On a generalization of triangulated graphs for domains decomposition of CSPs

Assef Chmeiss*
CRIL-CNRS FRE 2499
Université d'Artois - IUT de Lens
Rue de l'Université - SP 16
62307 Lens Cedex, France
chmeiss@cril.univ-artois.fr

Philippe Jégou and **Lamia Keddar**
LSIS - UMR CNRS 6168
Universite d'Aix-Marseille III
Avenue Escadrille Normandie-Niemen
13397 Marseille cedex 20, France
{philippe.jegou, lamia.keddar}@univ.u-3mrs.fr

Abstract

In [Jegou, 1993], a decomposition method has been introduced for improving search efficiency in the area of Constraint Satisfaction Problems. This method is based on properties of micro-structure of CSPs related to properties of triangulated graphs. This decomposition allows to transform an instance of CSP in a collection of sub-problems easier to solve, and then gives a natural and efficient way for a parallel implementation [Habbas *et al.*, 2000].

In this paper, we present a generalization of this approach, which is based on a generalization of triangulated graphs. This generalization allows to define the level of decomposition which can be fixed by a graph parameter. The larger this parameter is, the more level of decomposition that is the number of sub-problems is. As a consequence, we can then define the level of decomposition, with respect to the nature of the parallel configuration used (the number of processors).

First experiments reported here show that this extension increases significantly the advantage of the basic decomposition, already shown in [Habbas *et al.*, 2000].

1 Introduction

Constraint-satisfaction problems (CSPs) involve the assignment of values to variables which are subject to a set of constraints. Examples of CSPs are map coloring, conjunctive queries in a relational databases, line drawings understanding, pattern matching in production rules systems, combinatorial puzzles... CSP is known to be a NP-complete problem. So, during last twenty years, many works have been proposed to optimize the classical Backtrack procedure, as constraint propagation, intelligent backtracking or network decompositions.

In [Jegou, 1993], another approach has been proposed which is based on the decomposition of the "*micro-structure*" of the CSP. The micro-structure of a CSP is the graph defined by the compatible relations between variable-value pairs: vertices are these pairs, and edges are defined by pairs of compatible vertices. Given the micro-structure of a CSP and using graph properties, the method realizes a pre-processing to simplify the problem with a decomposition of the domains of variables. For a CSP $\mathcal{P}$, the decomposition generates a collection of subproblems $\mathcal{P}_1, \mathcal{P}_2, \ldots \mathcal{P}_k$, which is equivalent to the initial problem $\mathcal{P}$. Each subproblem $\mathcal{P}_i$ has the size of its domains less or equal to the size of domains of $\mathcal{P}$, so the time complexity of each $\mathcal{P}_i$ is necessarily less than for the initial CSP. Moreover, given this collection of subproblems, we can separately solve them, or recursively apply again the decomposition.

This decomposition allows to transform an instance of CSP in a collection of subproblems the domains size of which is less than domains size of the CSP. So, these sub-problems are theoretically easier to solve. Moreover, this decomposition defines and then gives a natural and efficient way for a parallel implementation to solve CSP. In [Habbas *et al.*, 2000], an experimental analysis of this approach is presented, realized on a parallel configuration. These experiments show clearly the efficiency of the approach: the decomposition generally offers better results than the results obtained to solving the problems without decomposition. This advantage is related to the natural parallelization of the approach since each subproblem is solved independently. This fact is clear for consistent CSPs. Moreover, if we consider inconsistent CSPs, the results are more insteresting: without parallelization, solving the problem after its decomposition is clearly better as solving it without decomposition.

In this paper, we present a generalization of this approach. This approach is based on a generalization of triangulated graphs. This generalization allows to define the level of decomposition which can be fixed by a graph parameter. The larger this parameter is, the more level of decomposition (that is the number of sub-problems) is. As a consequence, we can then define the level of decomposition, with respect to the power of the parallel configuration used.

To define this generalization, we exploit a generalization of triangulated graphs called *CSG^k graphs* which has been introduced in [Chmeiss and Jègou, 1997]. This class of graphs possesses the same kind of properties as triangulated graphs do. The most important here is the fact that the CLIQUE problem has a polynomial time complexity on these graphs.

