Electronic Colloquium on Computational Complexity, Report No. 38 (1999)

On the Complexity of Finding Satisfiable Subinstances in Constraint Satisfaction

Peter Jonsson
Department of Computer and Information Science
Linköping Universitet
S-58183 Linköping, Sweden
petej@ida.liu.se

Paolo Liberatore*
Department of Computer and System Science
University of Rome "La Sapienza"
Via Salaria 113
I-00198 Rome, Italy
liberato@dis.uniroma1.it

August 27, 1999

Abstract

We study the computational complexity of an optimization version of the constraint satisfaction problem: given a set F of constraint functions, an instance consists of a set of variables V related by constraints chosen from F and a natural number k. The problem is to decide whether there exists a subset $V' \subseteq V$ such that $|V'| \geq k$ and the subinstance induced by V' has a solution. For all possible choices of F, we show that this problem is either NP-hard or trivial. This hardness result makes it interesting to study relaxations of the problem

*This work has been done while the second author was visiting Linköping University.

ISSN 1433-8092

which may have better computational properties. Thus, we study the approximability of the problem and we consider certain compilation techniques. In both cases, the results are not encouraging.

1 Introduction

The constraint satisfaction problem provides a framework in which it is possible to express, in a natural way, many combinatorial problems encountered in computer science, mathematics and elsewhere. A constraint satisfaction problem CSP(F) consists of some finite set $F = \{f_i : D^{k_i} \to \{0,1\}\}$ of functions where D is some (not necessarily finte) fixed set. An instance of the problem is a set of constraint functions applied to specified subsets of variables and the aim is to find an assignment to the variables that satisfies all constraint applications, *i.e.*, make them evaluate to 1. Note that F is not part of the input of the problem. This problem is known to be NP-hard in the general case and NP-complete when the domains of the variables are finite. However, much more is known about the complexity of the constraint satisfaction problem and its variants. The most well-known result is probably that by Schaefer [1978]. He has shown that there are essentially four classes of polynomial-time solvable constraint satisfaction problems on domains of size 2, namely,

1. 0-valid problems, *i.e.*, problems where the all-zero assignment is always a solution, and similarly the 1-valid problems;
2. Horn problems, *i.e.*, problems where every constraint function can be expressed as the conjunction of Horn clauses, and similarly anti-Horn problems;
3. 2SAT problems where every constraint function can be expressed as the conjunction of clauses of length 2; and
4. affine problems where every constraint function is the solution set of some set of linear equations modulo 2.

In all other cases, the problem is NP-complete. Similar complete classifications have been obtained for a number of other problems over domains of size 2, including the following:

- The MAX-CSP and MIN-CSP problems where the objective is to maximize (minimize) the number of satisfied constraints [Creignou, 1995, Khanna *et al.*, 1997b, Khanna *et al.*, 1997a]. Khanna *et al.* also studies the MAX ONES and MIN ONES problems where the objective is to maximize (minimize) the number of variables that is given the value 1. Further variations on these problems are considered in Jonsson [1999].
- The problem of finding the lexicographically maximal or minimal solution [Reith and Vollmer, 1998].
- The problem of counting the number of solutions [Creignou and Hermann, 1996].
- The problem of telling whether a set of solutions represents the set of all solutions [Kavvadias and Sideri, 1998].

Complexity results for domains of cardinality larger than 2 are more sparse. However, a large number of polynomial-time subclasses have been identified, *c.f.*, [Feder and Vardi, 1998, Jeavons and Cooper, 1996, Cooper *et al.*, 1994]. Certain graph-theoretic results, such as the dichotomy theorem for H-colourings of graph [Hell and Nešetřil, 1990], can be transferred to the constraint satisfaction domain by using the connection between graph homomorphisms and CSP which has been pointed out several times [Feder and Vardi, 1998, Jeavons, 1998]. This results says that if we restrict ourselves to CSP problems containing only one relation R which is binary (*i.e.*, only involves two variables) and symmetric ($R(x, y)$ holds iff $R(y, x)$ holds), then the CSP problem is either polynomial or NP-complete. A classification of the complexity of the general CSP problem for arbitrary domain is not known.

The main result of this paper is a complete classification of the complexity of a problem we call MAXIMUM INDUCED CSP(F) (MAX-IND-CSP(F)). An instance of this problem consists of a set of variables V (taking values from a domain which may be infinite) related by constraints picked from the set F and a natural number k. The computational problem is to decide whether there exists a subset $V' \subseteq V$ such that $|V'| \geq k$ and the induced subinstance of the CSP instance has a solution. We show that depending on the choice of F, this problem is either NP-hard or polynomial. In fact, the problem is polynomial iff all instances of CSP(F) have a solution. Consequently, the MAX-IND-CSP(F) problems solvable in polynomial time are trivial and of no interest.

