# Max-CSP competition 2008:
# toulbar2 solver description

M. Sanchez[1], S. Bouveret[3], S. de Givry[1], F. Heras[2], P. Jégou[4], J. Larrosa[2], S. Ndiaye[4], E. Rollon[2], T. Schiex[1], C. Terrioux[4], G. Verfaillie[3], and M. Zytnicki[1]

[1] INRA, Toulouse, France
[2] Dep. LSI, UPC, Barcelona, Spain
[3] ONERA, Toulouse, France
[4] LSIS, Marseilles, France

**Abstract.** This document presents the key techniques used in toulbar2 solver submitted to the Max-CSP competition 2008. toulbar2 solves Weighted Constraint Satisfaction Problems (WCSPs), a generalisation of Max-CSP. Two complete solving methods that have been used for the competition are presented in this paper: Depth-First Branch and Bound (DFBB) and a new algorithm, Russian Doll Search with tree decomposition (RDS-BTD), which exploits the problem structure.

DFBB is commonly used to solve constraint optimization problems such as WCSPs. The worst-case time complexity of this algorithm can be improved by exploiting the constraint graph structure, identifying independent subproblems and caching their optima. However, the exploitation of the structure is done *a posteriori*: each time a new subproblem occurs, it has to be solved before its optimum can be used. RDS-BTD solves a relaxation of every subproblem before solving the whole problem, in the spirit of the Russian Doll Search algorithm. This relaxation allows to exploit subproblem lower bounds in a more proactive way.

## 1 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 standard 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 (or soft constraint) $w_S$ in $W$ is defined on a set of variables $S$ called its scope and 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 $\geq$. The operation $\ominus$ subtracts a cost $b$ from a larger cost $a$ where $a \ominus b = (a-b)$ if $a \neq m$ and $m$ otherwise.

For unary/binary cost functions, we use simplified notations: a unary (resp. binary) cost function on variable(s) $i$ (resp. $i$ and $j$) is denoted $w_i$ (resp. $w_{ij}$). If they do not exist, we add to $W$ a unary cost function $w_i$ for every variable $i$, and a nullary cost function, noted $w_\varnothing$ (a constant cost payed by any assignment). All these additional cost functions have initial cost 0, leaving the semantics of the problem unchanged.

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 usual projection of a tuple on the set of variables $S$. The problem of minimizing $Val_P(t)$ is an optimization problem with an associated NP-complete decision problem.

Enforcing a given local consistency property on a problem $P$ consists in transforming $P = (X,D,W,m)$ in a problem $P' = (X,D,W',m)$ which is equivalent to $P$ ($Val_P = Val_{P'}$) and which satisfies the considered local consistency property. This enforcing may increase $w_\varnothing$ and provide an improved lower bound on the optimal cost. Enforcing is achieved using Equivalence Preserving Transformations (EPTs) moving costs between different scopes [12, 8, 4, 6, 1, 3, 2].

A classical complete solving method is Depth-First Branch and Bound (DFBB). We give its pseudo-code in Algorithm 1. It enforces at each search node a given local consistency property Lc (line 1). The pruning condition is applied if the resulting $w_\varnothing \geq m$ (line 2). $m$ is updated to the cost of the last solution found (line 3). The initial call is DFBB($P$, $X$, $\emptyset$). It assumes an already local consistent problem $P$ and returns its optimum. $P/A$ denotes the subproblem $P$ under assignment $A$. The operator . is used to get an element of $P$. Function pop($S$) returns an element of $S$ and remove it from $S$.

DFBB worst-case time complexity is $O(d^n)$ and it uses linear space. In the next section, we briefly present how DFBB can be extended to exploit the problem structure.

## 2 Depth-First Branch and Bound with tree decomposition

Assuming connected problems, a tree decomposition of a 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 all 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 $V_e = C_e \setminus S_e$.

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.