*This work has been supported in part by the IUT de Lens, the CNRS and the Region Nord/Pas-de-Calais under the TACT Programme

First experiments reported here show that this extension increases significantly the advantage of the basic decomposition, already shown in [Habbas *et al.*, 2000]. Indeed, we present experiments with two levels of decomposition. The basic decomposition, and the second level (CSG^2 graphs). The results show that for the second level of decomposition we obtain better results.

The section 2 recalls the decomposition method while section 3 presents the generalization of the decomposition. Experimental results and a discussion about these result are presented in the section 4.

2 Preliminaries

2.1 Constraint Satisfaction Problems

A finite CSP (Constraint Satisfaction Problem) is defined as four tuple $\mathcal{P} = (X, D, C, R)$. X is a set of n variables $x_1, x_2, \ldots x_n$. D is a set of finite domains $D_1, D_2, \ldots D_n$, and C is a set of m constraints. Here we only consider binary CSPs, that is the constraints are defined on pairs of variables $\{x_i, x_j\}$ and will be denoted C_{ij}. To each constraint C_{ij}, we associate a subset of the cartesian product $(D_i \times D_j)$ which is denoted R_{ij} and specifies which values of the variables are compatible with each other. R is the set of all R_{ij}. A *solution* is an assignment of values to variables which satisfies all the constraints. For a CSP $\mathcal{P}$, the pair (X, C) is a graph called the *constraint graph*. Given a CSP, the problem is either to know if there exists any solution, to find solution, or to find all solutions. The first problem is known be to NP-complete.

CSPs are normally solved by different versions of backtrack search. In this case, if d is the size of domains (maximum number of values in domains D_i), the theoretical time complexity of search is then bounded by the size of the search space, that is d^n. Consequently, many works tried to improve the search efficiency. At present time, it seems that the most efficient algorithm is the procedure MAC which has been introduced in [Sabin and Freuder, 1994]. Other approaches use decomposition techniques based on structural properties of the CSP. These methods exploit the fact that the tractability of CSPs is intimately connected to the topological structure of their underlying constraint graph. Moreover, these methods give an upper bound to the complexity of the problem. As example of such decompositions, we have tree-clustering scheme [Dechter and Pearl, 1989]. The time complexity of this method is bounded by d^k where k is induced by the topological features of the constraint graph. Intuitively, more the constraint graph is dense, more the value k is large. For example, if the constraint network is a complete graph, then $k = n$. So, the complexity of these decomposition methods is the same as for classical backtracking, that is d^n. So, in [Jegou, 1993], Jégou proposed an alternative way to decompose CSPs which is not based on the constraint graph but on the *micro-structure* of a CSP. In the sequel, we called this decomposition scheme *TR-Decomposition*.

2.2 TR-Decomposition

Given a binary[1] CSP $\mathcal{P} = (X, D, C, R)$ such that (X, C) is a complete graph, $\mu(\mathcal{P}) = (X_D, C_R)$ is called *micro-structure* of $\mathcal{P}$ and it is a n-partite graph where $X_D = \{(x_i, a) : x_i \in X \text{ and } a \in D_i\}$ and $C_R = \{\{(x_i, a), (x_j, b)\} : (a, b) \in R_{ij}\}$.

Note that if (X, C) is not a complete graph, for pairs of variables x_i and x_j such that the constraint C_{ij} does not belong to set C, we consider the universal relation between these variables, that is $R_{ij} = D_i \times D_j$ (all pairs of values are compatible). The decomposition is based on a basic property of the micro-structure:

Theorem 1 Given a CSP $\mathcal{P}$ and its micro-structure $\mu(\mathcal{P})$, then $(a_1, \ldots a_n) \in Solutions(\mathcal{P})$, that is $(a_1, \ldots a_n)$ is a solution of $\mathcal{P}$, iff $\{(x_1, a_1), \ldots (x_n, a_n)\}$ is a n-clique of $\mu(\mathcal{P})$.