Observe the difference between MAX-IND-CSP and MAX-CSP: the aim in MAX-CSP is to maximize the number of satisfied constraints and not to maximize the size of the largest induced satisfiable subinstance. MAX-CSP is in a certain sense easier problem then MAX-IND-CSP since there exists a nontrivial set F of constraint functions such that MAX-CSP(F) is solvable in polynomial time. (*e.g.*, Khanna *et al.* [1997b] shows that this holds when F only contains so-called *2-monotone* constraint functions.) One should also observe that MAX-IND-CSP is not an artificial problem without natural applications. Consider, as an example, scheduling or timetabling problems where the variables denotes start and stop times. Knowing how many of the timing constraints that can be satisfied is probably not the most important piece of information in practice (the MAX-CSP problem); the important thing is to know how many of the start and stop times that simultaneously can be acheived (the MAX-IND-CSP problem). Furthermore, many well-studied combinatorial problems, such as the independent set problem for graphs or hypergraphs, can easily be expressed as MAX-IND-CSP problems.

Having shown that MAX-IND-CSP is computationally hard except in a few trivial cases, it is interesting to study relaxations of the problem. Thus, we also consider two variations of the original problem.

In the first case, we begin by observing that the input of certain problems can be divided into two parts: One, called the *fixed* part, is known in advance, while the other one, called the *varying* part, is only available at the same time as the actual computation. Such a distinction is interesting when many input instances share the same fixed part. Then, it makes sense to preprocess *off-line* the fixed part, putting it into a form such that solving *on-line* a set of instances becomes easier. This idea of preprocessing part of the input has been used in many areas of computer science such as computational geometry. For example, certain techniques for geometric searching (cf. [Preparata and Shamos, 1985]) use a costly preprocessing phase to speed up the on-line processing.

In our "compilation" version of MAX-IND-CSP(F), we divide the input into a fixed part Π consisting of a set of constraints over F, and a varying part $\langle V', k\rangle$ where V' is a subset of the variables and k is a natural number such that $k \leq |V'|$. The question is whether the CSP instance Π restricted to the variables in V' has a solution on $\geq k$ variables. Clearly, this problem is at least as hard as the original MAX-IND-CSP problem. Also note that the problem would be trivial if V' was not included in the varying part.

Concerning this problem, we show the following: for any set F of constraint functions such that there exists unsatisfiable instances of CSP(F), it is not possible to compile the fixed part into a polynomially-sized data structure such that the on-line processing can be carried out in polynomial time (unless NP $\subseteq$ P/poly, which implies that the polynomial hierarchy collapses [Karp and Lipton, 1980]). Obviously, it is only interesting to compile the fixed part to a small data structure since the problem becomes trivial if we allow compilation into arbitrarily-sized structures.

Finally, we study the approximability of MAX-IND-CSP(F) and the results are negative also in this case. We are not able to give a complete answer to the question of when MAX-IND-CSP(F) is efficiently approximable and when it is not. However, if all functions in F have arity 2 or the constant-zero function can be "implemented" by the functions in F, then MAX-IND-CSP(F) cannot be approximated within n^δ for some $\delta > 0$ in polynomial time, unless P = NP. Stronger nonapproximability bounds are also given in certain special cases and under other complexity-theoretic assumptions.

The paper has the following organization: Section 2 contains the basic definitions and Section 3 provides the complexity results for MAX-IND-CSP. The compilability and approximability results can be found in Sections 4 and 5, respectively. A brief discussion of the results and concluding remarks are collected in Section 6.

2 Preliminaries

In this section we define the problem of constraint satisfaction, and some related concepts. Basically, a *constraint* is a condition that can hold or not over a set of objects. An instance of the constraint satisfaction problem is a set of constraints, and our aim is to find a set of objects that satisfy all constraints. Constraints can refer to a single or multiple objects, and they are thus classified on the basis of their arity. For instance, a unary constraint is a condition over a single object, while a binary constraint specify that some relation must hold between two objects.

Let us refer to the basic case of algebra: objects are numbers, and constraints are relations over them. A unary constraint is for instance $x_1 > 0$. This is a condition over the variable x_1, since only numbers greater than 0 satisfy this condition. On the other hand, a binary constraint like $x_1 = x_2$

do not give a condition over x_1 alone: rather, it specify that x_1 and x_2 must be related in some way (in this case, they must be equal).

In this paper, we are concerned with constraints of arity ≥ 2, *i.e.*, we do not consider unary constraints at all. A discussion of unary constraints is in the conclusion of the paper: the complexity of the constraint satisfaction problem with unary constraints is left open by this work.

