Chapter 20

Dynamic heuristics for branch and bound search on tree-decomposition of Weighted CSPs

This paper deals with methods exploiting tree-decomposition approaches for solving weighted constraint networks. We consider here the practical efficiency of these approaches by defining three classes of variable orders more and more dynamic which preserve the time complexity bound. Then, we extend this theoretical time complexity bound to increase the dynamic aspect of these orders. For that, we define a constant k allowing us to extend the classical bound from $O(exp(w+1))$ firstly to $O(exp(w+k+1))$, and finally to $O(exp(2(w+k+1)-s^{-}))$, with w the "tree-width" of a Weighted CSP and s^{-} the minimum size of its separators. Finally, we empirically assess the defined theoretical extensions of the time complexity bound.

20.1. Introduction

The CSP formalism (Constraint Satisfaction Problem) offers a powerful framework for representing and solving efficiently many problems. Modeling a problem as a CSP consists in defining a set X of variables $x_1, x_2, \ldots x_n$, which must be assigned in their respective finite domain, by satisfying a set C of constraints which express restrictions between the different possible assignments. A solution is an assignment of every variable which satisfies all the constraints. Determining if a solution exists is a NP-complete problem. This framework has been extended in order to capture notions like preference or possibility or,

Chapter written by Philippe JÉGOU, Samba Ndojh NDIAYE and Cyril TERRIOUX.

when there is no solution, to produce an assignment minimizing a given criterion on constraint satisfaction. Hence, recently, many extensions of the CSP framework have been proposed (e.g. [FRE 92, BIS 95, SCH 95]).

In this paper, we focus our study on Weighted CSPs (WCSPs) which is a well known framework for soft constraints. In this extension, a weight (or a cost) is associated with each tuple of each constraint. So, each assignment has a cost defined by the sum of the costs of all the tuples included in the considered assignment. Solving a WCSP instance requires to find an assignment whose cost is minimum. This task is NP-hard. Many algorithms have been defined in the past years for solving this problem. On the one hand, the usual complete method for solving WCSPs is based on branch and bound search, which, in order to be efficient, must use both filtering techniques and heuristics for choosing the next variable or value. This approach, often efficient in practice, has an exponential theoretical time complexity in $O(exp(n))$ for an instance having n variables. On the other hand, some other methods are based on the dynamic programming approach (e.g. [VER 96, KOS 99, MES 00, MES 01, MES 02, LAR 03]). Some of them exploit the problem structure like [KOS 99, MES 00, LAR 02, LAR 03]. Exploiting the structure often leads to improve the solving methods and, in particular, to provide better theoretical time complexity bounds. Several bounds exist like the induced width [DEC 99] or the tree-height [FRE 85, MAR 04]. Yet, the best known complexity bounds are given by the "tree-width" of a WCSP (often denoted w). This parameter is related to some topological properties of the constraint graph which represents the interactions between variables via the constraints. It leads to a time complexity in $O(exp(w+1))$. Different methods have been proposed to reach this bound (see [GOT 00] for a survey and a theoretical comparison of these methods). They rely on the notion of tree-decomposition of the constraint graph. They aim to cluster variables such that the cluster arrangement is a tree. Depending on the instances, we can expect a significant gain w.r.t. enumerative approaches. Most of these works only present theoretical results. Few practical results have been provided (e.g. [GOT 02, JÉG 04]). So, we study these approaches by focusing on the BTD method (for Backtracking with Tree-Decomposition [JÉG 03]) which seems to be one of the most effective structural method.

The problem of finding the best decomposition (w.r.t. the tree-width) has been firstly studied in the literature from a theoretical viewpoint. More recently, some studies (e.g. [JÉG 05]) have been realized in the field of CSP, integrating as quality parameter for a decomposition, its efficiency for solving the considered CSP. Yet, these studies do not consider the questions related to an efficient use of the considered decomposition. This paper deals with this question. Given a tree-decomposition, we study the problem of finding good

orders on variables for exploiting this decomposition in a branch and bound search like one achieved by BTD. Similar works have been already performed for SAT or CSP (e.g. [HUA 03, JÉG 06]). As presented in [TER 03, JÉG 04], the order on the variables is static and compatible with a depth first traversal of the associated cluster tree. Since enumerative methods highlight the efficiency of dynamic variable orders, we give conditions which allow to exploit in a more dynamic way the tree-decomposition and guarantee the time complexity bound. We propose five classes of orders respecting these conditions, two of them giving more freedom to order variables dynamically. Consequently, their time complexity possesses larger bounds: $O(exp(w+k+1))$ and $O(exp(2(w+k+1)-s^{-}))$, where s^{-} is the minimum size of the separators and k is a constant to parameterize. Based on the properties of these classes, we define several heuristics which aim to compute a good order on clusters and more generally on variables. They rely on topological and semantic properties of the WCSP instance. Heuristics based on a dynamic variable ordering and those based on the expected number of solutions enhance significantly the performances of BTD. Finally, we report here experiments to assess the interest of our propositions.

This paper is organized as follows. Section 20.2 provides basic notions about WCSPs and BTD. Then, in section 20.3, we define several classes of variable orders which preserve the classical time complexity bound. Section 20.4 introduces two extensions giving new time complexity bounds. Section 20.5 presents the different heuristics we use for guiding the exploration of the cluster tree. Then, section 20.6 is devoted to empirical results. Finally, in section 20.7, we conclude and outline some future works.

