Russian doll search with tree decomposition

Marti Sanchez, David Allouche, Simon de Givry, Thomas Schiex

Russian Doll Search with Tree Decomposition

M. Sanchez, D. Allouche, S. de Givry, T. Schiex
UBIA, UR 875, INRA, F-31320 Castanet Tolosan, France
{msanchez,allouche,degivry,tschiex}@toulouse.inra.fr

Abstract

Optimization in graphical models is an important problem which has been studied in many AI frameworks such as weighted CSP, maximum satisfiability or probabilistic networks. By identifying conditionally independent subproblems, which are solved independently and whose optimum is cached, recent Branch and Bound algorithms offer better asymptotic time complexity. But the locality of bounds induced by decomposition often hampers the practical effects of this result because subproblems are often uselessly solved to optimality.

Following the Russian Doll Search (RDS) algorithm, a possible approach to overcome this weakness is to (inductively) solve a relaxation of each subproblem to strengthen bounds. The algorithm obtained generalizes both RDS and tree-decomposition based algorithms such as BTD or AND-OR Branch and Bound. We study its efficiency on different problems, closing a very hard frequency assignment instance which has been open for more than 10 years.

1 Introduction

Graphical model processing is a central problem in AI. The optimization of the combined cost of local cost functions, central in the valued CSP framework [Schiex *et al.*, 1995], captures problems such as weighted MaxSAT, Weighted CSP, Maximum Probability Explanation (MPE) in Bayesian networks or Markov random fields. It has applications in *e.g. resource allocation* [Verfaillie *et al.*, 1996; Cabon *et al.*, 1999] and *bioinformatics* [Sanchez *et al.*, 2008].

In the last years, in order to solve satisfaction, optimization or counting problems, several tree-search 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 Recursive Conditioning [Darwiche, 2001], Backtrack bounded by Tree Decomposition (BTD) [Terrioux and Jegou, 2003], AND-OR graph search [Marinescu and Dechter, 2006], all related to Pseudo-Tree Search [Freuder and Quinn, 1985]. Compared to traditional tree-search, they offer improved asymptotic time complexities, only exponential in the treewidth of the graph. This comes however at the price of restricted variable ordering (see [Jégou *et al.*, 2007]).

In the context of optimization (w.l.o.g., we consider minimization problems), an additional side-effect of problem decomposition is the loss of global bounds. In traditional Depth-First Branch and Bound (DFBB), pruning occurs when the cost of the best known solution of the problem (the upper bound) meets a specific lower bound computed on a relaxation of the whole problem. Using problem decomposition, when the assignment of the variables that connect a subproblem to the rest of the problem makes it independent, an optimal solution of the subproblem (conditionally to this assignment) is sought. This is an expensive and often useless work since this assignment is not necessarily part of the optimal global solution (because of the rest of the problem). To limit this thrashing behavior, one can inform the subproblem solver that it must absolutely produce a sufficiently good solution or stop if it proves this is unfeasible. This *initial upper bound* is obtained by subtracting from the global upper bound any available lower bound for the rest of the problem.

In this paper, we start from the approach of [de Givry *et al.*, 2006] which introduced lower bounds derived by *soft local consistency* [Cooper and Schiex, 2004] in the BTD algorithm. To improve the contribution of the global problem to the local upper bound which is used for solving subproblems, we replace the DFBB engine of BTD by the Russian Doll Search (RDS) [Verfaillie *et al.*, 1996] algorithm. RDS is extended to use a tree-decomposition and stronger local consistency. Inside RDS-BTD, relaxations of the conditionally independent identified subproblems are solved inductively. The optimal costs of these subproblems provide powerful lower bounds, and therefore improved local upper bounds which strongly reduce the trashing behavior.

In the rest of the paper, we first present the weighted CSP framework, the BTD and the RDS algorithms. We then introduce our new generic algorithm, which can also be specialized into either BTD or RDS. The RDS-BTD algorithm is then evaluated on different real problem instances of radio link frequency assignment (resource allocation in telecommunications) and tag SNP selection (genetics).

2 Weighted Constraint Satisfaction Problem

A Weighted CSP (WCSP) is a quadruplet (X, D, W, m). X and D are sets of n variables and finite domains, as in a stan-

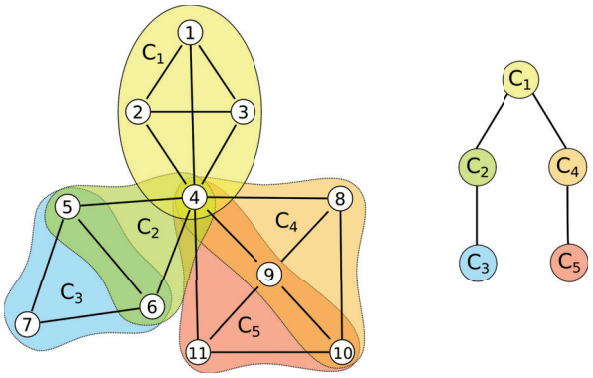


Figure 1: The graph of a WCSP and its tree decomposition.