*Example 1.* Consider the MaxCSP problem depicted in Figure 1. It has eleven variables with two values $(a,b)$ in their domains. Binary cost functions of difference ($w_{ij}(a,a) = w_{ij}(b,b) = 1, w_{ij}(a,b) = w_{ij}(b,a) = 0$) are represented by edges connecting the corresponding variables. In this problem, the optimal cost is 5 and it is attained with e.g. the assignment $(a,b,b,a,b,b,a,b,b,a,b)$ in lexicographic order. A $C_1$-rooted tree decomposition with clusters $C_1 = \{1,2,3,4\}, C_2 = \{4,5,6\}, C_3 = \{5,6,7\}, C_4 = \{4,8,9,10\}$,

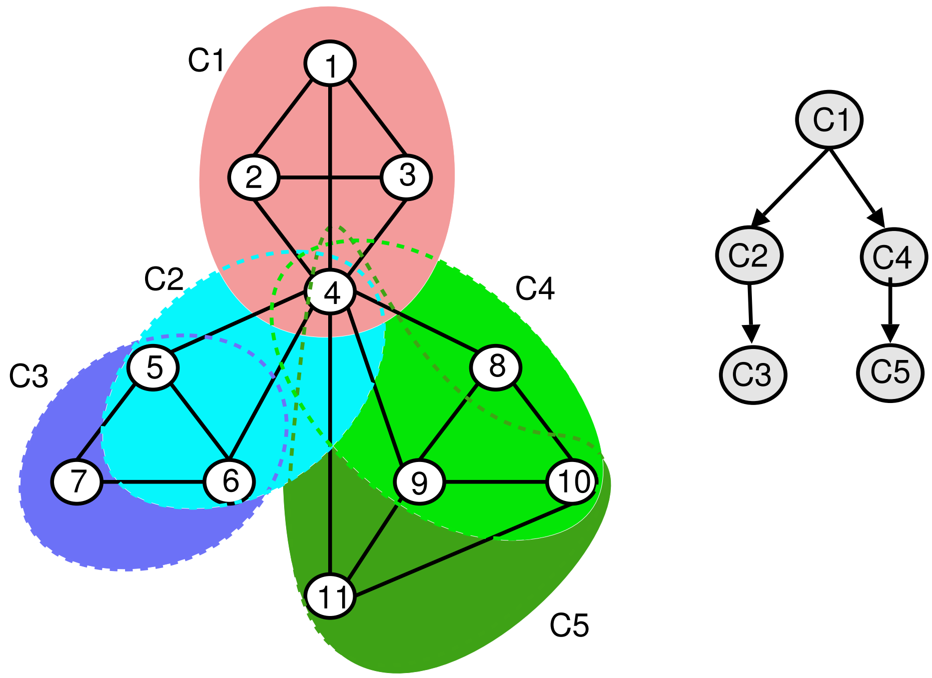


**Fig. 1.** The constraint graph of Example 1 and its associated tree decomposition.

and $C_5 = \{4, 9, 10, 11\}$, is given on the right hand-side in Figure 1. For instance, $C_1$ 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 $\{w_{5,7}, w_{6,7}, w_7\}$ ($w_7$ initially empty). $P_1$ corresponds to the whole problem.

Depth-First Branch and Bound with Tree Decomposition (BTD) [7, 5] exploits this property by restricting the variable ordering. Imagine all 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 $S_f$.

In [5], we show how to exploit a better initial upper bound for solving $P_f$. However this has the side-effect that the optimum of $P_f$ may be not computed but only a lower bound. The lower bound and the fact it is optimal can be recorded in $LB_{P_f/A_f}$ and $Opt_{P_f/A_f}$ respectively, initially set to 0 and *false*.

As in DFBB, BTD enforces local consistency during search. However, local consistency may move costs between clusters, thereby invalidating previously recorded information. We store these cost moves in a specific *backtrackable* data structure $\Delta W$ as defined in [5]. During the search, we can obtain the total cost that has been moved out of the subproblem $P_f/A_f$ by summing up all the $\Delta W_i^f(a)$ for all values $(i,a)$ in the separator assignment $A_f$ and correct any recorded information: $LB'_{P_f/A_f} = LB_{P_f/A_f} \ominus \bigoplus_{i\in S_f} \Delta W_i^f(A_f[i])$.

Moreover, we keep the nullary cost function local to each cluster: $w_\varnothing = \bigoplus_{C_e\in C} w_\varnothing^e$.