Let us formally define the problem of constraint satisfaction. Let D be an arbitrary non-empty collection of objects and $F = \{f_1, \ldots, f_n\}$ be a finite set of functions from $D^{k_i} \to \{0, 1\}$, $1 \leq i \leq n$ and $k_i \geq 2$ for every i. We call k_i the arity of function f_i. An instance of the constraint satisfaction problem is defined as follows.

Definition 1 *An instance of the* constraint satisfaction problem over F *(CSP(F)) consists of:*

1. *a finite set of variables, V;*
2. *a finite set of constraints $C = \{C_1, \ldots, C_m\}$. Each constraint C_i is a list $(f_i, x_1, \ldots, x_p)$ where $f_i \in F$, $x_1, \ldots, x_p$ are distinct members of V and $p = k_i$,* i.e.*, p equals the arity of f_i.*

Given a constraint $c = (f, x_1, \ldots, x_p)$, we say that f is the function *of c and the list $(x_1, \ldots, x_p)$ is the* argument *of c.*

The domain D is not explicitly given in the definition of CSP(F), since it can be easily inferred from F. Given an instance of CSP(F), the basic computational problem is to verify whether it is possible to assign a value to any variable in order to satisfy all constraints. To this extent, we give the following definition.

Definition 2 *A solution to an instance Π of CSP(F) is a function $S : V \to D$ such that for each constraint $(f, x_1, \ldots, x_p) \in C$, $f(S(x_1), \ldots, S(x_p)) = 1$. If Π has a solution, we say that Π is* satisfiable.

An instance of the constraint satisfaction problem may be unsatisfiable. In such cases, we may look for a partial solution, that is, an assignment of values to a subset of variables. We define an induced subinstance of a CSP problem as follows.

Let $\Pi = \langle V, C \rangle$ be an instance of the CSP(F) problem and assume that $V' \subseteq V$. We define the subinstance of Π *induced* by V', $\Pi|V' = \langle V', C' \rangle$ where

$$C' = \{(f, x_1, \ldots, x_n) \in C \mid x_1, \ldots, x_n \in V'\}.$$

The problem of finding solutions for a subinstance is defined as follows.

Definition 3 *An instance of the* MAXIMUM INDUCED CSP(F) *(*MAX-IND-CSP*(F)) problem consists of an instance* $\Pi = \langle V, C \rangle$ *of the* CSP(F) *problem and an integer* $1 \leq k \leq |V|$. *A solution to an instance of* MAX-IND-CSP*(F) is a subset* $V' \subseteq V$ *of the variables satisfying the following requirements:*

1. $|V'| \geq k$; *and*
2. $\Pi|V'$ *is satisfiable.*

The concept of graph corresponding to a CSP instance is very useful, as it allows for a graphical representation of CSP instances. This representation can be used whenever all functions have arity two.

Definition 4 *The* graph corresponding to a CSP instance *is defined as follows: for each variable of the CSP instance there is a node of the graph; for any constraint* (f, x, y), *there is an edge labeled* f *between the nodes corresponding to* x *and* y.

Given a graph, we can construct a CSP instance using one function: for each node of the graph there is a variable, and for each edge (u, v) there is a constraint (f, x, y), where x and y are the variables corresponding to u and v, respectively.

Let us consider an example.

Example 5 Let $D = \{0, 1\}$ and $F = \{f_1\}$, where the f_1 is a function defined as:

$$f_1(x, y) = \begin{cases} 1 & \text{if } x \neq y \\ 0 & \text{otherwise} \end{cases}$$

Let us consider the following constraint satisfaction problem:

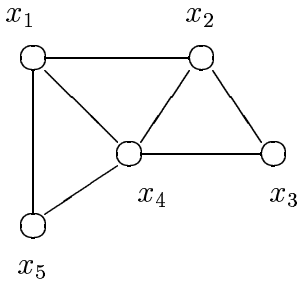


Figure 1: An instance of the constraint satisfaction problem.

$$
\begin{aligned}
V &= \{x_1, x_2, x_3, x_4, x_5\} \\
C &= \{(f_1, x_1, x_5), (f_1, x_1, x_2), (f_1, x_2, x_3), \\
&\quad (f_1, x_4, x_5), (f_1, x_1, x_4), (f_1, x_2, x_4), (f_1, x_3, x_4)\}
\end{aligned}
$$

Since all constraints are binary, we can represent this instance as a graph. Each node of the graph represents a variable of the problem, and constraints are represented by edges. The graph corresponding to the above instance is in Figure 1. Since f_1 is the only function, it is not necessary to label edges with this function.

It is easy to see that this instance is unsatisfiable: for instance, if we assign 1 to x_1, we are forced to assign 0 to x_2 because of the constraint (f_1, x_1, x_2), which is satisfied only if x_1 and x_2 are assigned different values. For the same reason, we must give 0 to x_4. This means that the constraint (f_1, x_4, x_2) is not satisfied, since x_2 and x_4 have the same value.