In other words, transforming a CSP as its micro-structure is clearly a polynomial reduction from CSP to *CLIQUE*, which is the problem of finding a clique of a given size belongs to a graph. As a consequence of this fact, as CSP, the CLIQUE problem is NP-Complete. Since the problem of finding a n-clique is NP-hard, TR-Decomposition exploits the fact that *triangulated graphs* constitute a polynomial class of instances for this problem. We recall briefly basic properties of these graphs. Given a graph $G = (V, E)$ and an ordering $\sigma = [v_1, ..., v_n]$ of V, the *successors* of a vertex v_i are the elements of the set $N^+(v_i) = \{v_j \in N(v_i) : i < j\}$ where $N(v) = \{u \in V : \{u, v\} \in E\}$; a vertex v_i is *simplicial in* σ if $N^+(v_i)$ is a clique. The ordering $\sigma = [v_1, ..., v_n]$ of V is a *perfect elimination ordering* if, for $i = 1, 2, ...n$, the vertex v_i is simplicial in the ordering σ. A graph $G = (V, E)$ is triangulated if and only if V admits a perfect elimination ordering. Triangulated graphs satisfies desirable properties. A triangulated graph on n vertices has at most n maximal cliques (a clique is maximal iff it is not included in an other clique) [Fulkerson and Gross, 1965]. Moreover, the problem of finding all maximal cliques in a triangulated graph is in $O(n + m)$ if n is the number of vertices and m the number of egdes [Gavril, 1972].

Given the micro-structure of any CSP, it is not possible to immediately use these properties because any micro-structure is not necessary a triangulated graph. So, in adding edges in E, we can build a new graph $\Gamma(G) = (V, E')$ which is triangulated. This addition of edges is called *triangulation*, and can be realized in a linear time in the size of the graph. So, after the triangulation, all maximal cliques of G can be found in linear time. Applied to a CSP $\mathcal{P} = (X, D, C, R)$, the triangulation of its micro-structure $\mu(\mathcal{P}) = (X_D, C_R)$ produces the graph $\Gamma(\mu(\mathcal{P}))$, the set of maximal cliques of which is $Y_1, \ldots Y_k$ ($k \leq d.n$). Every Y_i is a set of values, and then defines a subproblem of $\mathcal{P}$. More formaly, given a binary CSP $\mathcal{P} = (X, D, C, R)$, its micro-structure $\mu(\mathcal{P}) = (X_D, C_R)$, and Y a subset of X_D. The CSP induced by Y on

[1] Note that in [Hamadi, 1996], TR-Decomposition has been generalized to General CSPs, that is CSPs were constraints can be defined on more than two variables.

$\mathcal{P}$, denoted $\mathcal{P}(Y)$ is defined by:

- $D_Y = \{D_{Y,1}, \ldots D_{Y,n}\} : D_{Y,i} = \{a \in D_i : (x_i, a) \in Y\}$
- $R_{Y,ij} = \{(a,b) \in R_{ij} : (x_i, a), (x_j, b) \in Y\}$
- $\mathcal{P}(Y) = (X, D_Y, C, R_Y)$

Finally, to ensure the validity of the decomposition, presents the theorem below:

Theorem 2 [Jegou, 1993] Given a binary CSP $\mathcal{P}$, its micro-structure $\mu(\mathcal{P})$ and $\{Y_1, \ldots Y_k\}$, the set of the maximal cliques of $\Gamma(\mu(\mathcal{P}))$, then

$$Solutions(\mathcal{P}) = \cup_{i=1}^{k} Solutions(\mathcal{P}(Y_i)).$$

To summarize the approach, we recall the algorithm:

Algorithm: TR-Decomposition
Begin
1 Build the micro-structure $\mu(\mathcal{P})$
2 Triangulation($\mu(\mathcal{P})$,$\Gamma(\mu(\mathcal{P}))$,σ)
3 MaximalCliques($\Gamma(\mu(\mathcal{P}))$,σ,$\{Y_1, \ldots Y_k\}$)
4 **for** all Y_i **do**
5 **if** Y_i is a covering of all the domains in D
6 **then** Solve($\mathcal{P}(Y_i)$)
7 **else** $\mathcal{P}(Y_i)$ has no solution
endfor
End; { TR-Triangulation }

The steps 1, 2 and 3 can be realized in a linear time w.r.t. the size of the problem $\mathcal{P}$. The procedure *Triangulation* returns a perfect elimination ordering σ which allows to the procedure *MaximalCliques* to find sets Y_i. The step 4, that is the iteration **for**, can be replaced by a parallel execution on independent subproblems $\mathcal{P}(Y_i)$. In this case, if the CSP $\mathcal{P}$ is consistent, the cost is related to the cost of solving the easiest subproblem, while it is the cost of the hardest subproblem for inconsistent CSPs. A such implementation has been realized in [Habbas *et al.*, 2000]. In step 5, a set Y_i is not a covering of all the domains in D if there is a subdomain $D_{Y_i,j}$ induced by Y_i which is the empty set. This subproblem is then trivially inconsistent. In step 6, to solve $\mathcal{P}(Y_i)$, we can use any classical search method such that standard backtracking, Forward-Checking or MAC [Sabin and Freuder, 1994]. Finally, note that the quality of decomposition depends on the quality of the triangulation step.