20.2. Preliminaries

A *constraint satisfaction problem* (CSP) is defined by a tuple (X, D, C). X is a set $\{x_1, \ldots, x_n\}$ of n variables. Each variable x_i takes its values in the finite domain d_{x_i} from D. The variables are subject to the constraints from C. Given an instance (X, D, C), the CSP problem consists in determining if there is a solution (i.e. an assignment of each variable which satisfies each constraint). This problem is NP-complete. In this paper, we focus our study on an extension of CSPs, namely Weighted CSPs (WCSPs). In this extension, a weight (or a cost) is associated with each tuple of each constraint. The cost of a tuple allowed by a constraint is 0, while a forbidden one has a cost greater than 0. Then, each assignment has a cost defined by the sum of the costs of all the tuples included in the considered assignment. Solving a WCSP instance requires to find an assignment whose cost is minimum. This task is NP-hard. In this paper, without loss of generality, we only consider binary constraints (i.e. constraints which involve two variables). So, the structure of

a WCSP can be represented by the graph (X, C), called the *constraint graph*. The vertices of this graph are the variables of X and an edge joins two vertices if the corresponding variables share a constraint.

Methods providing interesting theoretical time complexity bounds often rely on the structure of the constraint graph, and in particular the notion of tree-decomposition of graphs [ROB 86]. Let $G = (X, C)$ be a graph, a *tree-decomposition* of G is a pair (E, T) where $T = (I, F)$ is a tree with nodes I and edges F and $E = \{E_i : i \in I\}$ a family of subsets of X, such that each subset (called cluster) E_i is associated to a node of T and verifies:

1) $\cup_{i \in I} E_i = X$,

2) for each edge $\{x, y\} \in C$, there exists $i \in I$ with $\{x, y\} \subseteq E_i$,

3) for all $i, j, k \in I$, if k is in a path from i to j in T, then $E_i \cap E_j \subseteq E_k$.

The width of a tree-decomposition (E, T) is equal to $max_{i \in I}|E_i| - 1$. The *tree-width* w of G is the minimal width over all the tree-decompositions of G.

Assume that we have a tree-decomposition of minimal width w, the reference structural method, Tree-Clustering [DEC 89], has a time complexity in $O(exp(w + 1))$ while its space complexity can be reduced to $O(n.s.d^s)$ with s the size of the largest minimal separators of the graph [DEC 01]. Note that Tree-Clustering does not provide interesting results in practical cases. So, an alternative approach, also based on a tree-decomposition of graphs was proposed in [TER 03]. This method, called BTD, seems to provide empirical results among the best ones obtained by structural methods.

BTD proceeds by a branch and bound search guided by a static pre-established partial order induced by a tree-decomposition of the constraint network. So, the first step of BTD consists in computing a tree-decomposition of the constraint graph. The computed tree-decomposition induces a partial variable ordering which allows BTD to exploit some structural properties of the graph and so to prune some parts of the search tree. In fact, variables are assigned according to a depth-first traversal of the rooted tree. In other words, we first assign the variables of the root cluster E_1, then we assign the variables of E_2, then E_3's ones, and so on. For example, $(x_1, x_2, \ldots, x_{14})$ or $(x_2, x_1, x_4, x_3, x_6, x_5, x_7, x_8, x_9, x_{11}, x_{10}, x_{12}, x_{13}, x_{14})$ are possible variable orderings for the problem whose constraint graph is presented in figure 20.1. Furthermore, the tree-decomposition and the variable ordering allow BTD to divide the problem $\mathcal{P}$ into several subproblems. Given two clusters E_i and E_j (with E_j a E_i's son), the subproblem rooted in E_j depends on the current assignment $\mathcal{A}$ on the separator $E_i \cap E_j$. It is denoted $\mathcal{P}_{\mathcal{A}, E_i/E_j}$. Its variable set is equal to $Desc(E_j)$ where $Desc(E_j)$ denotes the set of variables belonging to the cluster E_j or to any descendant E_k of E_j in the cluster tree rooted in E_j.

The domain of each variable in $E_i \cap E_j$ is restricted to its value in $\mathcal{A}$. Regarding the constraint set, it contains the constraints which involve at least one variable which exclusively appears in E_j or in a descendant of E_j. For instance, let us consider the WCSP whose constraint graph and a possible tree-decomposition are provided in figure 20.1. Given an assignment $\mathcal{A}$ on $E_2 \cap E_3$, the variable set of $\mathcal{P}_{\mathcal{A},E_2/E_3}$ is $Desc(E_3) = \{x_5, x_6, x_7, x_8, x_9\}$, and its constraint set is $\{c_{57}, c_{58}, c_{67}, c_{68}, c_{78}, c_{79}, c_{89}\}$ (with c_{ij} the constraint involving the variables x_i and x_j). Note that c_{56} does not belong to this constraint set since both x_5 and x_6 appear in E_2. Finally, the tree-decomposition notion permits to define the *valued structural good* notion. A valued structural good of E_i w.r.t. a son E_j is a pair $(\mathcal{A}, v)$ with $\mathcal{A}$ an assignment on $E_i \cap E_j$ and v the optimal cost of the subproblem $\mathcal{P}_{\mathcal{A},E_i/E_j}$.