Finding a solution of the maximal induced CSP problem amounts to find a maximal subset of variables which is satisfiable. In the example above, we were able to find a solution for x_1 and x_2, but not for x_1, x_2, and x_4. In Figure 2 we show the instance induced by $\{x_1, x_2, x_3, x_5\}$, which is the maximal satisfied induced subinstance.

So far, we have given the definitions of the problems of interest. We now define some concept that will be useful in the sequel. First, we define the concept of nontrivial set of functions.

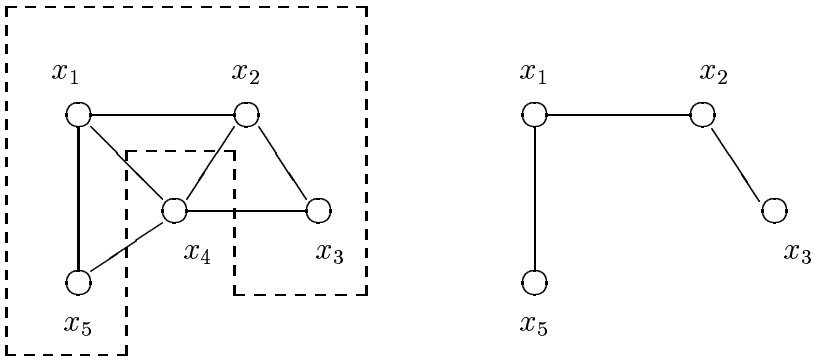


Figure 2: An induced subinstance

Definition 6 *A set of functions F is* nontrivial *if and only if there exists an instance of CSP(F) that have no solutions.*

If a set of functions is trivial, then any instance has a solution. As a result, the problem of deciding whether a solution exists is also trivial.

Symmetric functions are very important in our proofs. Defining symmetry for a binary function is straightforward: f is symmetric if and only if, for any x and y in the domain it holds $f(x, y) = f(y, x)$. To define symmetry in the general case, let us denote S_k the set of permutations on $\{1, \ldots, k\}$.

Definition 7 *A function $f : D^k \to \{0, 1\}$ is symmetric if and only if it holds*

$$f(x_1, \ldots, x_k) = f(x_{\sigma_{(1)}}, \ldots, x_{\sigma_{(k)}})$$

for any possible choice of $\sigma \in S_k$, and any $x_1, \ldots, x_k \in D$.

Given a domain D and a set of functions F, we define the intersection of the functions in F as follows.

Definition 8 *Let D be an arbitrary non-empty collection of objects and $F = \{f_1, \ldots, f_n\}$ be a finite set of functions from $D^{k_i} \to \{0, 1\}$, $1 \leq i \leq n$ and $k_i \geq 2$ for every i.*

Let $a = \max(k_1, \ldots, k_n)$ and define the function $f : D^a \to \{0, 1\}$ as

$$f(x_1, \ldots, x_a) = \min \left\{ f_i(x_{\sigma(1)}, \ldots, x_{\sigma(k_i)}) \mid 1 \leq i \leq n \text{ and } \sigma \in S_a \right\}$$

We call this function the intersection of $\{f_1, \ldots, f_n\}$.

It can be easily proved that, given an instance Π of CSP($\{f\}$), where f is the intersection of F, we can in polynomial time build an instance Π' of CSP(F) which is satisfiable if and only if Π is. Furthermore, if the maximal satisfied induced subinstance of Π contains k variables, then the maximal induced satisfiable subinstance of Π' also contains k variables (no new variables are introduced by the transformation).

Some other useful properties of f are the following:

1. f is a symmetric function; and
2. if F is nontrivial, then $\{f\}$ is nontrivial.

3 Complexity

In this section we present our results concerning the complexity of the maximum induced constraint satisfaction problem. We show that, for any nontrivial set F of functions, the problem of deciding the existence of an induced satisfiable subinstance containing a given number of variables is NP-hard, and then we show a sufficient condition for membership in NP. We first define the concept of induced subgraph.

Given a graph $G = \langle V, E \rangle$ and a subset of its nodes $V' \subseteq V$, the subgraph of G induced by V' is: $G|V' = \langle V', \{(u, v) \mid u, v \in V'\} \rangle$. The INDEPENDENT SET problem is a well-studied problem on graphs.

Definition 9 *The problem* INDEPENDENT SET *is defined as:*

INSTANCE: *graph $G = \langle V, E \rangle$, integer k*

QUESTION: *is there a $V' \subseteq V$ such that $|V'| \leq k$ and $G|V'$ has no edges?*

Our proofs use the concept of clique. A clique is a graph $G = \langle V, E \rangle$ such that, for any $v, w \in V$, it holds $(v, w) \in E$.