dard CSP. The domain of variable i is denoted D_i. The maximum domain size is d. For a set of variables $S \subset X$, we note $\ell(S)$ the set of tuples over S. W is a set of cost functions. Each cost function w_S in W is defined on a set of variables S called its scope, assumed to be different for each cost function. A cost function w_S assigns costs to assignments of the variables in S i.e. $w_S : \ell(S) \rightarrow [0, m]$. The set of possible costs is $[0, m]$ and $m \in \{1, \ldots, +\infty\}$ represents an intolerable cost. Costs are combined by the bounded addition $\oplus$, such as $a \oplus b = \min\{m, a + b\}$ and compared using $\geq$.

For unary/binary cost functions, we use w_i (resp. w_{ij}) to denote a cost function involving i (resp. i and j). For every variable i, we add a unary cost function w_i if it does not exist. We also add a nullary cost function, noted $w_\varnothing$ (a constant cost payed by any assignment). All these additional cost functions have initial cost 0, leaving an equivalent problem.

The cost of a complete assignment $t \in \ell(X)$ in a problem $P = (X, D, W, m)$ is $Val_P(t) = \bigoplus_{w_S \in W} w_S(t[S])$ where $t[S]$ denotes the projection of a tuple on the set of variables S. The problem of minimizing $Val_P(t)$ is NP-hard.

Enforcing a local consistency property on a problem P consists in transforming P in an equivalent problem P' ($Val_P = Val_{P'}$) satisfying the considered property. This enforcing is achieved by moving costs between different scopes, possibly increasing $w_\varnothing$, giving a better lower bound on the optimal cost [Cooper and Schiex, 2004; de Givry *et al.*, 2005].

3 Tree decompositions and DFBB

A tree decomposition of a connected WCSP is defined by a tree (C, T). The set of nodes of the tree is $C = \{C_1, \ldots, C_k\}$ where each 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 cost functions ($\forall w_S \in W, \exists C_e \in C$ *s.t.* $S \subset C_e$). Furthermore, if a variable i appears in two clusters C_e and C_g, i must also appear in the clusters C_f on the unique path from C_e to C_g in T.

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 $V_e = C_e \setminus S_e$.

Example 1 *Consider the MaxCSP in Figure 1. It has eleven variables with two values (a, b) in their domains. Edges represent binary difference constraints ($w_{ij}(a, a) = w_{ij}(b, b) = 1, w_{ij}(a, b) = w_{ij}(b, a) = 0$) between the two variables. An optimal solution is $(a, b, b, a, b, b, a, b, b, a, b)$ in lexicographic order, with optimal cost 5. A C_1-rooted tree decomposition with clusters $C_1 = \{1, 2, 3, 4\}, C_2 = \{4, 5, 6\}, C_3 = \{5, 6, 7\}, C_4 = \{4, 8, 9, 10\}$, and $C_5 = \{4, 9, 10, 11\}$, is given on the right hand-side in Figure 1. C_1 has sons $\{C_2, C_4\}$, the separator of C_3 with its father C_2 is $S_3 = \{5, 6\}$, and the set of proper variables of C_3 is $V_3 = \{7\}$. The subproblem P_3 has variables $\{5, 6, 7\}$ and cost functions $\{w_{5,7}, w_{6,7}, w_7\}$. P_1 is the whole problem.*

The essential property of tree decompositions 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 cost functions, together with the variables they involve, define the remaining subproblem.

A tree search algorithm can exploit this property as far as a suitable variable ordering is used: the variables of any cluster C_e must be assigned before the variables which remain in its son clusters. In this case, for any cluster $C_f \in$ *Sons*(C_e), once the separator S_f is assigned, the subproblem P_f conditioned by the current assignment A_f of S_f (denoted P_f/A_f) can be solved to optimality independently of the rest of the problem. The *optimal cost* of P_f/A_f may then be recorded which means it will never be solved again for the same assignment of S_f. This is how algorithms such as BTD or AND-OR graph search are able to keep the complexity exponential in the size of the largest cluster only.

In [de Givry *et al.*, 2006], the BTD algorithm is further sophisticated by maintaining lower bounds provided by soft local consistency. For every cluster C_e, a cluster specific constant cost function $w^e_\varnothing$ is used and possibly increased by soft local consistency, providing a lower bound on the optimal cost of each cluster. Beyond improved pruning, this allows to avoid to always solve subproblems to optimality by computing a required maximum cost for every subproblem P_f. For P_1, the cost of the best known solution defines the initial upper bound cub_1 (we want to improve over this cost). Consider now an arbitrary cluster C_f, son of C_e with an associated bound cub_e. Then the upper bound for the subproblem P_f is obtained by *subtracting* from cub_e the lower bounds associated to all the other sons clusters of C_e. Thus, stronger lower bounds on other subproblems will induce a stronger upper bound for the current problem.