For pruning the search, BTD uses the maximum between local consistency and recorded lower bounds as soon as their separator is completely assigned by the current assignment $A$. We denote by $lb(P_e/A)$ this lower bound:

$$lb(P_e/A) = w_\varnothing^e \oplus \bigoplus_{C_f\in Sons(C_e)} \max(lb(P_f/A), LB'_{P_f/A_f}) \tag{1}$$

*Example 2.* In the problem of Example 1, variables $\{1,2,3,4\}$ of $C_1$ are assigned first, e.g. using a dynamic variable ordering *min domain / max degree* inside each cluster.

Let assume $A = \{(4,a),(1,a),(2,b),(3,b)\}$ be the current assignment[5]. Enforcing EDAC local consistency [6] on $P_1/A$ produces $w_\varnothing^1 = 2, w_\varnothing^2 = w_\varnothing^4 = 1, w_\varnothing^3 = w_\varnothing^5 = 0$, resulting in $lb(P_1/A) = \bigoplus_{C_e\in C} w_\varnothing^e = 4$ (no lower bound recorded yet).

Then, subproblems $P_2/\{(4,a)\}$ and $P_4/\{(4,a)\}$ are solved independently, resulting in $LB_{P_2/\{(4,a)\}} = 1$, $LB_{P_4/\{(4,a)\}} = 2$, $Opt_{P_2/\{(4,a)\}} = Opt_{P_4/\{(4,a)\}} = true$ (no initial upper bound) which are recorded. A first complete assignment of cost $w_\varnothing^1 \oplus LB_{P_2/\{(4,a)\}} \oplus LB_{P_4/\{(4,a)\}} = 5$ (all $\Delta W$ costs are zero in this case) is found.

In Algorithm 1, we present the pseudo-code of the BTD algorithm combining tree decomposition and a given level of local consistency Lc. This algorithm uses our initial enhanced upper bound (line 4), value removal based on local cuts [5] and lower bound recording (lines 6 and 7). The initial call is BTD($P_1$, $V_1$, $\emptyset$, 0), with $P_1 = P$, an already local consistent problem, returning its optimum.

The lower bound $lb(P_e/A)$ of Equation 1 does not take into account a possible recorded lower bound $LB_{P_e/A_e}$, which may exist if $Opt_{P_e/A_e}$ =*false* and the same subproblem is solved again. We therefore ensure a monotonically increasing lower bound during the search by passing the best lower bound found recursively (line 5 and 9), resulting in a stronger pruning condition (line 8).

BTD time complexity is $O(md^{w+1})$ with $w = \max_{C_e\in C}|C_e| - 1$, the maximum cluster size minus one, called the tree-width of the tree decomposition. Its memory complexity is bounded by $O(d^s)$ with $s = \max_{C_e\in C}|S_e|$, the maximum separator size [5].

## 3 Russian Doll Search with tree decomposition

The original Russian Doll Search (RDS) algorithm [13] 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

[5] Variable 4 has been selected first as it has the highest degree in $C_1$.

**Algorithm 1**: DFBB, BTD, and RDS-BTD algorithms.

**Function** DFBB($P$, $V$, $A$) : $[0,+\infty]$
  **if** $(V=\emptyset)$ **then**
    **return** $P.w_\varnothing$ */* A new solution is found for P */* ;
  **else**
    $i$ :=pop($V$) */* Choose an unassigned variable of P */* ;
    $d := P.D_i$ ;
    */* Enumerate every value in the domain of i */* ;
    **while** $(d \neq \emptyset$ and $P.w_\varnothing < P.m)$ **do**
      $a$ :=pop($d$) */* Choose a value */* ;
      $P'$ :=Lc($P/A \cup \{(i,a)\}$) */* Enforce local consistency on* $P/A \cup \{(i,a)\}$ **/* ;
      **if** $(P'.w_\varnothing < P.m)$ **then**
        $P.m$ :=DFBB($P'$, $V$, $A \cup \{(i,a)\}$) ;
    **return** $P.m$ ;