In the sequel, we sometimes say "the graph G contains a clique..." meaning that there exists a subset of nodes $V' \subseteq V$ such that $G|V'$ is a clique. Moreover, a clique containing r nodes is denoted K_r.

For the sake of clarity, we show two separate theorems: first we prove the NP-completeness for the case in which all functions have arity 2, and then we prove the claim in the general case. The reason for doing this is that the proof of the first theorem is easier to understand thanks to the correspondence between CSP instances and graphs.

Theorem 10 MAX-IND-CSP is NP-hard, for any nontrivial set of binary functions.

Proof: We exhibit a reduction from INDEPENDENT SET. Given a nontrivial set of binary functions F, we prove the hardness by using an instance containing only the intersection of all functions in F, which we denote by f. As already remarked, NP-hardness of MAX-IND-CSP($\{f\}$) immediately carries over to MAX-IND-CSP(F).

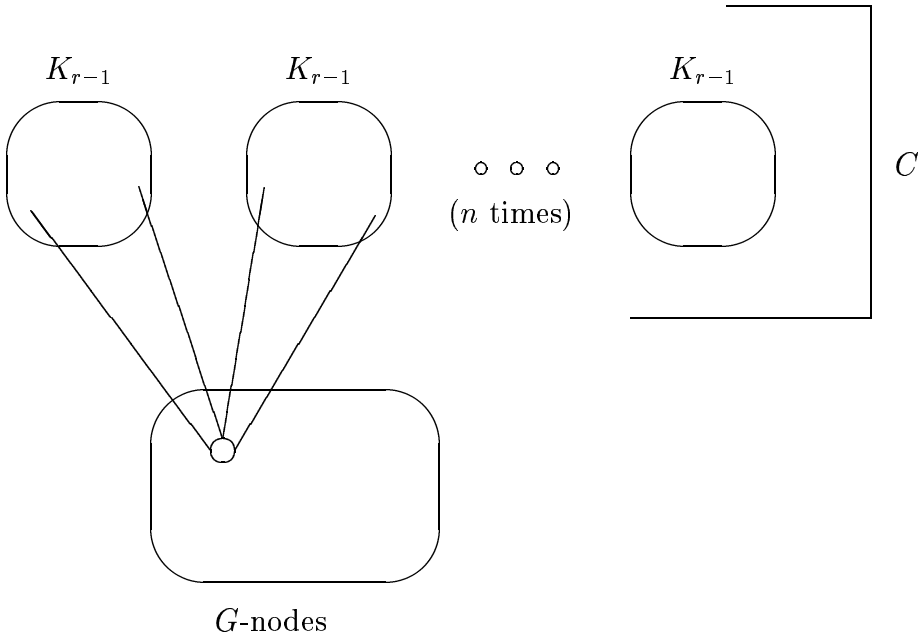
Given a CSP instance containing only one symmetric and binary function, we consider the corresponding graph according to Definition 4. From now on, we say that a graph (or a sub-graph) is satisfiable if and only if the corresponding CSP instance is.

Since there exist unsatisfiable instances of CSP(F), there exist unsatisfiable instances of CSP($\{f\}$) and, consequently, unsatisfiable graphs. Let r be the greatest number such that the clique K_r is satisfiable. This number exists since, otherwise, every clique would be satisfiable. Since each graph is a subgraph of a clique, and satisfiability is anti-monotonic (that is, removing edges from a satisfiable graph leads to a satistiable graph), this would imply that each graph is satisfiable, contradicting the assumption of non-triviality of $\{f\}$.

Let $\{v_1, \ldots, v_r\}$ be the values that must be assigned to the variables in order to satisfy a clique of size r. This means that $f(v_i, v_j) = 1$ for each $i \neq j$. Since there are unsatisfiable instance, it also holds that $f(v_i, v_i) = 0$ for any value v_i.

We can now give the reduction from INDEPENDENT SET to MAX-IND-CSP($\{f\}$). Given an arbitrary graph G, we create an instance of MAX-IND-CSP($\{f\}$) as follows. Let n be the number of nodes of G: we add n copies

of K_{r-1} to G, and then we link each node of G to each of the nodes of each copy of K_{r-1}. Let us denote this graph as H. We say that a node v is a G-node if and only if it appears in the graph G. Let C be the set of the nodes of the added cliques. We can represent H graphically as follows.



Let us consider the case $r = 1$ first, since this is the easiest case. The fact that $r = 1$ implies that the clique with two nodes is unsatisfiable. As a result, $f(x, y) = 0$ for any value of x and y. In this case, the graph H is equal to G itself, and an induced subset of it is satisfiable if and only if it contains no edges. As a result, an induced subgraph is satisfiable if and only if it is an independent set. We have thus reduced the INDEPENDENT SET problem to MAX-IND-CSP(F).