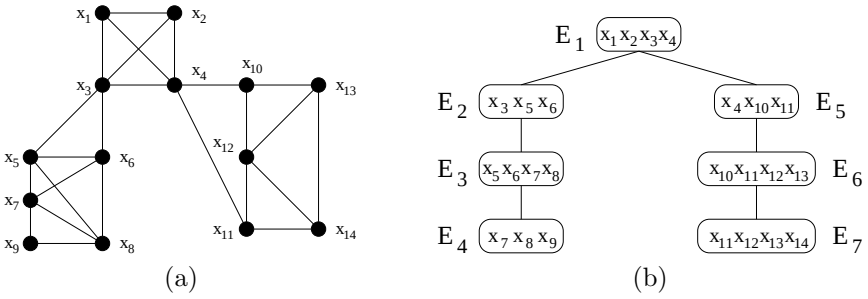

Figure 20.1. *(a) A graph and (b) a possible tree-decomposition.*

To satisfy the bounds of complexity, the variable ordering exploited in BTD is related to the cluster ordering. Formally, we consider a tree-decomposition (E, T) of the WCSP. We suppose that the elements of $E = \{E_i : i \in I\}$ are indexed w.r.t. the notion of *compatible numeration*. A numeration on E compatible with a prefix numeration of the tree T with E_1 the root is called compatible numeration. An order $\preceq_X$ of variables of X such that $\forall x \in E_i$, $\forall y \in E_j$, with $i < j$, $x \preceq_X y$ is a compatible enumeration order. The cluster numeration gives a partial order on the variables since the variables in E_i are assigned before those in E_j if $i < j$, except variables in the descent of a good, namely those located in the subproblem rooted on the cluster containing the good. In fact, using goods allows not to explore twice subproblems if their optimal cost is known. If we use a good $(\mathcal{A}, v)$ to avoid visiting again a subtree, we know that the variables in it can be optimally assigned with a cost v. So BTD does not assign them effectively, but they are considered done. They are named *assignable variables* thanks to a good. Of course, if we are interested in providing an optimal assignment, an additional work must be performed to assign these variables at the end of the search [JÉG 04]. Thus the variables in E_j are assigned if the variables in E_i are either already assigned or assignable

thanks to a good. To complete this order, we have to choose variable ordering heuristics inside a cluster. Finally, a compatible enumeration order on the variables is given by a compatible numeration on clusters and a variable order in each cluster.

In [JÉG 04], the results were presented without heuristics for the choice of the clusters and thus the choice of the variables, except on the level of the order used inside the cluster which corresponded to a traditional dynamic order. Obviously, the variable ordering have a great impact on the efficiency of enumerative methods. Thus, we study here how the benefits of variable orderings can be fully exploited in BTD. Yet, to guarantee the time complexity bound, it is necessary to respect some conditions. So, in the next section, we define classes of orders guaranteeing the complexity bounds.

20.3. Dynamic orders in $O(exp(w+1))$

The first version of BTD was defined with a compatible static variable ordering. We prove here that more dynamic orders can be considered without loosing the complexity bounds. The defined classes contain orders more and more dynamic. These orders are provided by the cluster order and the variable order inside each cluster.

Let $\mathcal{P} = (X, D, C)$ be a WCSP and (E, T) a tree-decomposition of the graph (X, C), we exploit an order σ_i on the subproblems $\mathcal{P}_{i_1}, \ldots, \mathcal{P}_{i_k}$ rooted on the sons E_{i_j} of E_i and an order γ_i on the variables in E_i. We define recursively the following classes of orders. In the three next classes, we choose the first cluster to assign (the root): E_1 among all the clusters and we consider $\mathcal{P}_1$ the subproblem rooted on E_1 (i.e. $\mathcal{P}$).

DEFINITION.– *We begin the search in E_1 and we try recursively to extend the current assignment on the subproblem rooted on E_i by assigning first the variables in E_i according to γ_i and then on $\mathcal{P}_{i_1}, \ldots, \mathcal{P}_{i_k}$ according to σ_i.*

– ***Class 1.*** *σ_i* ***and*** *γ_i* ***are static****. We compute σ_i and γ_i statically (before starting the search).*

– ***Class 2.*** *σ_i* ***is static and*** *γ_i* ***is dynamic****. We compute statically σ_i, while γ_i is computed during the search.*

– ***Class 3.*** *σ_i* ***and*** *γ_i* ***are dynamic****. Both, σ_i and γ_i are computed during the search. σ_i is computed w.r.t. the current assignment as soon as all the variables in E_i are assigned.*

– ***Class ++. Enumerative dynamic order****. The variable ordering is completely dynamic. So, the assignment order is not necessarily a compatible enumeration order. There is no restriction due to the cluster tree.*

The defined classes form a hierarchy since we have: *Class 1* $\subset$ *Class 2* $\subset$ *Class 3* $\subset$ *Class ++*.