**Function** BTD($P_e$, $V$, $A$, $blb$) : $[0,+\infty]$
  **if** $(V=\emptyset)$ **then**
    $S := Sons(C_e)$ ;
    */* Solve all cluster sons whose optima are unknown */* ;
    **while** $(S \neq \emptyset$ and $lb(P_e/A) < P_e.m)$ **do**
      $C_f$ :=pop($S$) */* Choose a cluster son */* ;
      **if** $(not(Opt_{P_f/A_f}))$ **then**
        $P_f.m := P_e.m \ominus lb(P_e/A) \oplus lb(P_f/A_f)$ ;
        $res$ :=BTD($P_f$, $V_f$, $A$, $lb(P_f/A_f)$) ;
        $LB_{P_f/A_f} := res \oplus \bigoplus_{i\in S_f} \Delta W_i^f(A[i])$ ;
        $Opt_{P_f/A_f} := (res < P_f.m)$ ;
    **return** $lb(P_e/A)$ */* A new solution is found for* $P_e$ **/* ;
  **else**
    $i$ :=pop($V$) */* Choose an unassigned variable in* $C_e$ **/* ;
    $d := P_e.D_i$ ;
    */* Enumerate every value in the domain of i */* ;
    **while** $(d \neq \emptyset$ and $\max(blb, lb(P_e/A)) < P_e.m)$ **do**
      $a$ :=pop($d$) */* Choose a value */* ;
      $P'_e$ :=Lc($P_e/A \cup \{(i,a)\}$) */* Enforce local consistency on* $P_e/A \cup \{(i,a)\}$ **/* ;
      **if** $(\max(blb, lb(P'_e/A \cup \{(i,a)\})) < P_e.m)$ **then**
        $P_e.m$ :=BTD($P'_e$, $V$, $A \cup \{(i,a)\}$, $\max(blb, lb(P'_e/A \cup \{(i,a)\}))$) ;
    **return** $P_e.m$ ;

**Function** RDS-BTD($P$, $P_e^{RDS}$) : $[0,+\infty]$
  **foreach** $C_f \in Sons(C_e)$ **do**
    RDS-BTD($P$, $P_f^{RDS}$) ;
  $P_e^{RDS}.m := P.m \ominus lb(P/\emptyset) \oplus lb(P_e^{RDS}/\emptyset)$ ;
  $LB_{P_e^{RDS}}$ :=BTD($P_e^{RDS}$, $V_e$, $\{(i,$EAC(i)$) \mid i \in S_e\}$, $lb(P_e^{RDS}/\emptyset)$) ;
  Set to *false* all recorded $Opt_{P_f/A}$ such that $C_f$ is a descendant of $C_e$, $S_f \cap S_e \neq \emptyset$, $A \in \ell(S_f)$ ;
  **return** $LB_{P_e^{RDS}}$ ;

algorithm with a static variable ordering heuristic following the nested subproblem decomposition order. The lower bound combines the optimum of the previously solved subproblems with the lower bound produced by enforcing soft local consistency.

RDS-BTD, recently proposed in [10], applies the RDS principle to a tree decomposition. The main difference with RDS is that the set of subproblems to solve is defined by a rooted tree decomposition $(C,T)$.

We define $P_e^{RDS}$ 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. $P_e^{RDS}$ has no cost function involving a variable in $S_e$, the separator with its father, and thus its optimum is a lower bound of $P_e$ for any assignment of $S_e$.

RDS-BTD solves $|C|$ subproblems ordered by a depth-first traversal of $T$, starting from the leaves to the root $P_1^{RDS} = P_1$.

Each subproblem $P_e^{RDS}$ is solved by BTD instead of DFBB. This allows to exploit decomposition and caching done by BTD. Because caching is only performed on completely assigned separators, and considering all possible assignments of $S_e$ would be too costly in memory and time, we assign $S_e$ before solving $P_e^{RDS}$. This is needed since otherwise, caching on $P_f$, a descendant of $C_e$, with $S_f \cap S_e \neq \emptyset$, would use a partially assigned $A_f$. To assign $S_e$, we use the fully supported value of each domain[6] (maintained by EDAC [6]) as temporary values used for caching purposes only.

The advantage of using BTD is that recorded lower bounds can be reused during the next iterations of RDS-BTD. However, the optimum found by BTD for a given subproblem $P_f$ when solving $P_e^{RDS}$ is no more valid in $P_{Father(e)}^{RDS}$ due to possible cost functions between variables in $C_{Father(e)}$ and in $P_f$. At each iteration of RDS-BTD, after $P_e^{RDS}$ is solved, we reset all $Opt_{P_f/A_f}$ such that $S_f \cap S_e \neq \emptyset$ (line 12).

