A Disjunctive Decomposition Scheme for Discrete Constraint Satisfaction Problems Using Complete No-Good Sets

Berthe Y. CHOUEIRY and Guevara NOUBIR
Industrial Computing Laboratory
Department of Computer Science
Swiss Federal Institute of Technology in Lausanne (EPFL)
{choueiry|noubir}@di.epfl.ch

Abstract

In this paper, we introduce a new disjunctive decomposition scheme for discrete constraint satisfaction problems (CSPs). This strategy is based on first identifying complete no-goods in a graph derived from the microstructure of the CSP, then using these no-goods to decompose the initial CSP into subproblems that exclude these no-goods. This decomposition produces a partition of the solution space and is guaranteed to keep all solutions while reducing the total number of possibilities to be considered. We describe the strategy, study its properties, and identify the number of possibilities that are excluded at any decomposition step. We describe a practical application to which this strategy can be efficiently applied and compare our technique to some decomposition methods reported in the literature.

1 Introduction

Many important applications, such as scheduling, resource allocation, vision, and query processing in data bases, can be modeled as a discrete Constraint Satisfaction Problem (CSP). The strategies for solving this problem are mainly variations around backtrack search. Backtrack search can be understood as an iterative decomposition of the initial problem space into a disjunction of subproblems obtained by partitioning the domain of one variable at a time. One advantage of disjunctive decomposition strategies is that they can be efficiently implemented in parallel: the resulting subproblems need not communicate and thus can be distributed among a number of processing units.

In [4], Freuder and Hubbe describe a general control schema for disjunctive decomposition of a CSP and show how previous resolution strategies (e.g., backtrack, forward checking, network consistency) can be formulated as instances of this schema. In [9], Jégou introduces yet another decomposition scheme based on using the characteristics of a graph structure called the microstructure of the CSP.

In this paper, after introducing some basic definitions, we introduce a new decomposition scheme also based on exploiting the microstructure but in a different way than proposed by Jégou. More specifically, we define the complementary microstructure of a CSP, which is a subgraph of the complement of its microstructure, and identify sets of complete no-goods as cliques of this subgraph. We then use these complete no-goods to induce subproblems of the original CSP that contain none of these no-goods. In this manner, we can eliminate very early on, during decomposition, subproblems that only contain non-consistent combinations (i.e., no solutions). We discuss the properties of the resulting decomposition and give an exact evaluation of the number of possibilities excluded. Then we identify the problem of resource allocation as one practical application to which our strategy can be efficiently applied. Finally, we compare our approach to the ones described in the literature, such as ordered search, IDC (Inferred Disjunctive Constraint [3]), FOF (Factor Out Failure [5]), VAD (Value Assignment Delay Heuristic [2]), and the strategy proposed by Jégou [9].

2 Definitions

A *constraint satisfaction problem* (CSP) is defined by $\mathcal{P} = (\mathcal{V}, \mathcal{D}, \mathcal{C})$; where $\mathcal{V} = \{V_1, V_2, \ldots, V_n\}$ is the set of variables, $\mathcal{D} = \{D_{V_1}, D_{V_2}, \ldots, D_{V_n}\}$ the set of values (or domains) associated with the variables, and $\mathcal{C}$ is the set of constraints that apply to the variables. A constraint C_{V_i,V_j} applicable to two variables V_i and V_j restricts the combination of values that the variables can be assigned at the same time and thus defines a relation $R_{V_i,V_j} \subseteq D_{V_i} \times D_{V_j}$, which is the set of tuples allowed by C_{V_i,V_j}. When the relation is exactly the Cartesian product of the variable domains (i.e., $R_{V_i,V_j} = D_{V_i} \times D_{V_j}$), the corresponding constraint is said to be *universal*. To solve a CSP is to assign one value to each variable such that all constraints are simultaneously satisfied. The *size* of a CSP is the number of all possible combinations and is equal to $\prod_{V_i \in \mathcal{V}} |D_{V_i}|$. A CSP is commonly represented by a *constraint graph* in which the variables are represented by nodes, the domains by node labels, and the constraints by edges that link the relevant nodes. Universal constraints are omitted from the constraint graph.

