along a path from the root to a leaf of the cluster tree.

For instance, in Fig. 1, let P_5 be the current subproblem and $A = \{(57, a), (72, b), (69, c)\}$ be the current partial assignment, any value $u \in D_i$ with variable $i \in V_5$ is removed

if $lb(P_5/A) + W_i(u) \geq cub$ and any value $v \in D_j$ with variable $j \in V_6 \cup V_7$ is removed if $\overline{W}_\emptyset^5 + W_j(v) \geq cub$.

Value removal based on local and global cuts Inside a subproblem, a value can be removed for local reasons if $\overline{W}_\emptyset^e + W_i(a) \geq cub$ but also, as in traditional branch and bound, for global reasons, if $W_\emptyset + W_i(a) \geq gub$. Because neither of these conditions includes the other, they can both be useful. However, the global condition may prune the exploration of cluster C_e to the point that we first lose the guarantee that if a solution is found, it is optimal, but also the guarantee that an improved lower bound can be deduced from the absence of solution. Even if a lower bound can still be inferred by collecting leaf costs, the number of successive resolutions is not bounded anymore by k and the only bound on the total number of nodes is always in $O(d^h)$. When recording is used, this approach is therefore dominated in theory (either in time or space) by previous approaches. Since experimental results (not reported) did not exhibit interesting performances, this double-cut approach is only considered without recording (Pseudo-tree search).

Below, we present the pseudo-code of the LC-BTD⁺ algorithm combining tree decomposition and a given level of local consistency (Lc). This algorithm uses our initial enhanced upper bound (line 1), value removal based on local cuts and optional recording (lines 2 and 3). The initial call is LC-BTD⁺(\emptyset, C_1, V_1, k). It assumes an already Lc-consistent problem and returns its optimum. Function pop(S) returns an element of S and remove it from S . LC-BTD⁺ has

```

Function LC-BTD+( $A, C_e, V, cub$ ) : integer
  if ( $V = \emptyset$ ) then
     $S := \text{Sons}(C_e)$ ;
    /* Solve all cluster sons whose optima are unknown */;
    while ( $S \neq \emptyset$  and  $lb(P_e/A) < cub$ ) do
       $C_f := \text{pop}(S)$  /* Choose a cluster son */;
      if ( $\text{not}(\text{Opt}_{P_f/A_f})$ ) then
1        $cub' := cub - lb(P_e/A) + lb(P_f/A_f)$ ;
2        $clb' := \text{LC-BTD}^+(A, C_f, V_f, cub')$ ;
3        $LB_{P_f/A_f} := clb' + \overline{\Delta W}_{P_f/A_f}$ ;
          $\text{Opt}_{P_f/A_f} := (clb' < cub')$ ;
      return  $lb(P_e/A)$ ;
  else
     $i := \text{pop}(V)$  /* Choose an unassigned variable in  $C_e$  */;
     $d := D_i$ ;
    /* Enumerate every value in the domain of  $i$  */;
    while ( $d \neq \emptyset$  and  $lb(P_e/A) < cub$ ) do
       $a := \text{pop}(d)$  /* Choose a value */;
      Assign  $a$  to  $i$  and enforce Lc on subproblem  $P_e/A_e$ ;
      if ( $lb(P_e/A \cup \{(i, a)\}) < cub$ ) then
         $cub := \min(cub, \text{LC-BTD}^+(A \cup \{(i, a)\}, C_e, V, cub))$ ;
    return  $cub$ ;

```

the usual worst case space complexity of tree decomposition based algorithms, exponential in the size of the largest separator S_e . Because it uses the enhanced local upper bound presented before, the number of nodes explored is bounded

by $O(k \cdot d^{w+1})$. Because of branch and bound pruning, it is possible that only a small fraction of the theoretical space will be used. Our implementation uses therefore sparse data structures (hash tables) for recording information.

Experimental results

In this section, we perform an empirical comparison of the algorithm LC-BTD⁺ with classical BTD exploiting Forward Checking (FC-BTD) (Jegou & Terrioux 2004), traditional depth-first branch and bound (MFDAC) (Larrosa & Schiex 2003), and Variable Elimination (VE) (Dechter 1999), a dynamic programming algorithm with time and space complexity in $O(d^{w+1})$, on random and real-world binary instances.