Because of this initial upper bound, the optimum of P_f is not necessarily computed but only a lower bound on it. This lower bound and its optimality are recorded in LB_{P_f/A_f} and Opt_{P_f/A_f} respectively, initially set to 0 and *false*.

For improved pruning and upper bounding, BTD uses the maximum between soft local consistency ($w^e_\varnothing$) and recorded (LB_{P_f/A_f}) lower bounds. Recorded lower bounds can only be used when the separator S_f is completely assigned by the current assignment A. This inductively defines the lower

bound used in [de Givry *et al.*, 2006] as $lb(P_e/A) = w^e_\varnothing \oplus \bigoplus_{C_f \in Sons(C_e)} \max(lb(P_f/A), LB_{P_f/A_f})$.

Example 2 *In Example 1, variables* $\{1,2,3,4\}$ *of* C_1 *are assigned first, using e.g. the* min domain/max degree *dynamic variable ordering. Let* $A = \{(4,a),(1,a),(2,b),(3,b)\}$ *be the current assignment. Enforcing EDAC local consistency [de Givry* et al.*, 2005] on* P_1/A *produces* $w^1_\varnothing = 2, w^2_\varnothing = w^4_\varnothing = 1, w^3_\varnothing = w^5_\varnothing = 0$*, resulting in* $lb(P_1/A) = \bigoplus_{C_e \in C} w^e_\varnothing = 4$ *since no recorded bound is available yet.*

Then, subproblems $P_2/\{(4,a)\}$ *and* $P_4/\{(4,a)\}$ *are solved independently and the corresponding optimal solutions are recorded as* $LB_{P_2/\{(4,a)\}} = 1$*,* $LB_{P_4/\{(4,a)\}} = 2$*,* $Opt_{P_2/\{(4,a)\}} = Opt_{P_4/\{(4,a)\}} =$ *true (no local upper bound can be derived since there is no known solution). A first complete assignment of cost* $w^1_\varnothing \oplus LB_{P_2/\{(4,a)\}} \oplus LB_{P_4/\{(4,a)\}} = 5$ *is found.*

The resulting BTD algorithm has worst-case time complexity exponential in the maximum cluster size minus one, called the treewidth w of the tree decomposition (C,T). Its space complexity is exponential in s with $s = \max_{C_e \in C} |S_e|$, the maximum separator size [de Givry *et al.*, 2006].

4 Russian Doll Search and tree decomposition

The original Russian Doll Search (RDS) algorithm [Verfaillie *et al.*, 1996] was proposed in 1996, when soft local consistency for WCSP had not yet been introduced. Its underlying mechanism was designed to build powerful lower bounds from weak ones using tree search. RDS consists in solving n nested subproblems of an initial problem P with n variables. Given a fixed variable order, it starts by solving the subproblem with only the last variable. Next, it adds the preceding variable in the order and solves this subproblem with two variables, and repeats this process until the complete problem is solved. Each subproblem is solved by a DFBB algorithm with a static variable ordering following the nested subproblem decomposition order. The improved lower bound is derived by combining the weak lower bound produced by partial forward checking (similar in power to Node Consistency [Larrosa and Schiex, 2004]) with the optimum of the problem previously solved by RDS.

We propose to exploit the RDS principle using a tree decomposition (RDS-BTD). The main difference with RDS is that the set of subproblems solved is defined by a rooted tree decomposition (C,T): RDS-BTD solves $|C|$ subproblems ordered by a depth-first traversal of T, starting from the leaves to the root $P^{\text{RDS}}_1 = P_1$.

We define P^{RDS}_e as the subproblem defined by the *proper variables* of C_e and all its descendant clusters in T and by all the cost functions involving *only* proper variables of these clusters. P^{RDS}_e has no cost function involving a variable in S_e, the separator with its father, and thus its optimum is a lower bound of P_e conditioned by *any* assignment of S_e. This optimum will therefore be denoted as $LB^{\text{RDS}}_{P_e}$

Each subproblem P^{RDS}_e is solved by BTD instead of DFBB. This allows to exploit the decomposition and the caching done in BTD, offering improved asymptotic time complexity. The lower bound used by BTD can now exploit the $LB^{\text{RDS}}_{P_e}$ lower bounds already computed on previous clusters together with soft local consistency and recorded lower bounds[1]. The lower bound corresponding to the current assignment A is now inductively defined as $lb(P_e/A) = w^e_\varnothing \oplus \bigoplus_{C_f \in Sons(C_e)} \max(lb(P_f/A), LB_{P_f/A_f}, LB^{\text{RDS}}_{P_f})$, which is obviously stronger.