In this document, we restrict our study to discrete binary CSPs: each domain D_{V_i} is a finite set of discrete values and each constraint applies to two variables.

The *microstructure* [9] of a CSP $\mathcal{P} = (\mathcal{V}, \mathcal{D}, \mathcal{C})$, denoted $\mu(\mathcal{P})$, is the graph $G(V, E)$ where $V = \{(V_i, v) \,|\, (V_i \in \mathcal{V}) \wedge (v \in D_{V_i})\}$ and $E = \{e_{(V_i,v),(V_j,w)} \,|\, (v \in D_{V_i}) \wedge (w \in D_{V_j}) \wedge (i \neq j) \wedge ((v, w) \in R_{V_i,V_j})\}$. Informally stated, $\mu(\mathcal{P})$ is the graph whose vertices are variable-value pairs for all variables and whose edges link variable-value pairs of distinct variables that are consistent with respect to the constraint applicable to the variables (allowed tuple).

We define the *complementary microstructure* of a CSP $\mathcal{P} = (\mathcal{V}, \mathcal{D}, \mathcal{C})$, denoted co-$\mu(\mathcal{P})$, to be the graph $G(V, E)$ where $V = \{(V_i, v) \,|\, (V_i \in \mathcal{V}) \wedge (v \in D_{V_i})\}$ and $E = \{e_{(V_i,v),(V_j,w)} \,|\, (v \in D_{V_i}) \wedge (w \in D_{V_j}) \wedge (i \neq j) \wedge ((v, w) \notin R_{V_i,V_j})\}$.

Informally stated, it is the graph of non compatible tuples. Obviously, co-$\mu(\mathcal{P}) = \overline{\mu(\mathcal{P})} - \{$edges among pairs corresponding to the same variable$\}$, where $\overline{G}$ denotes the complement of graph G. Figure 1 shows a simple example of a CSP and its associated microstructure and co-microstructure.

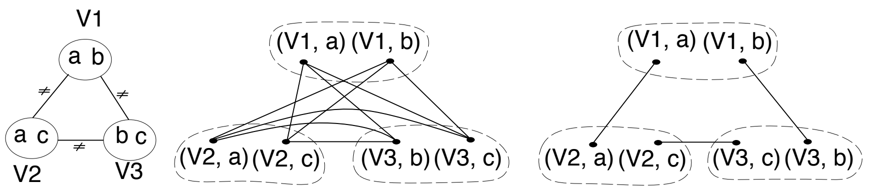

Figure 1: A simple example $\mathcal{P}$, $\mu(\mathcal{P})$, and co-$\mu(\mathcal{P})$.

A *no-good* is a set of variable-value pairs that are not consistent. Note that it is sufficient that any two pairs in this set be not compatible for the whole set to be a no-good. A *minimal no-good* is a no-good set of which no subset is a no-good. Note that for binary CSPs a minimal no-good set has exactly two elements. We define a *complete no-good* (CNG) set to be a no-good set of which any non-singleton subset is a no-good set. Note that it is not straightforward to define what is a *maximal* CNG-set since applying consistency checking techniques to the CSP may uncover new inconsistencies and thus increase the size of a given CNG-set.

A *conjunctive decomposition* [4] "breaks the problem into subproblems such that all the subproblems must be solved, and the solutions must fit together properly, for the original problem to be solved".

A *disjunctive decomposition* [4] of a CSP $\mathcal{P}$ is a set of subproblems $\{\mathcal{P}_i\}$, each of which is induced from $\mathcal{P}$ by restricting the domain of one or more variables (and accordingly updating the constraints). The decomposition is required to satisfy the following properties [4]:

- *Consistency.* Any pair of values from a pair of instantiated variables is consistent.
- *Simplification.* Each of the $\{\mathcal{P}_i\}$ has fewer possibilities or more instantiated variables than $\mathcal{P}$.
- *Semi-completeness.* If there is a solution to $\mathcal{P}$, then there will be a solution to at least one of the $\{\mathcal{P}_i\}$.