The last point is the most important. Nevertheless, it's well known that the problem of finding an optimal (w.r.t. the size of the largest induced clique) triangulation is NP-Hard (see [Arnborg *et al.*, 1987]). Several algorithms have been proposed for triangulation. In all cases, the aim is to minimize either the number of added edges or the size of cliques in G'. Three classes of approaches are usable. The first one consists in finding an optimal triangulation which produces a graph the maximum size clique of which has minimum size over all triangulations. [Shoikhet and Geiger, 1997] propose a such algorithm the time complexity of which is exponential. The second is to find a minimal triangulation, that is a triangulation computing a set E_1 such as if for any other triangulation $E_2 \subset E_1$ the graph $G' = (V, E \cup E_2)$ is not triangulated. Note that a triangulation can be minimal and not optimal. See [Berry, 1999] who presents an algorithm in $O(n(n+m))$. The last one consist in using a triangulation consuming a linear time which does not gurantee the minimality. Nevertheless, the purpose of this paper is not to propose a good triangulation, but to exploit a generalization of triangulated graphs to generalize this decomposition. One of the aims of this generalization is to avoid the problem of finding optimal triangulations.

3 Generalizing TR-Decomposition

3.1 A first basic property

To generalize TR-Decomposition, we can extend theorem 2. Consider a CSP $\mathcal{P} = (X, D, C, R)$ and its micro-structure $\mu(\mathcal{P}) = (X_D, C_R)$. Consider a graph $\Delta(\mu(\mathcal{P})) = (X_D, \delta(C_R))$ where $C_R \subseteq \delta(C_R)$. Consider now the set $\{Z_1, \ldots Z_j\}$ of maximal cliques of $\Delta(\mu(\mathcal{P}))$. Every Z_i is a set of values, and then, as in section 2.2, defines a subproblem of $\mathcal{P}$ which is the CSP induced by Z_i on $\mathcal{P}$ (it is denoted $\mathcal{P}(Z_i)$).

Theorem 3 Given a binary CSP $\mathcal{P}$, its micro-structure $\mu(\mathcal{P})$ and $\{Z_1, \ldots Z_j\}$, the set of the maximal cliques of $\Delta(\mu(\mathcal{P}))$, then $Solutions(\mathcal{P}) = \cup_{i=1}^{j} Solutions(\mathcal{P}(Z_i))$.

Proof. Given a binary CSP $\mathcal{P}$, its micro-structure $\mu(\mathcal{P})$ and by theorem 1, we know that each solution of $\mathcal{P}$ is a clique of size n. If we consider $\Delta(\mu(\mathcal{P})) = (X_D, \delta(C_R))$ where $C_R \subseteq \delta(C_R)$, each clique of $\mu(\mathcal{P})$ is included in a maximal clique Z_i of $\Delta(\mu(\mathcal{P}))$. Therefore, each solution of $\mathcal{P}$ appears in the associated subproblem $\mathcal{P}(Z_i)$.

While TR-Decomposition is limited to triangulated graphs, theorem 3 allows to define a larger class of decomposition of domains. Nevertheless, because the number of maximal cliques can be exponential in an arbitrary graph, and then, the number of induced subproblems, we must consider graphs $\Delta(\mu(\mathcal{P}))$ which possess a limited number of maximal cliques as triangulated graphs do. So, we exploit here a class of graphs which is a generalization of triangulated graphs.

3.2 A generalization of triangulated graphs

In [Chmeiss and Jègou, 1997], a new class of graphs, called *CSG^k graphs* has been introduced. These graphs which generalize triangulated graphs, possess the same kind of properties as triangulated graphs, e.g. hereditary property of subgraphs, existence of an elimination scheme, polynomial time recognizing algorithm, and polynomial class of graph for CLIQUE problem. Informally, a CSG^0 graph is a complete graph, a CSG^1 graph is a triangulated graph, and a CSG^2 graph is a graph allowing an elimination vertex scheme where the successors of every vertex induce triangulated graphs. More generally, CSG^k graphs are defined inductively.