In BTD, caching is only performed on completely assigned separators and P^{RDS}_e does not contain the separator variables S_e. We therefore assign S_e before solving P^{RDS}_e using the fully supported value maintained by EDAC [de Givry *et al.*, 2005][2] of each variable as temporary values used for caching purposes only. An alternative approach would be to cache lower bounds for partial assignments but this would require a complex cache management that we leave for future work.

The advantage of using BTD is that recorded lower bounds can also be reused between each iterations of RDS-BTD. However, an optimum cached by BTD for a given subproblem P_f when solving P^{RDS}_e is no longer necessarily optimum in $P^{\text{RDS}}_{Father(e)}$ if a cost function between P_f and variables in $C_{Father(e)}$ exists. Therefore, at each iteration of RDS-BTD, after P^{RDS}_e is solved, we reset all Opt_{P_f/A_f} such that $S_f \cap S_e \neq \varnothing$ (line 4).

Example 3 *Consider Example 1, RDS-BTD successively solves the subproblems* P^{RDS}_3*,* P^{RDS}_2*,* P^{RDS}_5*,* P^{RDS}_4 *and* $P^{\text{RDS}}_1 = P_1$*.* P^{RDS}_3 *contains just* $\{7\}$ *as variables and* $\{w_7\}$ *as cost functions. Before solving* P^{RDS}_3*, RDS-BTD assigns variables* $\{5,6\}$ *of the* S_3 *separator to their fully supported value (*$\{(5,a),(6,a)\}$ *in this example). When* P^{RDS}_2 *is solved, the optimum of* $P_3/\{(5,a),(6,a)\}$*, which is equal to zero since* $w_{5,6}$ *does not belong to* P_3 *is recorded and can be reused when solving* P_1*. When* P^{RDS}_4 *is solved, the optimum of* $P_5/\{(4,a),(9,a),(10,a)\}$*, also equal to zero is recorded. In this case, since variable* 4 *belongs to* $S_5 \cap S_4$ *and* P^{RDS}_4 *does not contain* $w_{4,11}$*, this recorded optimum is just a lower bound for subsequent iterations of RDS-BTD. So, we set* $Opt_{P_5/\{(4,a),(9,a),(10,a)\}}$ *to false before solving* P_1*.*

The optima found are $LB^{\text{RDS}}_{P_3} = LB^{\text{RDS}}_{P_5} = 0$*,* $LB^{\text{RDS}}_{P_2} = LB^{\text{RDS}}_{P_4} = 1$ *and* $LB^{\text{RDS}}_{P_1} = 5$*, the optimum of* P_1*.*

In this simple example, for an initial empty assignment $A = \varnothing$*, we have* $lb(P_1/\varnothing) = LB_{P^{\text{RDS}}_2} \oplus LB_{P^{\text{RDS}}_4} = 2$ *instead of* 0 *for BTD (assuming EDAC local consistency in preprocessing and no initial upper bound).*

The pseudo-code of the RDS-BTD algorithm is presented in Algorithm 1. The BTD algorithm used is the same as in [de

[1] Actually, because soft local consistency may move cost between clusters, the $LB^{\text{RDS}}_{P_f}$ lower bound must be adjusted. This can be achieved using the ΔW data-structure of [de Givry *et al.*, 2006] by subtracting $\bigoplus_{i \in S_f} \max_{a \in D_i} \Delta W^f_i(a))$ from it.

[2] Fully supported value $a \in D_i$ such that $w_i(a) = 0$ and $\forall w_S \in W$ *with* $i \in S, \exists t \in \ell(S)$ *with* $t[i] = a$ such that $w_S(t) = 0$.

Givry *et al.*, 2006] except for the fact that it uses the RDS enriched lower bound above. We assume that solving P_e^{RDS} with an initial assignment A of the separator S_e and an initial upper bound cub_e using BTD is achieved by a call to $\mathsf{BTD}(P_e^{\text{RDS}}, A, V_e, cub_e)$.

RDS-BTD calls BTD to solve each subproblem P_e^{RDS} (line 3). An initial upper bound for P_e^{RDS} is deduced from the global upper bound and the available RDS lower bounds (line 1). It initially assigns variables in S_e to their fully supported value at line 2 as discussed above. The initial call is $\mathsf{RDS\text{-}BTD}(P)$. It assumes an already local consistent problem $P_1^{\text{RDS}} = P$ and returns its optimum.