3 Decomposition

Jégou introduced the microstructure and showed how it can be exploited to decompose a CSP (see Section 5.3). The microstructure elicits the variable-value pairs that are allowed by the constraints. In our approach, we are motivated by the fact that the 'root' of the problem is the existence of non-compatible variable-value pairs, which creates an interaction among the variables that may lead to inconsistencies. Indeed, if no variable-value pairs in the co-microstructure were connected, then the computational problem of a

CSP simply disappears as all combinations of values for variables are allowed. Consequently, we resolve to studying the co-microstructure, which elicits the 'causes' of interaction among the variables.

3.1 Basic principle

In the co-microstructure of a binary CSP, it is easy to see that any two adjacent variable-value pairs constitute a minimal no-good set. Moreover, it is also easy to show that any *clique* in this graph constitutes a complete no-good set. By definition, any two variable-value pairs of a complete no-good set are inconsistent, they cannot possibly appear in the same solution to the CSP. So, any induced subproblem needs to contain no more than one variable-value pair[1] of a complete no-good set. Our strategy exploits the complete no-good sets of the co-microstructure. Thus, it is called co-microstructure based decomposition and denoted co-μBD. Our idea is to decompose the initial CSP into subproblems such that each resulting subproblem contains exactly one variable-value pair of a complete no-good set, plus one subproblem that contains none.

This decomposition is carried out as follows. Let $\{(V_1, v_1), (V_2, v_2), \ldots, (V_k, v_k)\}$ be a complete no-good set of size k. The decomposition generates $(k+1)$ subproblems $\mathcal{P}_i$. In each subproblem $\mathcal{P}_i$, the domains of the variables in $\mathcal{V}$ are reduced as follows.

Subproblem	Variable index	Domain
$\mathcal{P}_{1\leq i\leq k}$	$x = i$	$D_{V_x} \longleftarrow \{v_x\}$
	$(1 \leq x \leq k) \wedge (x \neq i)$	$D_{V_x} \longleftarrow D_{V_x} - \{v_x\}$
	$k+1 \leq x \leq n$	$D_{V_x} \longleftarrow D_{V_x}$
$\mathcal{P}_{k+1}$	$1 \leq x \leq k$	$D_{V_x} \longleftarrow D_{V_x} - \{v_x\}$
	$k+1 \leq x \leq n$	$D_{V_x} \longleftarrow D_{V_x}$

This process is illustrated in Fig. 2.

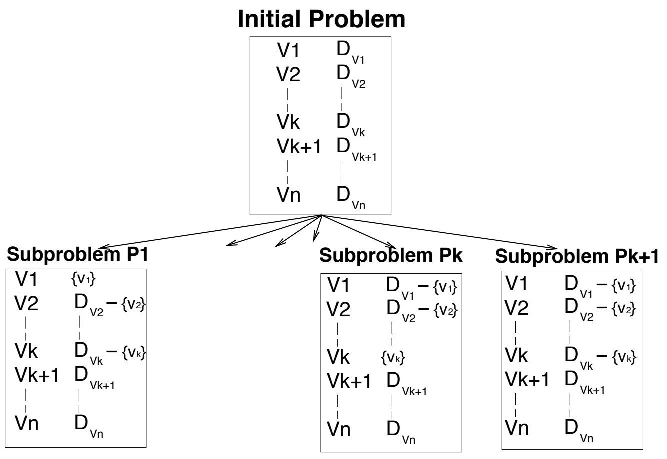

Figure 2: *Disjunction decomposition based on a complete no-good of size k.*

[1]We say that a problem contains a given variable-value pair when such a pair is a vertex in the co-microstructure corresponding to the problem.