Definition [Chmeiss and Jègou, 1997] The class of CSG^0 Graphs is the class of complete graphs. Given $k > 0$, the class of CSG^k graphs is the class of graphs $G = (V, E)$ such that there exists an ordering $\sigma = [v_1, ..., v_n]$ of V such that

for $i = 1, 2, ...n$, the graph $G(N^+(v_i))$ is a CSG^{k-1} Graph. The ordering σ is then called a CSG^k scheme.

It's easy to see that CSG^1 Graphs is the class of triangulated graphs. As for triangulated graphs, there exists a polynomial time algorithm R-CSG^k for recognizing CSG^k graphs; $\forall k \geq 2$, its time complexity is $O((n^2)^{k-1}(n+m))$. In the context of decomposition of micro-structure, the number of maximal cliques is an important criteria. CSG^k graphs inherit of a generalization of the property of Fulkerson and Gross [Fulkerson and Gross, 1965] related to this number since, a CSG^k Graphs with n vertices have less than n^k maximal cliques. Moreover, a CSG^2 graph with n vertices and m edges has at most $n + m$ maximal cliques. Finding these cliques is easy, since, given a constant value k the CLIQUE problem is polynomial on this class of graphs since the time complexity of finding the clique of maximum size in an undirected CSG^k graph is bounded by $O(n^{2(k-1)}(n+m))$.

3.3 TR^k-Decomposition

CSG^k graphs can be exploited as a way to generalizing of TR-Decomposition, defining TR^k-Decomposition as decompositions corresponding to the class of associated graphs. For $k = 1$, TR^k-Decomposition is exactly TR-Decomposition, while for $k = 2$, TR^k-Decomposition, that is TR^2-Decomposition, it's the decomposition induced by CSG^2 graphs. More generally, TR^k-Decomposition is summarized as:

Algorithm: TR^k-Decomposition
Begin
1 Build the micro-structure $\mu(\mathcal{P})$
2 k-Triangulation($\mu(\mathcal{P})$,$\Delta^k(\mu(\mathcal{P}))$,$\sigma$)
3 k-MaximalCliques($\Delta^k(\mu(\mathcal{P}))$,$\sigma$,$\{Z_1, \ldots Z_j\}$)
4 **for** all Z_i **do**
5 **if** Z_i is a covering of all the domains in D
6 **then** Solve($\mathcal{P}(Z_i)$)
7 **else** $\mathcal{P}(Z_i)$ has no solution
endfor
End;

Note that :

- The procedure *k-Triangulation* produces a graph $\Delta^k(\mu(\mathcal{P}))$ which is a CSG^k graph and returns an ordering σ which allows to the procedure *k-MaximalCliques* to find all the maximal cliques Z_i of $\Delta^k(\mu(\mathcal{P}))$.
- As for TR-Decomposition the step 4, that is the iteration **for**, can be replaced by a parallel execution on independent subproblems $\mathcal{P}(Z_i)$. In this case, if the CSP $\mathcal{P}$ is consistent, the cost is related to the cost of solving the easiest subproblem, while it is the cost of the hardest subproblem for inconsistent CSPs.
- For steps 5 and 6, remarks given for TR-Decomposition hold.
- Finally, the quality of decomposition depends on the quality of the k-triangulation step.

Because of the time complexity of managing CSG^k graphs, that is $O(n^{2(k-1)}(n+m))$, it seems reasonable to focus our attention on small values of k. So, in the sequel we limit our study to CSG^2 graphs. Therefore, the number of maximal cliques will be bounded now by $k \leq d.n + d^2.n^2$. Nevertheless, two problems must be adressed here. Finding an efficient algorithm for 2-triangulation, and an operational algorithm to find maximal cliques of a CSG^2 graph.

For 2-triangulation, we can imagine an algorithm the time complexity of which is theoreticaly bounded by $O(nm(n+m))$. This algorithm consider an ordering on vertices which is obtained by the Maximum Cardinality Search of Tarjan and Yannakakis. Given this ordering, we consider vertices v_i, and then, we run a triangulation on $G(N^+(v_i))$. This first step is in $O(n(n+m))$ but this approach does not guarantee to get a 2-triangulation. Indeed, given two vertices v_i and v_j, with $i < j$, the triangulation of $G(N^+(v_j))$ which is realized after the triangulation of $G(N^+(v_i))$ can add edges that restore the new subgraph $G(N^+(v_i))$ not triangulated. So, a second step of verification and repairing the subgraphs $G(N^+(v_i))$ in adding new edges must be realized. This additionnal work induces a theoretical complexity of $O(nm(n+m))$. Nevertheless, expermients show that this additionnal work will be rarely reached in practical cases (see section 4).