Now consider the case in which $r > 1$. If the graph G has n nodes, the number of nodes of H is $(r-1)n + n = nr$, thus linear in n. Now, we prove that there exits a set $V' \subseteq V$ of cardinality k, such that $G|V'$ has no edges if and only if there exists a set of variables X' of cardinality $(r-1)n + k$, such that $H|X'$ is satisfiable. We prove the claim in two steps.

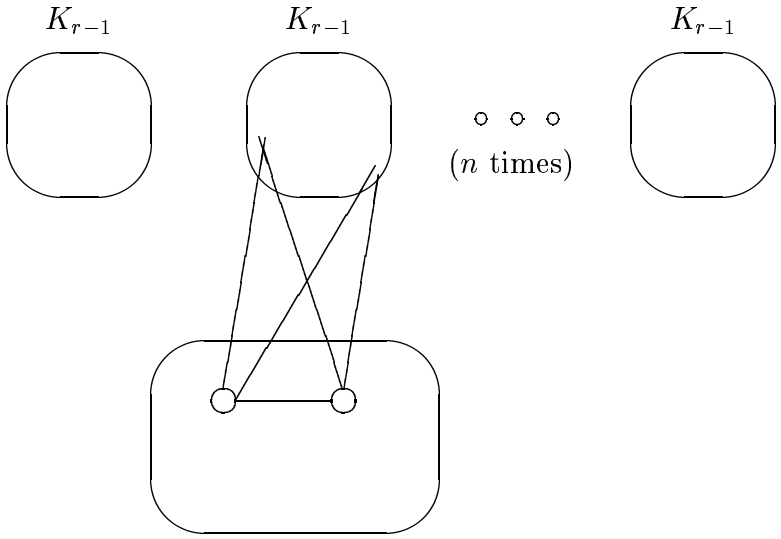
Only if. Assume that the graph $G|V'$ is an independent set (i.e. it has no edges). Let $X' = C \cup V'$. Since $G|V'$ is an independent set, there is no

edge between any two G-nodes of $H|X'$. The only edges in $H|X'$ are those between two nodes of a clique, or between a node of a clique and a G-node.

We prove that the graph $H|X'$ is satisfiable. Assign $\{v_1, \ldots, v_{r-1}\}$ to the nodes of the cliques, and v_r to all the G-nodes. Since there are no edges between any two G-nodes, we have that no pair of connected nodes are assigned the same value. Since $f(v_i, v_j) = 1$ whenever $v_i \neq v_j$, this instance is satisfied by the assignment.

If. Assume that, for any $V' \subseteq V$ of size k, the graph $G|V'$ contains some edges. We prove that, for any set of variables X' of size $(r-1)n+k$, the CSP instance $H|X'$ is unsatisfiable. We consider three cases: the first is when X' contains all the nodes of the cliques, the second is when X' contains all the G-nodes, and finally, the case in which none of the above is true.

1. Assume that $C \subseteq X'$. Since X' has size $(r-1)n+k$, and contains all the nodes of the cliques, it means that it contains exactly k of the G-nodes of H. As a result, there is still some edge between two G-nodes. Consider any clique, and the G-nodes of the graph.



There is at least a pair of G-nodes that are joined. These node are joined to each other, and to all the nodes of the clique. This means

that the clique plus the two nodes form a clique of size $r+1$, which we know is unsatisfiable.

2. Assume that X' contains all the G-nodes. Now, X' has size $(r-1)n+k$, while the number of G-nodes is n. As a result, that X' contains exactly $(r-2)n+k$ nodes from the cliques, which means that X' contains all nodes but $n-k$ chosen from C. Since there are n cliques, it follows that there is at least one clique from which no nodes are removed. Since G contains at least one edge, the proof of the previous case applies also here.

3. Assume that X' contains a part of the G-nodes and a part of the nodes of the clique. Since X' contains all nodes but $n-k$ it is still true that there is some clique from which no nodes are removed. Since G does not become an independent set by removing $n-k$ nodes, it also holds that $H|X'$ contains at least one edge between two G-nodes. Once again, the argument of the first case can be used to prove that the instance $G|X'$ is unsatisfiable.

As a result, we proved that G has an independent set of size k if and only if H has an induced satisfiable subinstance containing $(r-1)n+k$ variables. □

Note that, if the domain is finite, saying that the problem have unsatisfiable instances is equivalent to say that $f(x,x)=0$ for each x in the domain, where f is the intersection of all functions.

Now we prove that the result of NP-completeness holds even if we consider functions of arbitrary arity. In this second theorem we cannot use the correspondence between CSP instances and graphs, which makes the proof somewhat harder to follow.

Theorem 11 MAX-IND-CSP(F) is NP-hard, for any nontrivial set of functions F.

Proof: We prove the NP-hardness by exhibiting a polynomial-time reduction from INDEPENDENT SET.