During search, RDS-BTD exploits the maximum between local consistency, recorded, and RDS lower bounds. Let $LB_{P_e^{RDS}}$ denote the optimum of $P_e^{RDS}$ found by one iteration of RDS-BTD. Because costs can be moved between clusters, this information has to be corrected in order to be valid in the next iterations of RDS-BTD. For that, we use the maximum of $\Delta W$ on each current domain of the (possibly unassigned) separator variables. The lower bound corresponding to the current assignment $A$ is then:

$$lb(P_e/A) = w_\varnothing^e \oplus \bigoplus_{C_f \in Sons(C_e)} \max(lb(P_f/A), LB'_{P_f/A_f}, LB_{P_f^{RDS}} \ominus \bigoplus_{i \in S_f} \max_{a \in D_i} \Delta W_i^f(a)) \quad (2)$$

*Example 3.* Applied on the problem of Example 1, RDS-BTD solves five subproblems $(P_3^{RDS}, P_2^{RDS}, P_5^{RDS}, P_4^{RDS}, P_1)$ successively. For instance, $P_3^{RDS}$ has variable $\{7\}$ and cost function $\{w_7\}$. Before solving $P_3^{RDS}$, RDS-BTD assigns variables $\{5,6\}$ of the separator $S_3$ to their fully supported value ($\{(5,a),(6,a)\}$ in this example). In solving $P_2^{RDS}$, it can record e.g. the optimum of $P_3/\{(5,a),(6,a)\}$, equal to zero (recall that $w_{5,6}$ does not belong to $P_3$), that can be reused when solving $P_1$. In solving $P_4^{RDS}$, it can record e.g. the optimum of $P_5/\{(4,a),(9,a),(10,a)\}$, also equal to zero. However, due to the fact that variable 4 belongs to $S_5 \cap S_4$ and $P_4^{RDS}$ does not contain $w_{4,11}$, this recorded information is only a lower bound for subsequent iterations of RDS-BTD. So, we set

[6] 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 $w_S(t) = 0$.

$Opt_{P_5/\{(4,a),(9,a),(10,a)\}}$= *false* before solving $P_1$. The resulting optima are: $LB_{P_3^{RDS}} = LB_{P_5^{RDS}} = 0, LB_{P_2^{RDS}} = LB_{P_4^{RDS}} = 1$ and $LB_{P_1^{RDS}} = 5$, the optimum of $P_1$.

In this simple example, for $A = \{(4,a),(1,a),(2,b),(3,b)\}$, $lb(P_1/A)$ using Equation 1 or 2 is the same because EDAC propagation provides lower bounds equal to RDS lower bounds. In the contrary, for $A = \emptyset$, $lb(P_1/\emptyset) = LB_{P_2^{RDS}} \oplus LB_{P_4^{RDS}} = 2$ using Equation 2 and $lb(P_1/\emptyset) = 0$ using Equation 1 (assuming EDAC local consistency in preprocessing and no initial upper bound).

We present the pseudo-code of the RDS-BTD algorithm in Algorithm 1. RDS-BTD call BTD to solve each subproblem $P_e^{RDS}$ (line 11), using Equation 2 instead of Equation 1 to compute lower bounds. An initial upper bound for $P_e^{RDS}$ is deduced from the global problem upper bound and the already computed RDS lower bounds (line 10). It initially assigns variables in $S_e$ to their fully supported value (given by EAC function at line 11) as discussed above. The initial call is RDS-BTD($P$, $P_1^{RDS}$). It assumes an already local consistent problem $P_1^{RDS} = P$ and returns its optimum.

Notice that as soon as a solution of $P_e^{RDS}$ is found having the same optimal cost as $lb(P_e^{RDS}/\emptyset) = \bigoplus_{C_f \in Sons(C_e)} LB_{P_f^{RDS}}$, then the search ends thanks to the initial lower lound given at line 11.

The time and space complexity of RDS-BTD is the same as BTD.

## 4 Implementation details

We implemented DFBB and RDS-BTD in an open-source C++ solver named `toulbar2`[7]. DFBB uses default parameter values of `toulbar2`.