Property of the *Class 3*. Let Y be an assignment, $x \in E_j - (E_i \cap E_j)$ with E_i the parent of E_j: $x \in Y$ iff: i)$\forall v \in E_i$, $v \in Y$ ii) Let $E_{i_p} = E_j$, $\forall \mathcal{P}_{i_u}$ s.t. $\sigma_i(\mathcal{P}_{i_u}) \leq \sigma_i(\mathcal{P}_{i_p})$, $\forall v \in \mathcal{P}_{i_u}$, $v \in Y$ iii) $\forall v \in E_j$ s.t. $\gamma_j(v) \leq \gamma_j(x)$, $v \in Y$.

In [JÉG 04], the experiments use *Class 2* orders. Formally, only the orders of the *Class 1* are compatible. Nevertheless, for an order o_3 in the *Class 3* and a given assignment $\mathcal{A}$, one can find an order o_1 in the *Class 1* that assigns the variables in $\mathcal{A}$ in the same way and the same order o_3 does. This property gives to the *Class 3* (thus *Class 2*) orders the ability of recording goods and using them to prune branches in the same way *Class 1* orders do. The *Class ++* gives a complete freedom. Yet, it does not guarantee the time complexity bound because sometimes it is impossible to record goods. Indeed an order of *Class ++* may lead to assign some variables of a cluster E_j (with E_j a son of a cluster E_i) without having assigned the variables of the separator $E_i \cap E_j$. By so doing, we cannot safely compute the optimal solution of the subproblem rooted in E_j and so it is impossible to record a good on $E_i \cap E_j$. Hence, a subproblem may be solved several times and thus the time complexity bound is not guaranteed anymore. Meanwhile, the *Class 3* orders guarantee this bound.

THEOREM.– *The time complexity of BTD with a Class 3 order is* $O(exp(w+1))$.

PROOF.– *We consider a cluster* E_j *in the cluster tree, and we must prove that any assignment on* E_j *is computed only once. Let* E_i *be the cluster parent of* E_j *and suppose that all the variables in* E_i *are assigned and those in* $E_j - (E_i \cap E_j)$ *are not. Since the order belongs to the Class 3, the variables of the clusters on the path from the root to* E_i *are already assigned and those exclusively in the subtree rooted on* E_j *not yet. An assignment* $\mathcal{A}$ *on* $E_i \cap E_j$ *is computed at the latest when the variables in* E_i *are assigned (before those in the subproblem rooted on* E_j*). Solving the subproblem rooted on* E_j *leads to the computation of its optimal cost* v*. Then,* $(\mathcal{A}, v)$ *is recorded as a good. Let* $\mathcal{A}'$ *be an assignment on* E_j *compatible with* $\mathcal{A}$*. The next assignment of variables in* E_i *leading to* $\mathcal{A}$ *on* $E_i \cap E_j$ *will not be pursued on the subproblem rooted on* E_j *thanks to* $(\mathcal{A}, v)$*.* $\mathcal{A}'$ *is not computed twice, only the variables in* $E_i \cap E_j$ *are assigned again. So the time complexity is* $O(exp(w+1))$.

The properties of the *Class 3* offer more possibilities in the variable ordering. So it is possible to choose any cluster to visit next among the sons of the current cluster. And in each cluster, the variable ordering is totally free. In the next section, we propose two natural extensions of the complexity bound.

20.4. Bounded extensions of dynamic orders

We propose two extensions based on the ability given to the heuristics to choose the next variable to assign not only in one cluster, but also among k variables in a path rooted on the cluster that verifies some properties. So, we define two new classes of orders extending *Class 3*. First, we propose a generalization of the tree-decomposition definition before providing the definition of the *Class 4*.

DEFINITION.– *Let $G = (X, C)$ be a graph and k a positive integer, the set of directed k-covering tree-decompositions of a tree-decomposition (E, T) of G with E_1 its root cluster, is defined by the set of tree-decompositions (E', T') of G that verify:*

– *$E_1 \subseteq E'_1$, E'_1 the root cluster of (E', T')*

– *$\forall E'_i \in E'$, $E'_i \subseteq E_{i_1} \cup E_{i_2} \cup \ldots \cup E_{i_R}$ and $E_{i_1} \cup E_{i_2} \cup \ldots \cup E_{i_R-1} \subset E'_i$, with $E_{i_1} \ldots E_{i_R}$ a path in T*

– *$|E'_i| \leq w + k + 1$, where $w = max_{E_i \in E}|E_i| - 1$*

DEFINITION.– *Let (X, D, C) be a WCSP, (E, T) a tree-decomposition of the graph (X,C) and k a positive integer. A variable order is in the **Class 4**, if this order is in the Class 3 for one directed k-covering tree-decomposition of (E, T).*

We derive a natural theorem:

THEOREM.– *The time complexity of BTD with a Class 4 order is $O(exp(w + k + 1))$.*

PROOF.– *This proof is similar to one given for Class 3 since we can consider that BTD runs on a tree-decomposition (E', T') of width at most $w + k + 1$.*

A second extension is possible in exploiting during the search, a dynamic computing of the tree-decomposition (we can use several directed k-covering tree-decompositions during the search). Then, the time complexity bound changes because sometimes it would be impossible to record goods.