Let a be the maximal arity of the functions under consideration and let f be the intersection of F. Since F is nontrivial, there exists an unsatisfiable instance of CSP($\{f\}$).

Let us define $K(x_1, \ldots, x_r)$ to be the set of constraints having f as function and any subset of a variables of $x_1, \ldots, x_r$ as argument.

$$K(x_1, \ldots, x_r) = \{(f, y_1, \ldots, y_a) \mid \{y_1, \ldots, y_a\} \subseteq \{x_1, \ldots, x_r\}\}$$

Let r be the greatest number such that set of constraints $K(x_1, \ldots, x_r)$ is satisfiable while $K(x_1, \ldots, x_{r+1})$ is not. We can easily prove that such a number exist. Indeed, let Σ be the smallest (in terms of number of variables) unsatisfiable instance. Let s be the number of nodes it contains. We prove that $r = s - 1$. Since Σ is the smallest unsatisfiable instance, and $K(x_1, \ldots, x_r)$ has $s - 1$ variables, it follows that the latter is satisfiable. It also holds that $K(x_1, \ldots, x_{r+1})$ is unsatisfiable, since Σ is a subset of it. Note that r must be greater than or equal to a.

Consider an instance $\langle G, k \rangle$ of the INDEPENDENT SET problem. Let n be the number of nodes of the graph G. We construct an instance of MAX-IND-CSP($\{f\}$), such that the graph G contains an independent set of cardinality k if and only if there exists a subset of the variables of Π such that the induced subinstance of Π is satisfiable and the subset contains $n(r-1)+k$ variables.

The instance Π is quite similar to the construction used in the proof of the previous theorem. We have a set of variables $x_1, \ldots, x_n$ in one-to-one correspondence with the nodes of the graph, and a set $y_1^1, \ldots, y_{r-1}^n$ of $n(r-1)$ auxiliary variables.

The constraints are defined as follows: for each $i = 1, \ldots, n$ and each $x_j \in \{x_1, \ldots, x_n\}$ we add the constraints $K(x_j, y_1^i, \ldots, y_{r-1}^i)$. Moreover, if x_j and x_z are variables whose corresponding nodes in the graph G are joined by an edge, we add the set of constraints $K(x_j, x_z, y_1^i, \ldots, y_{r-1}^i)$ for each i. Note that the set of constraints $K(\ldots)$ is equivalent to a single constraint using a symmetric function, thus the ordering of the variables is irrelevant.

Thus, we have the following constraints:

1. For each subset of a variables of $y_1^i, \ldots, y_{r-1}^i$, there is a constraint having f as function.

2. For each variable x_j corresponding to a node of the graph, and for each subset of $a-1$ variables from $y_1^i, \ldots, y_{r-1}^i$ there is a constraint having f as function.

3. For each pair of variables x_j and x_z corresponding to nodes that are joined in the graph, and for each subset of $a-2$ variables from $y_1^i, \ldots, y_{r-1}^i$, there is a constraint having f as function.

Let us call this CSP instance Π. We prove that the graph G has an independent set of size k if and only if there exists a satisfiable subset of nodes of the CSP having size $n(r-1)+k$. In fact, we prove that there exists a set $V' \subseteq V$ containing all but h nodes, such that $G|V'$ contains no edges if and only if there exists a set X' of variables, containing all but h variables, such that $\Pi|X'$ is satisfiable.

There exists a set V' of $n-h$ nodes such that $G|V'$ contains no edges. Define the set X' such that it contains Y and the variables x_j that corresponds to the nodes of V'. Clearly, this set contains all the variables but h of them. The set X' does not contain any pair of variables x_j, x_z such that the corresponding nodes of the graph are joined by an edge. As a result, the instance $\Pi|X'$ does not contain any constraint in which two nodes x_j, x_z are in the argument at the same time.

By assumption, the set of constraints $K(y_1, \ldots, y_r)$ is satisfiable. As a result, there exist r elements of the domain $\{v_1, \ldots, v_r\}$ that satisfy all the possible constraints over a set of r variables. Assign the values $\{v_1, \ldots, v_{r-1}\}$ to the variables $y_1^i, \ldots, y_{r-1}^i$, for any value of $i \in \{1, \ldots, n\}$. Then, assign v_r to the all the variables $\{x_1, \ldots, x_n\}$.

For each set V' of $n-h$ nodes $G|V'$ contains edges. Let X' be a set containing all variables but h of them. We consider what happens when h variables are removed from Π.

First assume that all the removed variables are among those corresponding to the nodes of the initial graph G. Since it is impossible to obtain an independent set from the graph removing h nodes, it follows that there are two variables left x_j and x_z, such that the corresponding nodes of the graph are joined by an edge. As a result, the set of constraints $K(x_j, x_z, y_1^i, \ldots, y_{r-1}^i)$ is still intact for any i. We know that it is impossible to satisfy all the constraints over $r+1$ variables. As a result, Π is unsatisfiable.