Algorithm 1: RDS-BTD algorithm

Function $\mathsf{RDS\text{-}BTD}(P_e^{\text{RDS}}) : [0, +\infty]$
  foreach $C_f \in Sons(C_e)$ **do** $\mathsf{RDS\text{-}BTD}(P_f^{\text{RDS}})$;
1 $cub_e := cub_1 - lb(P/\varnothing) + lb(P_e^{\text{RDS}}/\varnothing)$;
2 Let A be the assignment of S_e to fully supported values;
3 $LB_{P_e}^{\text{RDS}} := \mathsf{BTD}(P_e^{\text{RDS}}, A, V_e, cub_e)$;
  foreach C_f descendant of C_e *s.t.* $S_f \cap S_e \neq \varnothing$ **do**
4 Set to *false* all recorded $Opt_{P_f/A}$, for all $A \in \ell(S_f)$;
  return $LB_{P_e}^{\text{RDS}}$;

Notice that as soon as a solution of P_e^{RDS} is found having the same optimal cost as $lb(P_e^{\text{RDS}}/\varnothing) = \bigoplus_{C_f \in Sons(C_e)} LB_{P_f^{\text{RDS}}}$, then the search ends.

The time and space complexity of RDS-BTD is the same as BTD. Notice that RDS-BTD without caching, using only node consistency and a pseudo-tree based tree decomposition (i.e. a cluster for each variable, implying a static variable ordering) is equivalent to Pseudo-Tree RDS [Larrosa *et al.*, 2002]. If we further restrict the algorithm to use a specific tree decomposition (C, T) such that $|C| = n, \forall e \in [1, n], C_e = \{1, \ldots, e\}$, and $\forall e \in [2, n], Father(C_e) = C_{e-1}$, then it becomes equivalent to RDS.

5 Experimental results

We implemented DFBB, BTD, RDS-BTD, and RDS in `toulbar2` C++ solver[3]. The *min domain / max degree* dynamic variable ordering, breaking ties with maximum unary cost, is used inside clusters (BTD and RDS-BTD) and by DFBB. The dynamic variable ordering heuristic is modified by a conflict back-jumping heuristic as suggested in [Lecoutre *et al.*, 2006]. EDAC local consistency is enforced [de Givry *et al.*, 2005] during search. RDS enforces Node Consistency [Larrosa and Schiex, 2004] only. Tree decompositions are built using the Maximum Cardinality Search (MCS) heuristic, with the largest cluster used as root (except for `scen07`). From the MCS derived tree decomposition, we computed alternative tree decompositions based on a maximum separator size s_{max} as proposed in [Jégou *et al.*, 2007]: starting from the leaves of the MCS tree decomposition, we merge a cluster with its parent if the separator size is strictly greater than s_{max}. A variable ordering compatible with the rooted tree decomposition used is used for DAC enforcing [Cooper and Schiex, 2004] and RDS.

Recorded (and if available RDS) lower bounds are exploited by soft local consistency enforcing as soon as their separator variables are fully assigned. If the recorded lower bound is optimal or strictly greater than the lower bound produced by EDAC, then the corresponding subproblem (P_e/A_e) is disconnected from soft local consistency enforcing and the positive difference in lower bounds is added to its parent cluster lower bound ($w_{\varnothing}^{Father(C_e)}$), allowing for new value removals by node consistency enforcing.

All the solving methods exploit a binary branching scheme. If $d > 10$, the *ordered* domain is split in two parts (around the middle value), else the variable is assigned to its EDAC fully supported value or this value is removed from the domain. In both cases, it selects the branch which contains the fully supported value first, except for RDS and BTD-like methods where it selects first the branch which contains the value in the last solution(s) found if available. Unless mentioned, no initial upper bound is provided. Reported CPU times correspond to finding the optimum and proving its optimality.

5.1 Radio Link Frequency Assignment

Among the different CELAR instances [Cabon *et al.*, 1999] which can be described as binary weighted CSP, we selected the two difficult instances `scen06` and `scen07`. The `scen07` instance was reduced by several preprocessing rules (domain pruning by singleton EDAC consistency and Koster's dominance rules [de Givry, 2004], and variable elimination of small degree) leading to an instance with $n = 162$, $d = 44$ and $e = 764$. The tree decomposition used $s_{max} = 3$, with a treewidth of 53. An initial upper bound (354008) was provided to all methods. This bound and the solution were found by DFBB enhanced by Limited Discrepancy Search. This solution is also used by RDS-BTD as an initial value choice (we found DFBB and BTD performed worse with it). The root cluster was selected based on the first fail principle (most costly cluster). BTD and RDS-BTD found the optimum of 343592 and proved optimality in respectively 6 and 4.5 days (26% improvement for RDS-BTD) on a 2.6 GHz computer with 32GB. BTD learnt 90528 recorded lower bounds (20% of total separator assignments) compared to 29453 (6.4%) by RDS-BTD. DFBB did not finish in 50 days. This is the first time this 10-year-old open problem is solved to optimality.