3.2 Properties

According to the decomposition strategy described above, any generated subproblem contains at most one variable-value pair of a given CNG-set. So, all subproblems that contain two or more pairs are pruned out. Consequently, given a binary CSP $\mathcal{P} = (\mathcal{V}, \mathcal{D}, \mathcal{C})$, if n is the total number of variables, if k is the size of the complete no-good set S, and assuming that all the variables have the same domain size d ($\forall V_i \in \mathcal{V}, |D_{V_i}| = d$), we have the following relations[2]. The number of all possibilities of $\mathcal{P}$ is:

$$\text{Size}(\mathcal{P}) = d^n. \tag{1}$$

The total number of possibilities in the generated subproblems $\mathcal{P}_i$ is:

$$\text{Size}(\{\mathcal{P}_i\}) = k(d-1)^{k-1}d^{n-k} + (d-1)^k d^{n-k}. \tag{2}$$

The number of possibilities that are ruled out by decomposing according to S is:

$$\begin{aligned} \text{Gain}(\text{co}-\mu\text{BD}(S)) &= \sum_{i=2}^{k} \tbinom{k}{i}(d-1)^{k-i}d^{n-k} \\ &= (1) - (2) \end{aligned} \tag{3}$$

Our decomposition strategy prunes out only possibilities that are necessarily inconsistent and is guaranteed to keep all consistent solutions. The generated subproblems determine a partition of the solutions of the initial problem.

In addition to the basic properties listed in Section 2, Freuder and Hubbe [4] identify the following desirable properties of a decomposition strategy:

- *Completeness.* Any solution to $\mathcal{P}$ will be a solution to at least one of the $\{\mathcal{P}_i\}$.
- *Non-redundancy.* Any possibility for $\mathcal{P}$ appears in at most one of the $\{\mathcal{P}_i\}$.
- *Reducibility.* The sum of the sizes of the $\{\mathcal{P}_i\}$ may be less than the size of $\mathcal{P}$ (i.e., there are problems $\mathcal{P}$ for which at least one possibility does not appear in any of the $\{\mathcal{P}_i\}$).

It is easy to show[3] that the decomposition strategy described in Section 3.1 is consistent, simplifying, semi-complete, complete, non-redundant, and reducible.

[2] When the domains have different sizes, the Expressions (1) and (2) respectively become: $\text{Size}(\mathcal{P}) = \prod_{V_i \in \mathcal{V}} |D_{V_i}|$ and $\text{Size}(\{\mathcal{P}_i\}) = \prod_{i=k+1}^{n} |D_{V_i}| \times [\sum_{i=1}^{k} \prod_{1 \leq j \leq k, j \neq i} |D_{V_j}| + \prod_{i=1}^{k}(|D_{V_i}| - 1)]$.

[3] The proofs are straightforward and are not reported in this paper for lack of space.

3.3 Discussion

When the size of the CNG-set chosen for decomposition is equal to 2, co-μBD reduces into a particular form of forward-checking in which *binary* constraints applicable to a given variable-value pair are treated individually. For non-binary CSPs, one can apply this technique to the dual CSP, which is binary, obtained as described in [12].

According to Expression (3), the gain depends on the size of the CNG-set chosen: the bigger the set, the bigger the gain. Also, if we were to apply the decomposition strategy iteratively, this choice also reduces the number of iterations. Consequently, there is a clear advantage of choosing CNG-sets that are as big as possible, ideally the *maximal* cliques of the co-microstructure. However, the problem of finding all maximal cliques in a graph is known to be NP-complete [10]. This constitutes a limitation to the 'optimal' application of our strategy without, however, endangering the basic mechanism. Indeed, maximal cliques are desired but do not constitute a necessary requirement since our decomposition can exploit cliques of any sizes. Also, to find a clique of a *fixed* size k can be done in polynomial time[4]. Consequently, co-μBD is flexible with respect to two important points: one can selectively adapt (1) the gain drawn from the decomposition to the computational effort that one is ready to perform for finding the cliques and (2) the size of the clique at a given decomposition step to the number of processors available for parallelization.

In general, we expect this strategy to be more efficient in the case of *loose* constraints, which imply a co-microstructure of a relatively small number of edges. These arguments need to be investigated more deeply and experiments on randomly generated problems should be run for this purpose.

Note however, that if the co-microstructure of the CSP is a chordal graph, finding all maximal cliques can be done in linear time of the size of the graph. Such a possibility is not unrealistic and in Section 4, we report a practical application that we have modeled as a CSP and which co-microstructure is a triangulated graph.