4 Experiments

Our aim in this section is to show the usefulness of the generalized decomposition method comparing to the MAC (Maintaining Arc Consistency) algorithm and the decomposition method by triangulation of the micro-structure and to give an idea about the efficiency of the proposed method. For the generalized decomposition method we will consider the TR^2-Decomposition, i.e. $k = 2$.

As mentionned in section 3, we will not focus on the triangulation and 2-triangulation algorithms because it is not the principal objective of this paper. In experiments, we will use the Tarjan and Yannakakis's triangulation algorithm [Tarjan *et al.*, 1984]. This algorithm requires a precomputed elimination ordering on the vertices. Then, it adds edges, with respect to the ordering, which maintain the chordality property (i.e. when a vertex is treated, its successors must form a clique). Also, for the 2-triangulation, we will use an algorithm based on the Tarjan and Yannakakis's triangulation one as described in section 3.3. In practice, we have observed that the additionnal step is rarely realized and then the observed time of the 2-triagnluation is in $n(n+m)$. So, for CSPs of great size, the CPU time used to decompose the problem is generally not significant with respect to the time of solving the CSP.

We have expermiented these methods on random CSPs generated at the phase transition. These CSPs are generated according to the model B generator [Prosser, 1996]. Such a generator admits four parameters:

- the number N of variables
- the common size of the initial domains D
- the proportion $p1$ of constraints in the network (or the number $C = p1 * N \cdot (N-1)/2$ of constraints)

$< N, D, T, C >$	#SubProblems			#checks			#nodes		
	TR	TR^2		MAC	TR	TR^2	MAC	TR	TR^2
$< 20, 5, 6, 106 >$	4	14	Sat (29/50)	7943	5260	2695	23	22	20
			Unsat (21/50)	15219	11888	5355	15	12	6
$< 20, 5, 11, 47 >$	3	12	Sat (31/50)	2295	1710	1119	21	20	20
			Unsat (19/50)	2892	2189	1500	2	1	1
$< 20, 10, 30, 130 >$	5	40	Sat (10/50)	101141	72138	21279	57	46	26
			Unsat (40/50)	322397	262212	63768	133	111	27
$< 20, 10, 50, 60 >$	6	42	Sat (39/50)	20816	10227	5372	28	22	20
			Unsat (11/50)	48294	35760	16044	18	15	6
$< 20, 20, 130, 150 >$	7	109	Sat (31/50)	8598276	2312841	377929	2641	720	130
			Unsat (19/50)	19178630	16179016	1672898	5825	4958	517
$< 20, 20, 201, 85 >$	11	131	Sat (17/50)	768460	236799	48223	193	70	27
			Unsat (33/50)	1611125	1187395	252259	339	253	50
$< 30, 5, 5, 190 >$	3	15	Sat (39/50)	21672	14142	6954	48	39	32
			Unsat (11/50)	67897	60267	19846	72	86	23
$< 30, 5, 10, 80 >$	3	13	Sat (32/50)	4778	3663	2243	32	31	30
			Unsat (18/50)	6997	6145	4876	6	6	7
$< 30, 10, 30, 190 >$	5	37	Sat (14/50)	875070	429786	176796	410	214	101
			Unsat (36/50)	1774848	1472370	422599	742	619	178
$< 30, 10, 50, 90 >$	5	39	Sat (37/50)	51855	32627	11648	49	42	32
			Unsat (13/50)	116232	91314	54609	50	40	20
$< 30, 15, 90, 155 >$	6	78	Sat (18/50)	2549883	1177789	497667	769	364	158
			Unsat (32/50)	6717878	5442121	1778683	1846	1513	465
$< 30, 15, 115, 110 >$	7	87	Sat (16/50)	1130136	285589	66599	323	101	42
			Unsat (34/50)	1517131	1314240	408847	371	329	95
$< 40, 5, 6, 210 >$	3	14	Sat (19/50)	37505	23780	11327	76	60	47
			Unsat (31/50)	62690	51937	26762	58	50	24
$< 40, 5, 4, 341 >$	3	14	Sat (21/50)	116682	85003	42475	138	111	70
			Unsat (29/50)	291075	272612	119047	244	230	101
$< 40, 10, 55, 105 >$	5	40	Sat (22/50)	75832	44089	18412	70	54	42
			Unsat (28/50)	142332	126415	74768	48	48	23
$< 40, 10, 35, 205 >$	5	43	Sat (16/50)	1566087	965645	233136	758	491	137
			Unsat (34/50)	3467369	2810709	1194567	1468	1202	477