Dynamic variable ordering (*min domain / max degree*, breaking ties with maximum unary cost) is used inside clusters (RDS-BTD) and by DFBB. EDAC local consistency is enforced on binary [6] and ternary [11] cost functions during search. Larger arity cost functions are delayed from propagation until they become ternary or less.

We use the Maximum Cardinality Search heuristic to build a tree decomposition and choose the largest cluster as the root. In order to relax the restriction imposed by RDS-BTD on the dynamic variable ordering heuristic, we propose to merge clusters with their parent if their separator is too large. Starting from the leaves of a given tree decomposition, we merge a cluster with its parent if the separator size is strictly greater than $r = 4$ (parameter *B2r4* in `toulbar2`).

Recorded (and if available RDS) lower bounds are exploited by local consistency enforcing as soon as their separator variables are fully assigned. If the recorded lower bound is optimal ($Opt_{P_e/A_e}$ =*true*) or strictly greater than the one produced by local consistency, i.e. $\max(LB'_{P_e/A_e}, LB_{P_e^{RDS}} \ominus \bigoplus_{i \in S_e} \Delta W_i^e(A[i])) > \bigoplus_{P_f \subseteq P_e} w_\varnothing^f$, then the corresponding subproblem ($P_e/A_e$) is deconnected from local consistency enforcing and the positive difference in lower bounds is added to its parent cluster lower bound ($w_\varnothing^{Father(C_e)}$), allowing possible new value removals by node consistency enforcing on the remaining problem.

[7] Version 0.7 available at http://mulcyber.toulouse.inra.fr/gf/project/toulbar2

All the solving methods exploit a binary branching scheme depending on the domain size $d$ of the branching variable. If $d > 10$ then it splits the *ordered* domain into two parts (by taking 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-BTD where it selects the branch which contains the value corresponding to the last solution(s) found first if available.

At each search node, before branching, DFBB and RDS-BTD eliminate all variables (except variables occuring in a separator for RDS-BTD) with a degree less than or equal to two, possibly creating new binary cost functions on the fly. They apply successively EDAC propagation (which may assign some variables and reduce current degrees) and 2-degree variable elimination until there is no more elimination nor propagation.

The dynamic variable ordering heuristic is modified by a conflict back-jumping heuristic as suggested in [9]. It branches on the same variable again if the first branch in the binary branching scheme was directly pruned by propagation.

No initial upper bound is provided.

*Acknowledgments* `toulbar2` solver has been partly funded by the French *Agence Nationale de la Recherche* (`STALDECOPT` project).

## References

1. M. Cooper. High-order consistency in valued constraint satisfaction. *Constraints*, 10(3):283–305, 2005.
2. M. Cooper, S. de Givry, M. Sanchez, T. Schiex, and M. Zytnicki. Virtual arc consistency for weighted csp. In *Proc. of AAAI-08*, Chicago, IL, 2008.
3. M. Cooper, S. de Givry, and T. Schiex. Optimal soft arc consistency. In *Proc. of IJCAI-07*, pages 68–73, Hyderabad, India, 2007.
4. M. Cooper and T. Schiex. Arc consistency for soft constraints. *Artificial Intelligence*, 154:199–227, 2004.
5. 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.
6. 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.
7. P. Jégou and C. Terrioux. Decomposition and good recording. In *Proc. of ECAI-2004*, pages 196–200, Valencia, Spain, 2004.
8. J. Larrosa and T. Schiex. Solving Weighted CSP by Maintaining Arc-consistency. *Artificial Intelligence*, 159(1-2):1–26, 2004.
9. C. Lecoutre, L. Sais, S. Tabary, and V. Vidal. Last conflict based reasoning. In *Proc. of ECAI-2006*, pages 133–137, Trento, Italy, 2006.
10. M. Sanchez, D. Allouche, S. de Givry, and T. Schiex. Russian doll search with tree decomposition. In *Workshop on Preferences and Soft Constraints*, Sydney,Australia, 2008.
11. 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.
12. T. Schiex. Arc consistency for soft constraints. In *Proc. of CP-2000*, pages 411–424, Singapore, 2000.
13. 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.