4 Application to resource allocation

In our previous work [2], we have studied a resource allocation (RA) problem that consists of allocating qualified and specialized technicians to a set of pre-scheduled surgical operations in the surgical unit of a hospital. We defined the RA problem as follows: "Given a set of tasks with fixed endpoints, and given, for each task a set of resources that can carry out the task, assign one resource to each of the tasks such that no resource is assigned to two different tasks at the same time".

We represented this problem as a list coloring[5] problem in interval graphs

[4]A brute force search has a worst-case time complexity of $O(n^k)$, where n is the size of the graph and k is the size of the clique.

[5]List coloring differs from ordinary graph coloring, also called minimal coloring, in that the set of colors allowed for each node is restricted, and may be different for each node. In the literature, list coloring is also called *restricted coloring*.

and also as a discrete binary CSP with constraints of mutual exclusion. RA is known to be NP-complete [1]. Fig. 3 shows a simple example of an RA problem, the corresponding interval graphs and constraint graph. Note that, although

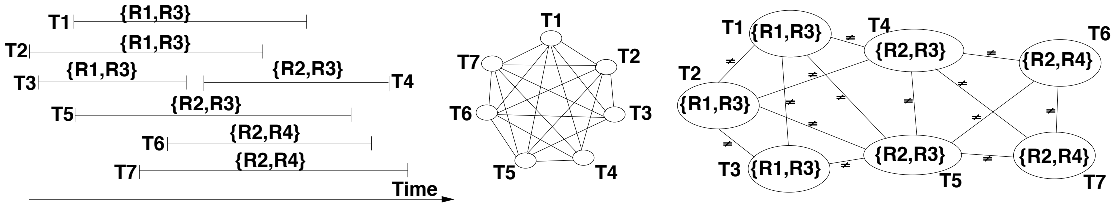

Figure 3: *Left:* A simple RA problem. *Center:* Its interval graph. *Right:* Its constraint graph.

the interval graph is triangulated [7], the constraint graph is not because the labels of the nodes in the constraint graph corresponding to two overlapping intervals can have an empty intersection thus implying a universal constraint between the nodes.

Fig. 4 shows the complementary microstructure corresponding to this example. It is easy to show that this graph is necessarily triangulated because of the underlying interval graph. This graph may actually consist of several non-connected components, all triangulated. Since co-μ(RA) is necessarily triangu-

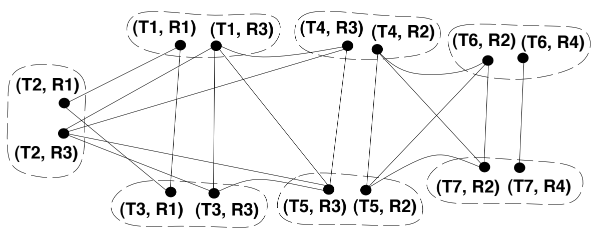

Figure 4: *The complementary microstructure of the example shown in Fig. 3.*

lated, finding maximal cliques can be done in linear time in the size of the graph [6]. As a result, the decomposition strategy that we described in Section 3.1 can be efficiently carried out for RA.

5 Relation to previous work

Our approach fits in the very general control schema proposed in [4]. A succinct comparison to other decomposition methods is shown in Table 5. Below, we discuss how it resembles and how it differs from some approaches to decomposition reported in the literature.

5.1 Ordered search

It is legitimate to draw a parallel between the variable and value ordering heuristics in an ordered backtrack search and the problem of ‘managing the

	BT	FC	NC	BT-CPR	IDC	μBD	co-μBD
Consistency	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$
Simplification	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$
Semi-completeness	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$
Completeness	$\surd$	$\surd$	$\surd$	$\surd$	$\times$	$\surd$	$\surd$
Non-redundancy	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\times$	$\surd$
Reducibility	$\times$	$\surd$	$\surd$	$\surd$	$\surd$	$\times$	$\surd$

$\surd$ =yes $\times$ = no

BT = Backtrack [4]. IDC = Inferred Disjunctive Constraint [3]
FC = Forward Checking [4]. μBD = Microstructure Based Decomposition [9].
NC = Network Consistency [11, 4]. co-μBD = co-Microstructure Based Decomposition.
BT-CPR = Backtrack with Cartesian Product Representation [8, 4].