We also solved `scen06` instance (100 variables with treewidth of 11) without preprocessing rules nor upper bound. BTD and RDS-BTD took 221 and 316 seconds respectively to find the optimum and prove optimality. There were only 5633 and 7145 recorded lower bounds respectively despite the fact that we did not restrict separator sizes ($s_{max} = 8$). DFBB took 2588 seconds and RDS did not finish in 10 hours.

5.2 The Tag SNP selection

This problem occurs in genetics and polymorphism analysis. Single nucleotide polymorphisms, or SNPs, are DNA sequence variations that occur when a single nucleotide

[3] http://mulcyber.toulouse.inra.fr/projects/toulbar2 version 0.8.

(A,T,C,or G) in the genome sequence of an individual is altered. For example a SNP might change the DNA sequence AAGGCTAA to A**T**GGCTAA. For a variation to be considered a SNP, it must occur in at least 1% of the population. There are several millions SNPs in the 3 billions nucleotides long human genome, explaining up to 90% of all human genetic variation. SNPs may explain a portion of the heritable risk of common diseases and can affect respond to pathogens, chemicals, drugs, vaccines, and other agents. The TagSNP problem is a sort of lossy compression problem which consists in selecting a small subset of SNPs such that the selected SNPs, called tag SNPs, will capture most of the genetic information. The goal is to capture a maximally informative subset of SNPs to make screening of large populations feasible [Hirschhorn and Daly, 2005].

The correlation measure r^2 between a pair of SNPs can be computed on a first small population. A tag SNP is considered as representative of another SNP if the two SNPs are sufficiently linked. The simplest TagSNP problem is to select a minimum number of SNPs (primary criteria) such that all SNPs are represented. This is captured by the fact that the r^2 measure between the two SNPs is larger than a threshold θ (often set to $\theta = 0.8$ [Carlson *et al.*, 2004]). We therefore consider a graph where each vertex is a SNP and where edges are labelled by the r^2 measure between pairs of nodes. Edges are filtered if their label is lower than the threshold θ. The graph obtained may have different connected components. The TagSNP problem then reduces to a set covering problem (NP-hard) on these components. This simplest variant has been studied in [A. Choi *et al.*, 2008] where it is solved using the d-DNNF compiler c2d with good results.

In practice the number of optimal solutions may still be extremely large and secondary criteria are considered by state-of-the-art tools such as FESTA [Qin *et al.*, 2006] (relying on two incomplete algorithms). Between tag SNPs, a low r^2 is preferred, to maximize tag SNP dispersion. Between a non tag SNP and its representative tag SNP, a high r^2 is preferred to maximize the representativity.

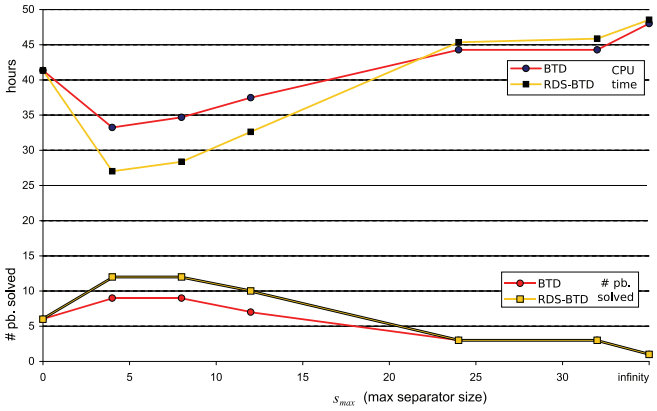
For a given connected graph $G = (V, E)$, we build a binary WCSP with integer costs capturing the TagSNP problem with the above secondary criteria. For each SNP i, two variables i_s and i_r are used. i_s is a boolean variable that indicates if the SNP is selected as a tag SNP or not. The domain of i_r is the set of neighbors of i together with i itself. It indicates the representative tag SNP which covers i. For a SNP i, hard binary cost functions (with 0 or infinite costs) enforces the fact that $i_s \Rightarrow (i_r = i)$. Similar hard cost functions enforce $(i_r = j) \Rightarrow j_s$ with neighbor SNPs j in G. A unary cost function on every variable i_s generates an elementary cost U if the variable is *true*. The resulting weighted CSP captures the set covering problem defined by TagSNP.