DEFINITION.– *Let (X, D, C) be a WCSP, (E, T) a tree-decomposition of the graph (X,C) and k a positive integer. A variable order o_5 is in the **Class 5**, if for a given assignment $\mathcal{A}$, one can find one directed k-covering tree-decomposition (E', T') of (E, T) such that $\forall E'_i \in E'$, $E'_i = E_{i_1} \cup E_{i_2} \cup \ldots \cup E_{i_R}$, with $E_{i_1} \ldots E_{i_R}$ a path in T and an order o_3 on (E', T') in the Class 3 that assigns the variables in $\mathcal{A}$ in the same way and the same order o_5 does.*

THEOREM.– *The time complexity of BTD with a Class 5 order is* $O(exp(2(w+k+1)-s^-))$.

PROOF.– *We have to prove that any assignment on a set* V *of* $2(w+k+1)-s^-$ *variables on a path of the tree* T *is computed only once. Let* $\mathcal{A}$ *be an assignment containing* V*. The order in which the variables of* $\mathcal{A}$ *were assigned is in the Class 3 for a directed* k*-covering tree-decomposition* (E',T') *of* (E,T) *that verifies* $\forall E'_i \in E'$, $E'_i = E_{i_1} \cup E_{i_2} \cup \ldots \cup E_{i_R}$, *with* $E_{i_1} \ldots E_{i_R}$ *a path in* T*. The size of the clusters in* (E',T') *is bound by* $w+k+1$*, so the set* V *is covered by at least two clusters since* s^- *is the minimum size of the separators. Let* $E'_{i_1} \ldots E'_{i_q}$ *be a path on* (E',T') *covering* V*. The solving of the subproblem rooted on* E'_{i_1} *with the assignment* $\mathcal{A}$ *leads to the recording of goods on the separators of these clusters. If* E'_{i_1} *is the root cluster of* (E',T')*, then* V *contains the root cluster* E_1 *of* (E,T)*. Thus* $\mathcal{A}$ *will not be computed again because it contains the first variables in the search. We suppose that* E'_{i_1} *is not the root cluster of* (E',T')*. Since* $q \geq 2$*, we record a good on the separator of* E'_{i_1} *and its parent and at least another on the separator* $E'_{i_1} \cap E'_{i_2}$*. Let* $\mathcal{B}$ *be a new assignment that we try to extend on* V *with the same values in* $\mathcal{A}$*. One of the goods will be computed first. Thus before all the variables in* V *are assigned, the search is stopped thanks to this good. So* $\mathcal{A}$ *is not computed again. We prove that any assignment on* V *is computed only once.*

Note that the new defined classes are included in the hierarchy presented in section 20.3: *Class 3* $\subset$ *Class 4* $\subset$ *Class ++* and *Class 3* $\subset$ *Class 5* $\subset$ *Class ++*.

To define the value of k, we have several approaches to choose variables to group. A good one consists in trying to reduce the value of the parameter s (the minimum size of the tree-decomposition separators) and, by this way, to enhance the space complexity bound. Then, we can observe that grouping clusters with large separators permits to achieve a significant reduction of s.

20.5. Heuristics

We propose here several heuristics in order to improve the performances of BTD.

20.5.1. *Cluster orders*

We first define several heuristics for computing the order the clusters are visited for the *Classes 1, 2* and 3. They are static for the *Class 1* and dynamic for the *Classes 2* and 3. They consist in choosing the first visited cluster (called

the root cluster) and ordering the sons of each cluster. These orders are used by assuming that the early use of goods does not enforce another order. Indeed, this early use of goods improves a lot the method, by increasing faster the lower bound. In fact, as soon as all the variables in the separator between the current cluster and one of its sons are assigned, we check whether this assignment is a good. If so, we do not explore the subtree rooted on this son cluster and the cost related to this good is added to the lower bound.

Static orders A static order is defined before the search begins. We propose criteria for the choice of the root cluster:

– *minexp*: this heuristic is based on the expected number of partial solutions of clusters [SMI 94] and on their size. Exploiting the expected number of solutions may appear surprising in the WCSP framework. However, some subproblems may have solutions while the whole problem has none. If so, it could be interesting to begin the search with the subproblems having no solution since they have a positive cost, what may result in increasing quickly the lower bound. The heuristic chooses as root cluster one which minimizes the ratio between the expected number of solutions and the size of the cluster. It allows to start the exploration with a large cluster having few solutions or no solution.

– *size*: this local criterion chooses the cluster of maximum size as root cluster

– *bary*: it is a global criterion based on the location of the cluster in the tree. For this criterion, we use the notion of distance, denoted $dist(x, y)$, between two vertices x and y of a graph G, which is defined by the length of a shortest path between x and y. A *barycentre* of G is a vertex x s.t. x minimizes $\Sigma_{y \in X} dist(x, y)$. The *bary* heuristic chooses a barycentre cluster as a root cluster.

Likewise, we propose heuristics for ordering cluster sons:

– $minexp_s$: this heuristic is similar to *minexp* and orders the son clusters according to the increasing value of their ratio.

– $minsep_s$: we order the son clusters according to the increasing size of their separator with their parent.

Dynamic orders A dynamic order is defined during the search. But, the choice of the root cluster is done at the beginning of the search. So one can only use static heuristics to choose the root. We also propose a new heuristic: *nv*. The dynamic variable ordering heuristics improve very significantly the runtime of enumerative methods. To derive benefit of this property, we choose a dynamic variable ordering heuristic and the root cluster is one containing the first variable w.r.t. the chosen variable order. The dynamic aspect of the cluster orders is in the son cluster ordering.