Table 1: *A concise comparison of the properties of the main decomposition strategies.*

agenda'[6] of a decomposition schema (i.e., which subproblem to decompose first, which no-good set to choose first). Contrary to the commonly adopted 'wisdom' of first choosing the least constrained value for a variable, our approach shows how choosing 'completely constrained' values yields an interesting pruning of the problem space. Experiments on random problems and real-world data still need to be carried out for drawing empirical evidence.

5.2 VAD-based conjunctive decomposition

In [2], we introduced a decomposition strategy, based on the Value Assignment Delay heuristic (VAD), for solving resource allocation (RA) problems as defined in Section 4. The VAD acts by iteratively 'hiding' the most constrained values in RA thus gradually 'removing' the links in the constraint graph. Eventually, the initial problem is separated into subproblems, that are either independently solved or used to localize and characterize conflicts. The strategies advocated in the VAD and co-μBD can be contrasted as follows.

1. The decomposition generated by the VAD is conjunctive while co-μBD is disjunctive.

2. In order to identify the values that are most constrained, the VAD uses a cumulative curve over time of the contention of each value. Because of the interval graph structure underlying the RA problem, this choice is exactly the same as identifying maximal cliques in co-μ(RA).

3. Since the VAD chooses to 'hide' the most solicited set of values, it explores the subproblem denoted $\mathcal{P}_{k+1}$ in Section 3.1 (see also Fig. 2). The remaining alternatives (i.e., $\mathcal{P}_1$, $\mathcal{P}_2$, ..., $\mathcal{P}_k$) are partially explored by the conflict resolution procedure that complements the VAD.

[6]An agenda [4] is a heap that keeps track of the subproblems in the leaves of a tree hierarchy created by an iterative application of a given decomposition technique.

4. Finally, we think that the CNG-sets can be used to generalize to application of the VAD[7] to general CSPs in order to define new conjunctive decomposition strategies. Indeed, when the values involved in a CNG-set are 'hidden' from the domains of the variables involved, the corresponding edges in the co-microstructure are deleted. When this operation is applied iteratively, the constraint graph may eventually break down to non-connected components, which constitute the subproblems of the conjunctive decomposition. We believe that interesting strategies of problem solving and conflict localization can be developed based on this generalization of the application of the VAD to general CSPs.

5.3 Microstructure-based decomposition

Jégou [9] indicated that any maximal clique of size n in the microstructure of a CSP of n variables is a solution to the CSP. Because the identification of the maximal cliques is a difficult task, he advised to first triangulate the microstructure (which can be done in linear time) before identifying the maximal cliques (which can be efficiently carried out after triangulation). The triangulation procedure may add edges to the microstructure and the additional edges may appear in the identified maximal cliques. Because these edges correspond to non-consistent tuples, the maximal cliques are compared against the initial CSP and used to induce subproblems that are a reduction of the initial problem. Thus, the decomposition strategy (μBD) proposed by Jégou exhibits the following properties. It is consistent, simplifying, semi-complete, and complete, but it is redundant and non-reducible. The main distinctions between μBD and co-μBD can be summarized as follows:

1. The goal pursued by Jégou is to find solutions to the CSP by identifying the maximal cliques in $\mu(\mathcal{P})$. The necessity to decompose the CSP results from the difficulty of finding these cliques. In μBD, decomposition is *not* pursued for its own sake or for the sake of eliminating inconsistent solutions. Thus, μBD and co-μBD appear to appeal to decomposition for fundamentally distinct reasons.

2. The former is redundant (the same possibilities may appear in one or more $\mathcal{P}_i$), whereas the subproblems identified by the latter create a *partition* of the solution set of the CSP.

3. μBD *requires* the computation of the maximal cliques of a graph and directly suffers from the fact that this cannot be efficiently done. Conversely, co-μBD benefits from identifying maximal cliques but can correctly operate with cliques of any size.