To account for the representativity, a unary cost function is associated with every variable i_r that generates cost when $i_r \neq i$. In this case, the cost generated is $\lfloor 100.\frac{1-r^2_{i,i_r}}{1-\theta} \rfloor$. For dispersion between SNPs i and j, a binary cost function between the boolean i_s and j_s is created which generates a cost of $\lfloor 100.\frac{r^2_{ij}-\theta}{1-\theta} \rfloor$ when $i_s = j_s =$ *true* The resulting WCSP captures both dispersion and representativity. In order to keep these criteria as secondary, we just use a large enough value for U (the elementary cost used for tag selection).

This problem is similar to a set covering problem with additional binary costs. Such secondary criteria are ignored by [A. Choi *et al.*, 2008]. Here, c2d yields a compact compiled representation of the set of solutions of the pure set covering problem, but the number of solutions is so huge (typically more than billions) that applying the second criteria on solutions generated by c2d would be too expensive. A direct compilation of the criteria in the d-DNNF does not seem straightforward and would probably necessitate a Max-SAT formulation as the authors acknowledge in their conclusion.

The instances considered have been derived from human chromosome 1 data provided by courtesy of Steve Qin [Qin *et al.*, 2006]. Two values, $\theta = 0.8$ and 0.5 have been tried. For $\theta = 0.8$, a usual value in tag SNP selection, $43,251$ connected components are identified among which we selected the 82 largest ones. These problems, with 33 to 464 SNPs, define WCSP with domain sizes ranging from 15 to 224 and are relatively easy. Solving to optimality selects 359 tag SNPs in 2h37' instead of 487 in 3' for FESTA-greedy (21% improvement) or 370 in 39h17' for FESTA-hybrid (3% improvement, 15-fold speedup).

To get more challenging problems, we lowered θ to 0.5. This defined $19,750$ connected components, among which 516 are not solved to optimality by FESTA. We selected the 25 largest one. These problems, with 171 to 777 SNPs have graph densities between 6% and 37%. They define WCSP with max domain size ranging from 30 to 266 and include between 8000 to $250,000$ cost functions. The decomposability of these problems, estimated by the ratio between the treewidth of the MCS tree-decomposition ($s_{max} = +\infty$) and the number of variables varies from 14% to 23%.



All the problems were solved with an initial upper bound found by FESTA-greedy on 2.8 Ghz CPU with 32 GB RAM. To better show the importance of bounded separator size (using s_{max}), we considered values ranging from 0 (DFBB), 4, 8, 12, 24, 32 to $+\infty$ for both BTD and RDS-BTD. We report both the number of problems solved within a 2-hour limit per instance and the total amount of CPU time used (an unsolved instance contributes for 2 hours). RDS exhibited poor performances and is not reported here.

Using $s_{max} = 4$, our implementation improves the compression ratio of FESTA-greedy by 15% (selecting 2952 tag

SNPs instead of 3477 for the $516-13$ solved instances). Note that the differences in CPU time between BTD and RDS-BTD would increase if a larger time limit was used. From a practical viewpoint, the criterion of the TagSNP problem could be further refined to include : sequence annotation information (*e.g.* preferring tag SNPs occurring in genes), and measures between triplets of markers as proposed in [A. Choi *et al.*, 2008] (SNPs covered by a pair of tag SNPs). The good performances of RDS-BTD may allow to tackle this more complex problem with realistic $\theta = 0.8$.

6 Conclusion

The practical exploitation of tree decompositions to solve combinatorial optimization problems that have structure is not always straightforward. Optimization problems explicitly define a global criterion which needs to be optimized on the complete problem. By solving conditionally independent subproblems, algorithms that exploit a tree decomposition of the problems often lose a global view on the criterion by exploiting loose local bounds. By providing strong lower bounds associated with each subproblem, the Russian Doll Search approach allows to inject more global information in each subproblem resolution through a strengthened initial upper bound, avoiding a lot of thrashing and offering important speedups.

Beside this, our experiments show that, even on problems that have a nice visible structure, it is often very profitable and sometimes crucial to restrict the maximum size of the separators of the decomposition exploited. Theory says that separator size influences the space complexity of the structural algorithm like BTD and RDS-BTD, but even with unbounded separator size, none of the instances considered here ran out of memory in our experiments. In practice, the improvement in efficiency is mostly explainable by the added freedom in variable ordering allowed by cluster merging, an observation consistent with [Jégou *et al.*, 2007] conclusions.

Our current algorithm still leaves areas for improvements. For example, the synergy between RDS and BTD could be improved by allowing BTD to cache on partial assignments. This would reduce redundant searches between successive RDS-BTD iterations and would be also useful if an iterative deepening strategy is used similarly to what has been done in [Cabon *et al.*, 1998].

Acknowledgments This research has been partly funded by the French *Agence Nationale de la Recherche* (`STALDECOPT` project).

References

[A. Choi *et al.*, 2008] A. Choi, N. Zaitlen, B. Han, K. Pipatsrisawat, A. Darwiche, and E. Eskin. Efficient Genome Wide Tagging by Reduction to SAT. In *Proc. of WABI-08*, volume 5251 of *LNCS*, pages 135–147, 2008.