Several versions of our algorithm can be considered: without recording (lines 2 and 3 removed), we get a variant of pseudo-tree search enhanced with local consistency enforcing denoted FDAC-PTS. This algorithm is similar to the AOMFDAC algorithm of (Marinescu & Dechter 2005b). In this case, local and global cuts are used for value removal. For local consistency Lc, we tested Node Consistency (NC-BTD+) and the more powerful FDAC (FDAC-BTD+).

All implementations are in C code (open source solver TOOLBAR <http://carlit.toulouse.inra.fr/cgi-bin/awki.cgi/SoftCSP>). Experiments were performed on a 2.4 GHz Xeon with 4 GB.

Randomly generated instances Tree decomposition approaches are useless for pure unstructured random problems. We designed a structured Max-CSP (violation has unit cost) random problem generator using a binary clique tree. The model parameters are (w, s, h', d, t) where $w+1$ is the clique size, s the separator size, h' the height of the clique tree, d the domain size, and t the constraint tightness (percentage of forbidden tuples). The total number of variables is equal to $n = s + (w + 1 - s)(2^{h'} - 1)$. Our tree decomposition method produces a tree-width equal to w and a tree-height equal to $h = h'(w + 1 - s) + s$. We have generated and solved $(w = 9, h' = 3, d = 5)$ clique trees with varying separator sizes $s \in \{2, 5, 7\}$ and varying constraint tightness $t \in [30, 90]\%$ (over-constrained instances only). Average over 50 instances are reported. All algorithms (except VE) use the *dom/deg* dynamic variable ordering heuristic (locally inside clusters for tree decomposition based methods). For value selection we consider values in increasing order of unary cost W_i . The DAC variable ordering is lexicographic. The tree decomposition and the variable elimination order are computed using the maximum cardinality search (MCS) heuristic (Tarjan & Yannakakis 1984) with linear time complexity in $O(n + |E|)$. A root minimizing the tree-height is used. The cpu-time reported excludes the time to compute a tree-decomposition (always less than 10 ms) and is limited to 5 minutes per instance (unsolved instances are assigned a 5 minute solving time). Fig. 2 reports times in seconds and memory usage in number of recorded lower bounds.

For weak local consistencies such as FC and NC (which provide the same lower bound), our enhanced initial upper bound saves space at the cost of time on these prob-

lems. With a stronger local consistency, FDAC-BTD+ is always among the best algorithms and may outperform FC-BTD by two orders of magnitude for large separator size and high constraint tightness. For small/medium separators, recording seems to be a crucial ingredient for efficiency and FDAC-PTS is even dominated by NC/FC recording approaches in this case. MFDAC was unable to solve the instances with small separator size and large number of variables. Finally, variable elimination took constant time and space (bounded by $d^{w+1} = 5^{10} = 9.8 \cdot 10^6$). Over all instances, the number of successive resolutions of any sub-problem P_e/A_e never exceeded 11 for FDAC-BTD+ and 17 for NC-BTD+, far from the theoretical bound defined by the largest optimal cost, here equal to 202.

Real-world instances The Radio Link Frequency Assignment Problem (RLFAP) (Cabon *et al.* 1999) consists in assigning frequencies to a set of radio links in such a way that all the links may operate together without noticeable interference. Some RLFAP instances can be naturally cast as weighted binary CSPs. We focused on instance SCEN-06 and its sub-instances SUB₁ and SUB₄. Compared to the previous settings, we used the best known solution cost (optimal) as the initial global upper bound, preprocessed the MCS tree decomposition by merging clusters with a separator size strictly greater than 3 (MCS time always less than 10 ms) and used the *2-sided Jeroslow-like* heuristic for variable selection. The root cluster chosen maximizes the product of domain sizes. By doing so, the root clusters of SCEN-06 and SUB₄ are the same (20 variables). Time limit is 10 hours. The same statistics on time and number of recorded lower bounds (#LB) are reported.