Table 1: experimental results

- the proportion $p2$ of forbidden pairs of values in a constraint (or the number $T = p2 * D * D$ of forbidden pairs)

In the following, we will use C and T instead of $p1$ and $p2$. C is called the *density* of the constraint graph and T the *tightness* of the constraints. We have analysed decomposition using the procedure MAC as basic search algorithm (the used heuristic is dom/deg [Bessière and Régin, 1996]). That is MAC has been used before decomposition on the initial problem and after decomposition on each induced subproblem.

In order to have a credible idea on the performance of these methods, we consider several classes of problems. We vary two parameters (the number of variables N and the domains size D). In order to deal with classes near the phase transition, we search the appropriate values of the *tightness* (T) and the *density* (C).

For each value of N, we vary D from small domains size to larger ones. So, we can observe the behavior of the decomposition when D grows. Also, we have remarked that the performance of the decomposition method was not the same for satisfiable and unsatisfiable problems. So, we have chosen to present the results seperately.

Results are reported in three tables. Each table represents results for a given number of variables N. As measures of comparison, we used the number of consistency checks (#checks) performed by each of the methods and the number of visited nodes (#nodes). We mention that, the execution time is proportional to the number of checks.

In the tables, results are organized as following : the first column contains the calsses of problems represented by the four parameters $< N, D, T, C >$. For each class, we give results of the differents methods for satisfiable (Sat) and unsatisfiable (Unsat) problems seperately. So, we have two lines per class. The second and the third columns give the number of subproblems induced by the TR-Decomposition and the TR^2-Decomposition respectively. The fourth column indicates the number of satisfiable and unsatisfiable problems over the 50 randomly generated CSPs. The next three columns give the number of consistency checks for MAC, TR-Decomposition and the TR^2-Decomposition respectively. The last three columns present the number of visited nodes by the different methods.

As the subproblems induced by the decomposition are independent, we can envisage a parallel implementation. In other words, we can imagine one process by subproblem. So, for saisfiable problems we consider the faster subproblem (the firstly solved subproblem since we have the response the it is satisfiable) while we consider the hardest subproblem for unsatisfiable ones.

Tables show, that decomposition improve the MAC algorithm on all classes. The improvement is more significant for larger domains. Also, the performance of MAC is improved more significately for denser constraint graphs. If we look at table 1 where $N = 20$, we remark that the TR^2-Decomposition is five times faster than MAC for the $< 20, 10, 30, 130 >$. For classes with $D = 20$ the improvement is more significant. This remark is confirmed when

the constraint graphe is denser. For example, for the class $< 20, 20, 130, 150 >$ and for satisfiable problems, the performance ratio between TR^2-Decomposition and MAC is about 1 to 22. In other words, for each consistency check performed by TR^2-Decomposition l'algorithm MAC performs 22 checks for satisfiable problems. For unsatisfiable problems, the performance ratio is about 1 to 12.

This overall view on experiments shows that the generalized decomposition method is promising and results are encouraging. Finally, we would like to mention the importance of the triangulation (2-triangulation) phase. We think that the efficieny of the decomposition method is related to the quality of the triangulation (2-triangulation). We believe that other triangulation algorithms might offer a better decomposition.

5 Conclusion

In this paper, we have presented a generalization of the decomposition of micro-structure, introduced in [Jegou, 1993]. This decomposition allows to transform an instance of CSP in a collection of sub-problems easier to solve, and then gives a natural and efficient way for a parallel implementation [Habbas *et al.*, 2000]. While the original method is based on triangulated graphs, our generalization is based on a generalization of triangulated graphs called CSG^k graphs. For a given value k, we get a special sub-class of graphs. E.g. CSG^0 graphs are complete graphs and CSG^1 graphs are triangulated graphs. The value k allows to define a particular level of decomposition. So, we have introduced *TR^k-Decomposition* as the generalized decomposition based on CSG^k graphs.