[Cabon *et al.*, 1998] B. Cabon, S. de Givry, and G. Verfaillie. Anytime Lower Bounds for Constraint Optimization Problems. In *Proc. of CP-98*, pages 117–131, Pisa, Italy, 1998.

[Cabon *et al.*, 1999] B. Cabon, S. de Givry, L. Lobjois, T. Schiex, and J.P. Warners. Radio link frequency assignment. *Constraints Journal*, 4:79–89, 1999.

[Carlson *et al.*, 2004] C. S. Carlson, M. A. Eberle, M. J. Rieder, Q. Yi, L. Kruglyak, and D. A. Nickerson. Selecting a maximally informative set of single-nucleotide polymorphisms for association analyses using linkage disequilibrium. *Am. J. Hum. Genet.*, 74(1):106–120, 2004.

[Cooper and Schiex, 2004] M. Cooper and T. Schiex. Arc consistency for soft constraints. *Artificial Intelligence*, 154:199–227, 2004.

[Darwiche, 2001] A. Darwiche. Recursive Conditioning. *Artificial Intelligence*, 126(1-2):5–41, 2001.

[de Givry *et al.*, 2005] S. de Givry, M. Zytnicki, F. Heras, and J. Larrosa. Existential arc consistency: Getting closer to full arc consistency in weighted CSPs. In *Proc. of IJCAI-05*, pages 84–89, Edinburgh, Scotland, 2005.

[de Givry *et al.*, 2006] S. de Givry, T. Schiex, and G. Verfaillie. Exploiting Tree Decomposition and Soft Local Consistency in Weighted CSP. In *Proc. of AAAI-06*, Boston, MA, 2006.

[de Givry, 2004] S. de Givry. Singleton consistency and dominance for weighted CSP. In *Proc. of Soft Constraints workshop*, 2004.

[Freuder and Quinn, 1985] E. Freuder and M. Quinn. Taking advantage of stable sets of variables in constraint satisfaction problems. In *Proc. of IJCAI-85*, pages 1076–1078, Los Angeles, CA, 1985.

[Hirschhorn and Daly, 2005] J.N. Hirschhorn and M.J. Daly. Genome-wide association studies for common diseases and complex traits. *Nature Reviews Genetics*, 6(2):95–108, 2005.

[Jégou *et al.*, 2007] P. Jégou, S. N. Ndiaye, and C. Terrioux. Dynamic management of heuristics for solving structured CSPs. In *Proc. of CP-07*, pages 364–378, Providence, USA, 2007.

[Larrosa and Schiex, 2004] J. Larrosa and T. Schiex. Solving Weighted CSP by Maintaining Arc-consistency. *Artificial Intelligence*, 159(1-2):1–26, 2004.

[Larrosa *et al.*, 2002] J. Larrosa, P. Meseguer, and M. Sanchez. Pseudo-tree search with soft constraints. In *Proc. of ECAI-02*, pages 131–135, Lyon, France, 2002.

[Lecoutre *et al.*, 2006] C. Lecoutre, L. Sais, S. Tabary, and V. Vidal. Last conflict based reasoning. In *Proc. of ECAI-2006*, pages 133–137, Trento, Italy, 2006.

[Marinescu and Dechter, 2006] R. Marinescu and R. Dechter. Memory intensive branch-and-bound search for graphical models. In *Proc. of AAAI-06*, Boston, MA, 2006.

[Qin *et al.*, 2006] Z. S. Qin, S. Gopalakrishnan, and G. R. Abecasis. An efficient comprehensive search algorithm for tagsnp selection using linkage disequilibrium criteria. *Bioinformatics*, 22(2):220–225, 2006.

[Sanchez *et al.*, 2008] M. Sanchez, S. de Givry, and T. Schiex. Mendelian error detection in complex pedigrees using weighted constraint satisfaction techniques. *Constraints*, 13(1):130–154, 2008.

[Schiex *et al.*, 1995] T. Schiex, H. Fargier, and G. Verfaillie. Valued constraint satisfaction problems: hard and easy problems. In *Proc. of the 14^{th} IJCAI*, pages 631–637, Montréal, Canada, 1995.

[Terrioux and Jegou, 2003] C. Terrioux and P. Jegou. Hybrid backtracking bounded by tree-decomposition of constraint networks. *Artificial Intelligence*, 146(1):43–75, 2003.

[Verfaillie *et al.*, 1996] G. Verfaillie, M. Lemaître, and T. Schiex. Russian Doll Search for Solving Constraint Optimization Problems. In *Proc. of AAAI-96*, pages 181–187, Portland, OR, 1996.