RLFAP optimum n, d, w, h	SUB ₁ 2669 14, 44, 13, 14		SUB ₄ 3230 22, 44, 19, 21		SCEN-06 3389 100, 44, 19, 67	
Method	time	#LB	time	#LB	time	#LB
FC-BTD	1197	0	-	0	-	-
NC-BTD+	490	0	-	0	-	-
FDAC-BTD+	14	0	929	0	10,309	326
FDAC-PTS	14	<i>n/a</i>	851	<i>n/a</i>	-	<i>n/a</i>
MFDAC	14	<i>n/a</i>	984	<i>n/a</i>	-	<i>n/a</i>

Again, FDAC-BTD+ shows its robustness. Slightly dominated by FDAC-PTS on the SUB₄ problem, it is the only algorithm that is able to prove optimality of the full SCEN-06 instance. The recording, even if quite limited (326 over a possible maximum of roughly one million(M), used 5.5M times during the exploration of 16M search nodes), is the only difference (except the value removal policy) with FDAC-PTS algorithm that didn't solve the problem in the 10 hours limit ("-" symbol) but finished in 2 days. Variable elimination didn't solve any instance due to the 4 GB limit.

Compared with related work, in (Jegou & Terrioux 2004), FC-BTD without initial global upper bound solved SUB₄ in 122,933 seconds. AOMFDAC using a static variable ordering solved the same sub-instance in 47,115 seconds (Marinescu & Dechter 2005b). Both experiments were performed on a 2.4 GHz Pentium IV computer but used different tree decomposition from us. First solved by a partitioning ap-

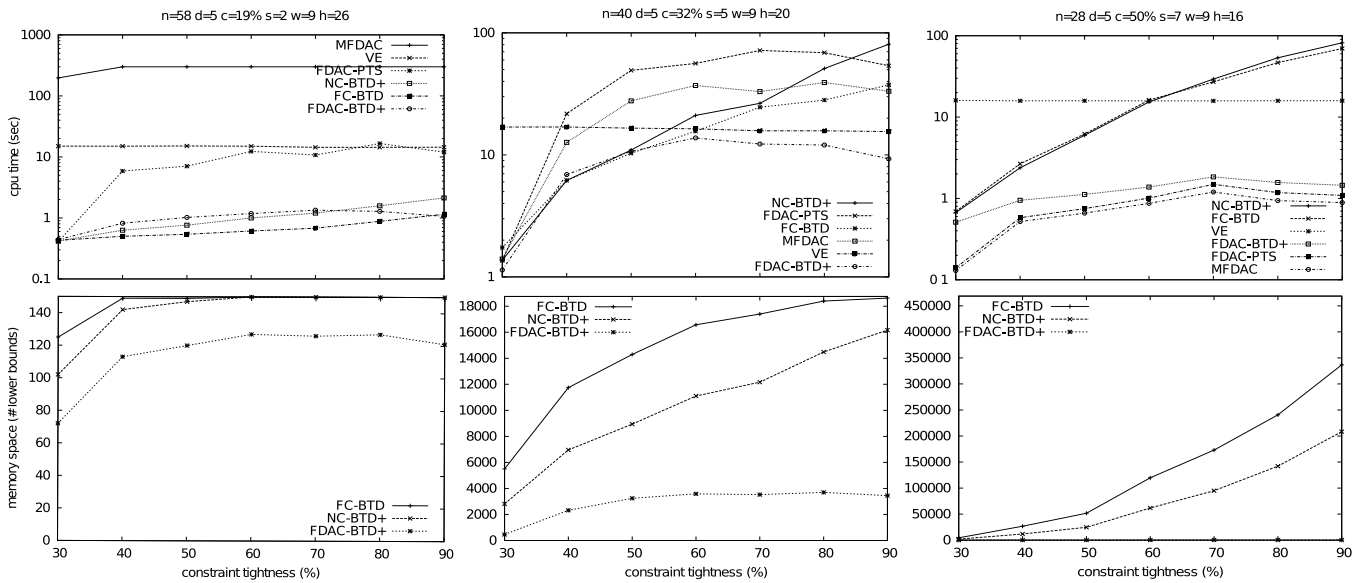


Figure 2: Time in seconds (y-axis in log-scale) and number of recorded lower bounds (y-axis in linear-scale, its maximum corresponds to the theoretical maximum) to find the optimum and prove optimality of random binary clique trees with various separator sizes s equal to 2, 5, and 7 (from left to right). Algorithms are sorted from the worst (top) to the best (bottom) at $t = 90\%$. n , d , c , w and h are the variable number, domain size, graph connectivity, tree-width, and tree-height respectively.