There are two motivations for this generalization of the decomposition:

- to extend the level of decomposition,
- to be able to fix this level to give an optimal exploitation for the parallel configuration used.

Indeed, it has been experimentally observed in [Habbas *et al.*, 2000] that the first level of decomposition, that is TR^1-Decomposition, outperforms classical algorithms, and that the more the number of sub-problems is, the more the efficiency of TR^1-Decomposition is.

Our experimental results confirm these facts. We have studied two levels of decomposition, for $k = 1$ and $k = 2$ and the experiments show that TR^2-Decomposition outperforms TR^1-Decomposition. Moreover, the number of sub-problems is greater for $k = 2$ than for $k = 1$.

We have limited our experiments to $k = 2$ because the time complexity of managing CSG^k graphs is related to the value k. More precisely, $O(n^{2(k-1)}(n+m))$ is the time complexity to recognize these graphs. So in future works, we will try to propose efficient algorithms to realize decomposition.

References

[Arnborg *et al.*, 1987] S. Arnborg, D.G. Corneil and A. Proskurowski. Complexity of finding embeddings in a k-tree. *SIAM J. Disc. Meth.*, 8:277–284, 1987.

[Bessière and Régin, 1996] C. Bessière and J-C. Régin. MAC and combined heuristics: Two reasons to forsake FC (and CBJ?) on hard problems. In *Proceedings of CP'96, LNCS 1118*, pages 61–75, Cambridge, MA, USA, 1996.

[Berry, 1999] A. Berry. A Wide-range Efficient Algorithm for Minimal Triangulation. In *Proceedings of SODA'99 SIAM Conference*, USA, 1999.

[Chmeiss and Jègou, 1997] A. Chmeiss and P. Jègou. A Generalization of Chordal Graphs and the Maximum Clique Problem. *Information Processing Letters*, 62(2):61–66, 1997.

[Dechter and Pearl, 1989] R. Dechter and J. Pearl. Tree Clustering for Constraint Networks. *Artificial Intelligence*, 38:353–366, 1989.

[Fulkerson and Gross, 1965] D.R. Fulkerson and O. Gross. Incidence matrices and interval graphs. *Pacific J. Math.*, 15:835–855, 1965.

[Gavril, 1972] F. Gavril. Algorithms for minimum coloring, maximum clique, minimum covering by cliques, and maximum independent set of a chordal graph. *SIAM J. Comput.*, 1(2):180–187, 1972.

[Habbas *et al.*, 2000] Z. Habbas, D. Singer and R. Krajecki. Domain Decomposition for Parallel Resolution of Constraints Satisfaction Problems with OpenMP. In *Proceedings of the Second European Workshop on OpenMP*, Edinburgh, Scotland, UK, 2000.

[Hamadi, 1996] R. Hamadi. A domain decomposition method to solve n-ary CSPs. In *Proceedings of the International Workshop on Constraint-Based Reasonning, FLAIRS-Constraints-96*, pages 84–88, Key West Florida, USA, 1996.

[Jegou, 1993] P. Jégou. Decomposition of domains based on the structure of finite constraints satisfaction problems. In *Proceedings of the National Conference on Artificial Intelligence - AAAI'93*, pages 731–736, Washington DC, USA, July 1993.

[Prosser, 1996] P. Prosser. An empirical study of phase transition in binary constraint satisfaction problems. *Artificial Intelligence*, 81:81–109, 1996.

[Sabin and Freuder, 1994] D. Sabin and E. Freuder. Contradicting conventional wisdom in constraint satisfaction. In *Proceedings of ECAI'94*, pages 125–129, Amsterdam, Holland, 1994.

[Shoikhet and Geiger, 1997] K. Shoikhet and D. Geiger. A practical algorithm for finding optimal triangulation. In *Proceedings of AAAI'97*, pages 185–190, Providence, USA, 1997.

[Tarjan *et al.*, 1984] R. Tarjan and M. Yannakakis. Simple Linear-Time Algorithms to test Chordality of graphs, test acyclicity of hypergraph and selectively reduce acycli hypergraph. *SIAM Journal on Computing*, 13(3):566–579, 1984.