4. It is impossible to predict the number of subproblems obtained by μBD at a given decomposition step, whereas this number can always be deliberately chosen in co-μBD.

[7]The VAD was first introduced for coloring problems (i.e., constraints of mutual exclusion).

5. It is impossible to assess the gain obtained by applying μBD, whereas the gain drawn from applying co-μBD is exactly determined by Expression (3). Indeed, the subproblems obtained by μBD directly depend on the algorithm used for triangulation (and thus are uncontrollable); whereas, in co-μBD, the content and size of each subproblem is exactly determined.

6. We suspect that the complementarity relation that exist between $\mu(\mathcal{P})$ and co-$\mu(\mathcal{P})$ may directly reflect on the effectiveness of applying these two strategies to CSPs. More specifically, we suspect that μBD may be more effective on tightly constrained CSPs (few edges in $\mu(\mathcal{P})$) whereas co-μBD may be more effective on loosely constrained CSPs (few edges in co-$\mu(\mathcal{P})$). These claims need further investigations and an empirical evaluation on random problems.

5.4 Decomposition using IDC

In [3], Freuder introduces a strategy[8] for decomposing a CSP into disjunctive subproblems that we describe from the perspective of our approach. The decomposition is carried out according to all minimal no-good sets in which a given variable-value pair is involved. The subproblem that contains none of the variable-value pairs of the no-good sets involved (i.e., subproblem $\mathcal{P}_{k+1}$ in our notation) is ruled out, this can be done while guaranteeing semi-completeness[9] but sacrificing completeness[10].

IDC exploits minimal no-good sets whereas co-μBD exploits complete no-good sets. When the co-$\mu(\mathcal{P})$ contains no cliques of size equal or bigger to 3, both strategies become equivalent. IDC can be more efficient than co-μBD (especially if the co-$\mu(\mathcal{P})$ is not chordal). However, it cannot be used if one is seeking all solutions, which is anyway a rather rare requirement. Also, for a given complete no-good set, IDC may generate more decomposition steps (thus, more subproblems) than co-μBD, which processes all no-goods of the set at once.

5.5 Subproblem extraction

In [5], Freuder and Hubbe introduce yet another disjunctive decomposition scheme based on extracting a given subproblem from the initial CSP. They argue that one may choose to do so if one knows that the subproblem is not solvable or if one is not interested in the possibilities that it contains (thus, the 'Factor Out Failure' (FOF) strategy that they propose). The extraction mechanism can be summarized as follows. Let $\mathcal{P}_e$ be the subproblem to extract from the initial CSP $\mathcal{P}$. A current problem $\mathcal{P}_c$ is initialized to $\mathcal{P}$. Iteratively for each variable V_i of $\mathcal{P}_c$, two subproblems are generated:

- The first subproblem contains for V_i its domain as it appears in $\mathcal{P}_e$ and for the other variables their domain specified in $\mathcal{P}_c$.

[8]Freuder also describes some variations around this strategy that reduce the size of the generated subproblems while keeping at least one solution, when the initial CSP has one.

[9]At least one solution is kept.

[10]Some consistent solutions are lost.

- The second subproblem contains for V_i its domain as it appears in $\mathcal{P}_c$ of which is subtracted its domain specified in $\mathcal{P}_e$, and for the other variables their domain specified in $\mathcal{P}_c$. $\mathcal{P}_c$ is reset to this second generated subproblem.

It is easy to verify that, if the values that are outside the domains specified in $\mathcal{P}_e$ belong to a complete no-good set, then this extraction mechanism generates subproblems of size bigger than the size of those generated by co-μBD (see the example below). Also, unnecessary intermediate subproblems would need to be generated. Hence, when CNG-sets can be identified, co-μBD is more suited than the extraction procedure described above for use in the 'Factor Out Failure' strategy of [5]. Such a situation is illustrated in Fig. 5, where $\mathcal{P}_e$ is shown in a dark rectangle and the CNG-set $\{(V_1, a), (V_2, a), (V_3, a)\}$ has been selected for applying co-μBD.