proach (SUB₄ being the largest component), the full SCEN-06 instance was also solved by a specific dynamic programming algorithm in 27,102 seconds on a DEC 2100 A500MP workstation (Koster 1999). However, our approach is more generic, we do not use any specific preprocessing dedicated to frequency assignment problems, and uses less memory (326 compared to roughly one million (Koster 1999)).

Conclusion

In this paper, we have proposed a branch and bound algorithm exploiting tree decomposition and local consistency. Among the possible tree decomposition based approaches, we found that the most promising and robust approach combines initial enhanced upper bounds, strong local consistency, *corrected* recorded lower bounds, and value removal based on local cuts. The resulting algorithm sacrifices a bit of theory for improved practical (time and space) overall efficiency as shown on random and real-world instances.

We have presented our algorithm in the framework of weighted binary CSP. Our approach is also applicable to non-binary soft constraints, opening the door to other domains such as Max-SAT and MPE in Bayesian Networks.

References

- Bacchus, F.; Dalmao, S.; and Pitassi, T. 2003. Value elimination: Bayesian inference via backtracking search. In *UAI-03*, 20–28.
- Bodlaender, H. 2005. Discovering treewidth. In *Theory and Practice of Computer Science - SOFSEM'2005*, 1–16.
- Cabon, B.; de Givry, S.; Lobjois, L.; Schiex, T.; and Warners, J. 1999. Radio link frequency assignment. *Constraints* 4:79–89.
- Cooper, M., and Schiex, T. 2004. Arc consistency for soft constraints. *Artificial Intelligence* 154:199–227.

- Darwiche, A. 2001. Recursive Conditioning. *Artificial Intelligence* 126(1-2):5–41.
- Dechter, R. 1999. Bucket elimination: A unifying framework for reasoning. *Artificial Intelligence* 113(1-2):41–85.
- Freuder, E. C., and Quinn, M. J. 1985. Taking advantage of stable sets in constraint satisfaction problems. In *Proc. of IJCAI-85*, 1076–1078.
- Jegou, P., and Terrioux, C. 2004. Decomposition and good recording. In *Proc. of ECAI-04*, 196–200.
- Koster, A. 1999. *Frequency assignment: Models and Algorithms*. Ph.D. Dissertation, Maastricht, The Netherlands.
- Larrosa, J., and Schiex, T. 2003. In the quest of the best form of local consistency for weighted CSP. In *Proc. of IJCAI-03*, 239–244.
- Larrosa, J., and Schiex, T. 2004. Solving Weighted CSP by Maintaining Arc-consistency. *Artificial Intelligence* 159(1-2):1–26.
- Larrosa, J.; Meseguer, P.; and Sanchez, M. 2002. Pseudo-tree search with soft constraints. In *Proc. of ECAI-02*, 131–135.
- Marinescu, R., and Dechter, R. 2005a. Advances in and/or branch-and-bound search for constraint optimization. In *CP-2005 workshop on Preferences and Soft Constraints*.
- Marinescu, R., and Dechter, R. 2005b. And/or branch-and-bound for graphical models. In *Proc. of IJCAI-05*, 224–229.
- Schiex, T.; Fargier, H.; and Verfaillie, G. 1995. Valued constraint satisfaction problems: hard and easy problems. In *Proc. of IJCAI-95*, 631–637.
- Tarjan, R., and Yannakakis, M. 1984. Simple linear-time algorithms to test chordality of graphs, test acyclicity of hypergraphs and selectively reduce acyclic hypergraphs. *SIAM J. Comput.* 13(3):566–579.
- Terrioux, C., and Jegou, P. 2003. Hybrid backtracking bounded by tree-decomposition of constraint networks. *Artificial Intelligence* 146(1):43–75.