– $minexp_{sdyn}$: the next cluster to visit minimizes the ratio between the current expected number of solutions and the size of the cluster. The current expected number of solutions of a cluster is modified by filtering the domains of unassigned variables. So we compute this number for unordered clusters as soon as their parent is fully assigned. So the choice of the next cluster is more precise.

– nv_{sdyn}: this heuristic is similar to nv. We visit first the son cluster where appears the next variable in the variable order among the variables of the unvisited son clusters.

20.5.2. *Variable orders*

We define here static and dynamic variable orders according to which the variables inside a cluster are assigned.

Static orders A static order is defined before the search begins.

– mdd: the variables are ordered according to the increasing value of the ratio domain/degree. This heuristic gives good results compared to other static ones.

Dynamic orders A dynamic order is defined during the search.

– mdd_{dyn}: the next variable to assign minimizes the ratio domain/degree. The current ratio of a variable is modified by the domain filtering. So we compute again this number each time the domain is filtered. This heuristic gives very good results.

20.5.3. *Heuristics for grouping variables in the Classes 4 and 5*

Grouping variables gives more freedom for dynamic variable ordering heuristics which may improve significantly the enumerative methods runtime. Furthermore, it is necessary to find a good value of the parameter k besides which BTD does not profit sufficiently of the problem structure and therefore its time complexity increases a lot. We propose several criteria for grouping variables. For *Class 4* orders, these criteria are exploited as a preliminary step before computing the order.

– sep: this heuristic has one parameter which is the maximum size of separators. We merge clusters $< parent, son >$ if their separator size exceeds the value of the parameter.

– pv: this heuristic has one parameter which is the minimum number of proper variables in a cluster. A proper variable of a cluster is a variable of a

cluster which does not belong to the parent cluster. We merge a cluster with its parent if its number of proper variables is less than the value of the parameter.

All the heuristics we have defined, try to satisfy the first-fail principle, doing first the most constrained choices.

20.6. Experimental study

Applying a structural method on an instance generally assumes that this instance presents some particular topological features. So, our study is performed on instances having a structure which can be exploited by structural methods. In practice, we assess the proposed strategies on particular random WCSPs and real-world instances in order to point up the best ones w.r.t. the WCSP solving. Regarding the random instances, we exploit partial structured instances. A random structured instance of a class (n, d, w, t, s, n_c) is built according to the model described in [JÉG 03]. This structured instance consists of n variables having d values in their domain. Its constraint graph is a clique tree with n_c cliques whose size is at most w and whose separator size does not exceed s. Each constraint forbids t tuples. For each forbidden tuple, a weight between 1 and 10 is associated randomly. Then, for building a partial structured instance of a class (n, d, w, t, s, n_c, p), we remove randomly $p\%$ edges from a structured instance of a class (n, d, w, t, s, n_c). Secondly, we experiment the proposed heuristics on some real-world instances, namely radio-link frequency assignment problems from the FullRLFAP archive (for more details, see [CAB 99]).

All these experimentations are performed on a Linux-based PC with a Pentium IV 3.2GHz and 1GB of memory. For each considered random partial structured instance class, the presented results are the average over 30 solved instances. In the following tables, the letter M means that at least one instance cannot be solved because it requires more than 1GB of memory.

In [JÉG 05], a study was performed on triangulation algorithms to find out the best way to compute a good tree-decomposition w.r.t. the solving. As MCS [TAR 84] obtains the best results, we use it to compute tree-decompositions in this study.

In the following, the results for Class 5 are not presented since we cannot get good results. Table 20.1 shows the runtime of BTD based on FC with several heuristics of Classes 1, 2 and 3 on random partial structured instances. Clearly, we observe that the choice of the root cluster seems more important than the son ordering. Indeed, the obtained results appear to be similar as soon as we choose the root cluster with the same heuristic. Moreover, the main difference

Instance	Class 1	Class 2			Class 3		
	size	*bary*	*minexp*	*size*	*minexp*	*size*	*nv*
	$minsep_s$	$minsep_s$	$minexp_s$	$minsep_s$	$minexp_{sdyn}$	nv_{sdyn}	nv_{sdyn}
(75,10,15,30,5,8,10)	M	22.31	M	M	M	M	M
(75,10,15,30,5,8,20)	3.27	4.77	6.13	3.34	6.24	2.88	M
(75,10,15,33,3,8,10)	8.30	6.16	7.90	8.67	7.87	8.82	5.36
(75,10,15,34,3,8,20)	2.75	2.29	3.42	2.82	3.52	2.84	2.14
(75,10,10,40,3,10,10)	11.81	1.33	3.02	11.89	4.73	11.87	1.43
(75,10,10,42,3,10,20)	1.02	0.67	0.76	1.02	0.83	1.03	0.79
(75,15,10,102,3,10,10)	11.76	3.74	12.10	12.07	12.09	11.70	4.93
(100,5,15,13,5,10,10)	M	M	M	M	M	M	M

Table 20.1. *Runtime (in s) on random partial structured CSPs with mdd for class 1 and mdd_{dyn} for classes 2 and 3.*