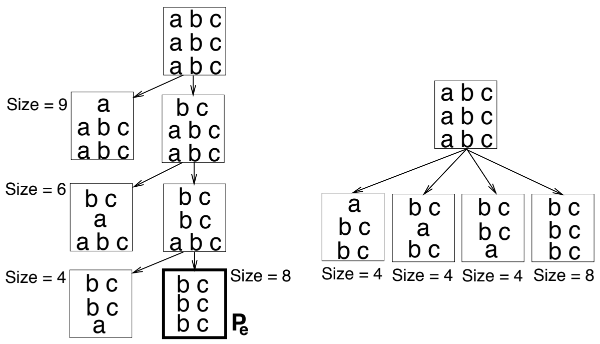

Figure 5: *Left:* Extraction of subproblem. *Right:* Decomposition using co-μBD.

6 Conclusions

In this paper, we introduced a new disjunctive decomposition strategy based on identifying cliques in the complementary microstructure of a CSP. One important characteristic of this strategy is its flexibility with respect to the choice the clique size. This flexibility can be exploited in selectively choosing the number of subproblems to be generated and also the amount of effort required for finding the cliques. We studied the properties of the strategy and compared it to those of techniques reported in the literature. We also identified a practical application for which this strategy can be efficiently applied. One interesting direction for future research is the exploitation of complete no-good sets for generating a conjunctive decomposition strategy and for localizing conflicts applicable to general CSPs, as a generalization of the application of VAD heuristic, reported in [2], to general CSPs. Future efforts should also be concerned with doing experimental studies on randomly generated problems in order to characterize the effectiveness of this approach with respect to CSP parameters (e.g., tightness, looseness, density).

Acknowledgments

The authors are indebted to Rainer Weigel for various and very helpful discussions and to the anonymous reviewers for their comments.

References

[1] Esther Arkin and Ellen Silverberg. Scheduling Jobs with Fixed Start and End Times. *Discrete Applied Mathematics*, 18:1–8, 1987.

[2] Berthe Y. Choueiry and Boi Faltings. A Decomposition Heuristic for Resource Allocation. In *Proc. of the 11 th ECAI*, pages 585–589, Amsterdam, The Netherlands, 1994.

[3] Eugene C. Freuder. Using Inferred Disjunctive Constraints to Decompose Constraint Satisfaction Problems. In *Proc. of the 13 th IJCAI*, pages 254–260, Chambéry, France, 1993.

[4] Eugene C. Freuder and Paul D. Hubbe. A Disjunctive Decomposition Control Schema for Constraint Satisfaction. In Vijay Saraswat and Pascal Van Hentenryck, editors, *Principles and Practice of Constraint Programming*, pages 319–335. MIT Press, Cambridge, MA, 1995.

[5] Eugene C. Freuder and Paul D. Hubbe. Extracting Constraint Satisfaction Subproblems. In *Proc. of the 14 th IJCAI*, pages 548–555, Montreal, Canada, 1995.

[6] 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.

[7] Martin C. Golumbic. Algorithmic Aspects of Perfect Graphs. *Annals of Discrete Mathematics*, 21:301–323, 1984.

[8] Paul D. Hubbe and Eugene C. Freuder. An Efficient Cross Product Representation of the Constraint Satisfaction Problem Search Space. In *Proc. of AAAI-92*, pages 421–427, San Jose, CA, 1989.

[9] Philippe Jégou. Decomposition of Domains Based on the Micro-Structure of Finite Constraint-Satisfaction Problems. In *Proc. of AAAI-93*, pages 731–736, Washington, DC, 1993.

[10] Richard M. Karp. Reducibility among Combinatorial Problems. In E. M. Miller and J.W. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, New York, 1972.

[11] Alan K. Mackworth. On Reading Sketch Maps. In *Proc. of the 5 th IJCAI*, pages 598–606, Cambridge, MA, 1977.

[12] Francesca Rossi, Charles Petrie, and Vasant Dhar. On the Equivalence of Constraint Satisfaction Problems. In *Proc. of the 9 th ECAI*, pages 550–556, Stockholm, Sweden, 1990.