Instance	Class 4				
	minexp	*size*	*minexp*	*size*	*nv*
	$minexp_s$	$minsep_s$	$minexp_{sdyn}$	nv_{sdyn}	nv_{sdyn}
(75,10,15,30,5,8,10)	9.42	18.99	8.69	18.30	16.77
(75,10,15,30,5,8,20)	1.65	1.67	1.56	1.53	2.32
(75,10,15,33,3,8,10)	5.22	4.31	5.26	4.18	3.24
(75,10,15,34,3,8,20)	1.50	1.61	1.48	1.58	1.40
(75,10,10,40,3,10,10)	0.58	0.81	0.58	0.85	0.52
(75,10,10,42,3,10,20)	0.41	0.42	0.51	0.41	0.38
(75,15,10,102,3,10,10)	5.50	4.73	5.41	4.63	3.13
(100,5,15,13,5,10,10)	9.40	9.05	9.60	9.10	11.71

Table 20.2. *Runtime (in s) on random partial structured CSPs with mdd_{dyn} for class 4 orders and sep heuristic (the separator size is bounded to 5).*

between the heuristics are observed for different choices of root cluster. The son cluster ordering has a limited effect because the considered instances have a few son clusters reducing the possible choices and so their impact. We can expect a more important improvement for instances with more son clusters. The heuristics *size* and *minexp* often provide interesting results but sometimes make bad choices. The heuristic *nv* leads to promising results except for one instance class which cannot be solved due to the required amount of memory. The heuristic *bary* seems the more robust heuristic: it obtains good results and succeeds in solving the instances of class $(75, 10, 15, 30, 5, 8, 10)$ while BTD with any other heuristics requires a too large amount of memory space. This memory problem can be solved by exploiting a *Class 4* order with the *sep* heuristic for grouping variables. Table 20.2 gives the runtime of BTD for this class with a separator size bounded to 5. When we analyze the value of the parameter k, we observe that in general, its value is limited (between 1 to 3). The results

Instance	Class 1	Class 2			Class 3		
	size	*bary*	*minexp*	*size*	*minexp*	*size*	*nv*
	$minsep_s$	$minsep_s$	$minexp_s$	$minsep_s$	$minexp_{sdyn}$	nv_{sdyn}	nv_{sdyn}
SUB_0	9.48	5.71	9.65	9.57	9.62	9.64	9.52
SUB_1	448	511	516	515	516	518	520
SUB_2	520	703	705	701	702	700	702
SUB_3	5,575	-	6,596	6,553	6,640	6,595	6,570
SUB_4	8,146	-	9,677	9,693	9,780	9,672	9,694

Table 20.3. *Runtime (in s) on some instances from the FullRLFAP archive with mdd for class 1 and* mdd_{dyn} *for classes 2 and 3.*

of Class 4 orders improve significantly ones obtained for the Classes 2 and 3. Like previously, the results are mostly influenced by the choice of the root cluster. The best results are obtained by $nv + nv_{sdyn}$, but $minexp + minexp_s$ and $minexp + minexp_{sdyn}$ have very close performances. For most of instance classes, the other heuristics obtain similar results. Unfortunately, for some other classes, some bad choices for the root cluster significantly increase the runtime.

Finally, in table 20.3, we assess the behaviour of the proposed heuristics on some real-world instances. Surprisingly, the considered Class 1 order obtains the best results. This result can be explained by the weak size of the considered instances (between 16 and 22 variables). It ensures that the cost of computing dynamically the variable ordering or the son ordering is not compensated. The same reason explains the close results obtained for Classes 2 and 3 orders. Whereas it obtains the more promising results on random instance, the heuristic *bary* requires more than 8 hours for solving the SUB_3 and SUB_4.

20.7. Discussion and Conclusion

In this article, we have studied the WCSP solving methods based on tree-decompositions in order to improve their practical interest. This study was done both theoretically and empirically. The analysis of the variable orders allows us to define more dynamic heuristics without loosing the time complexity bound. So, we have defined classes of variable orders which allow a more and more dynamic ordering of variables and preserve the theoretical time complexity bound. This bound has been extended to enforce the dynamic aspect of orders that has an important impact on the efficiency of enumerative methods. Even though these new bounds are theoretically less interesting that the initial, it allows us to define more efficient heuristics which improve significantly the runtime of BTD. This study, which could not be achieved previously, takes now an importance for solving hard instances with suitable structural properties.

We have compared the classes of variable orders with relevant heuristics w.r.t. WCSP solving. This comparison points up the promising results obtained by *Class 4* orders. These orders give more freedom to the variable order heuristic while their time complexity is $O(exp(w + k + 1))$ where k is a constant to parameterize. For the Class 5 (the most dynamic one), we get a time complexity in $O(exp(2(w + k + 1) - s^-))$.

The experimental study presented in this paper is a preliminary step. We only assess the interest of some heuristics. Other variable heuristics must be studied (e.g. the Jeroslow-like heuristic [GIV 03]). Likewise, for the choice of a root cluster or the son cluster ordering, we must propose heuristics well-adapted to the WCSP problem. For instance, the heuristic based on the expected number of solution must be extended by taking into account the weights associated to each tuple. Then, for *Class 4*, we aim to improve the criteria used to compute the value of k and to define more general ones by exploiting better the problem features. Finally, these experiments must be performed by exploiting BTD jointly with local consistency techniques [GIV 06].

Acknowledgments This work is supported by an ANR grant (STAL-DEC-OPT project).

20.8. References

[BIS 95] Bistarelli S., Montanari U., Rossi F., "Constraint solving over semirings", *Proceedings of IJCAI*, p. 624-630, 1995.

[CAB 99] Cabon C., de Givry S., Lobjois L., Schiex T., Warners J. P., "Radio Link Frequency Assignment", *Constraints*, vol. 4, p. 79-89, 1999.

[DEC 89] Dechter R., Pearl J., "Tree-Clustering for Constraint Networks", *Artificial Intelligence*, vol. 38, p. 353-366, 1989.

[DEC 99] Dechter R., "Bucket elimination: A unifying framework for reasoning", *Artificial Intelligence*, vol. 113, p. 41-85, 1999.

[DEC 01] Dechter R., Fattah Y. E., "Topological Parameters for Time-Space Tradeoff", *Artificial Intelligence*, vol. 125, p. 93-118, 2001.

[FRE 85] Freuder E., Quinn M., "Taking Advantage of Stable Sets of Variables in Constraint Satisfaction Problems", *Proceedings of IJCAI*, p. 1076-1078, 1985.

[FRE 92] Freuder E., Wallace R., "Partial constraint satisfaction", *Artificial Intelligence*, vol. 58, p. 21-70, 1992.

[GIV 03] de Givry S., Larrosa J., Meseguer P., Schiex T., "Solving Max-SAT as weighted CSP", *Proceedings of CP*, 2003.

[GIV 06] de Givry S., Schiex T., Verfaillie G., "Exploiting Tree Decomposition and Soft Local Consistency in Weighted CSP", *Proceedings of AAAI*, 2006.

[GOT 00] Gottlob G., Leone N., Scarcello F., "A Comparison of Structural CSP Decomposition Methods", *Artificial Intelligence*, vol. 124, p. 343-282, 2000.

[GOT 02] Gottlob G., Hutle M., Wotawa F., "Combining hypertree, bicomp and hinge decomposition", *Proceedings of ECAI*, p. 161-165, 2002.

[HUA 03] Huang J., Darwiche A., "A structure-based variable ordering heuristic for SAT", *Proceedings of IJCAI*, p. 1167-1172, 2003.

[JÉG 03] Jégou P., Terrioux C., "Hybrid backtracking bounded by tree-decomposition of constraint networks", *Artificial Intelligence*, vol. 146, p. 43-75, 2003.

[JÉG 04] Jégou P., Terrioux C., "Decomposition and good recording for solving Max-CSPs", *Proceedings of ECAI*, p. 196-200, 2004.

[JÉG 05] Jégou P., Ndiaye S. N., Terrioux C., "Computing and exploiting tree-decompositions for solving constraint networks", *Proceedings of CP*, p. 777-781, 2005.

[JÉG 06] Jégou P., Ndiaye S. N., Terrioux C., "An extension of complexity bounds and dynamic heuristics for tree-decompositions of CSP", *Proceedings of CP*, 2006.

[KOS 99] Koster A., Frequency Assignment - Models and Algorithms, PhD thesis, University of Maastricht, November 1999.

[LAR 02] Larrosa J., Meseguer P., Sánchez M., "Pseudo-Tree Search with Soft Constraints", *Proceedings of ECAI*, p. 131-135, 2002.

[LAR 03] Larrosa J., Dechter R., "Boosting Search with Variable Elimination in Constraint Optimization and Constraint Satisfaction Problems", *Constraints*, vol. 8, num. 3, p. 303-326, 2003.

[MAR 04] Marinescu R., Dechter R., "AND/OR tree search for constraint optimization", *Proceedings of CP workshop on Soft Constraints and Preferences*, 2004.

[MES 00] Meseguer P., Sánchez M., "Tree-based Russian Doll Search", *Proceedings of CP Workshop on soft constraint*, 2000.

[MES 01] Meseguer P., Sánchez M., "Specializing Russian Doll Search", *Proceedings of CP*, p. 464–478, 2001.

[MES 02] Meseguer P., Sánchez M., Verfaillie G., "Opportunistic Specialization in Russian Doll Search", *Proceedings of CP*, p. 264-279, 2002.

[ROB 86] Robertson N., Seymour P., "Graph minors II: Algorithmic aspects of tree-width", *Algorithms*, vol. 7, p. 309-322, 1986.

[SCH 95] Schiex T., Fargier H., Verfaillie G., "Valued Constraint Satisfaction Problems: hard and easy problems", *Proceedings of IJCAI*, p. 631-637, 1995.

[SMI 94] Smith B., "The Phase Transition and the Mushy Region in Constraint Satisfaction Problems", *Proceedings of ECAI*, p. 100-104, 1994.

[TAR 84] Tarjan R., Yannakakis M., "Simple linear-time algorithms to test chordality of graphs, test acyclicity of hypergraphs, and selectively reduce acyclic hypergraphs", *SIAM Journal on Computing*, vol. 13 (3), p. 566-579, 1984.

[TER 03] Terrioux C., Jégou P., "Bounded backtracking for the valued constraint satisfaction problems", *Proceedings of CP*, p. 709-723, 2003.

[VER 96] Verfaillie G., Lemaître M., Schiex T., "Russian Doll Search for Solving Constraint Optimization Problems", *Proceedings of AAAI*, p. 181-187, 1996.