The case in which the variables are removed from the set $y_1^1, \ldots, y_{r-1}^n$ is identical: since $h \leq n-1$, we are left with a set $y_1^i, \ldots, y_{r-1}^i$ from which no variable is removed. Since there is at least a pair of variables x_j and x_z whose corresponding nodes are joined by an edge, there is still a set of constraints $K(x_j, x_z, y_1^i, \ldots, y_{r-1}^i)$, which is unsatisfiable.

The case in which some removed variables are from the clique and some from the nodes corresponding to the graph, and some from $y_1^1, \ldots, y_{r-1}^n$ is similar: since $h < n$, there is some i such that no variable from $y_1^i, \ldots, y_{r-1}^i$ is removed. Moreover, since we have removed less than h variables from $\{x_1, \ldots, x_n\}$, and the corresponding graph has some nodes, it holds that the set X' still contains a pair of variables x_j and x_z that correspond to nodes of the graph joined by an edge. We can thus repeat the proof used in the case above.

This proves that, for any given nontrivial set F of functions of arity greater or equal than two, INDEPENDENT SET can be polynomially reduced to MAX-IND-CSP(F). Consequently, the latter problem is NP-hard. □

If the domain is finite, we can also give a proof of membership to NP.

Theorem 12 MAX-IND-CSP(F) is in NP, if the domain is finite.

Proof: We show that a single non deterministic guess suffices to solve the problem. Given an instance, we guess the set of variables X' and the function S that assigns values to variables. We can verify in polynomial time whether all constraints in the subinstance induced by X' are satisfied. □

As a result, if the domain is finite, and the set of functions is nontrivial, the problem is NP-complete.

Corollary 13 MAX-IND-CSP(F) is NP-complete, if the domain is finite and F is nontrivial.

4 Compilability

In this section we study the possibility of reducing the complexity of MAX-IND-CSP(F) via compilation. This section is composed of two parts: in the

first we recall the basic definitions and results about compilability; in the second one we give our results about compilability of constraint satisfaction.

4.1 Preliminaries on Compilability

Since the MAX-IND-CSP problem is NP-hard for every nontrivial set of functions, we cannot expect to compute solutions to the general problem in polynomial time. However, it is still possible that slightly relaxed or modified versions of the problem can be solved efficiently. An observation that may lead to a reduction of the complexity is that hard problems often have a useful property: the input is divided in two parts, where one of them is known long before the rest of the input, or we may want to compute many instances having some part in common. In such cases, it is possible to take advantage of the structure of the problem, by doing some computation on the first part of the input only. Since either this part is known in advance, or it is shared by many instances to solve, it makes sense to spend more time on this phase. This is useful if the result of this preliminary phase can speed-up the process of solving the problem given the second part of the input.

The computation made in advance is called *preprocessing*, or *compilation*. The actual solving of the problem, given the result of this phase and the rest of the input is called *on-line processing*.

Of course, any algorithm can be seen as composed of a do-nothing compilation phase and a do-it-all on-line processing. In this case, nothing is actually gained from compilation. The rationale of compilation is actually that the on-line processing should be faster than the original algorithm thanks to compilation. If such efficiency improving is possible, compilation is useful, since it reduces the cost of solving the problem in an amortized sense.

We call the part of the input we preprocess *fixed part*, while the rest of the input is called *varying part*. When the complexity of the problem decreases thanks to the compilation of the fixed part, we call the problem *compilable*.

In order to formalize the concept of preprocessing, we recall the definitions of polysize functions, compilable classes and reductions. These concepts have been introduced in [Cadoli *et al.*, 1996], where more examples and motivations are provided. A function f is called *polysize* if there exists a polynomial p such that for all x it holds $||f(x)|| \leq p(||x||)$, that is, the *size* of the result of f is bounded by a polynomial in the size of x.

Our intuitive notion of compilability states that a problem S can be

reduced into a simpler problem by preprocessing its fixed part. Given a complexity class C, we introduce the class of problems compilable into C, denoted by compC. As an example, the class compP contains the problems that can be solved in polynomial time after a preprocessing phase.

Definition 14 *A language of pairs $S \subseteq \Sigma^* \times \Sigma^*$ belongs to compC if and only if there exist a polysize function f and a language of pairs S' such that for all $x, y \in \Sigma^*$ we have that*

1. $\langle x, y\rangle \in S$ *iff* $\langle f(x), y\rangle \in S'$.
2. $S' \in C$.

Notice that no restriction is imposed on the *time* needed to compute f, but only on the size of the result. This definition can be represented in terms of a computing machine as in Figure 3.

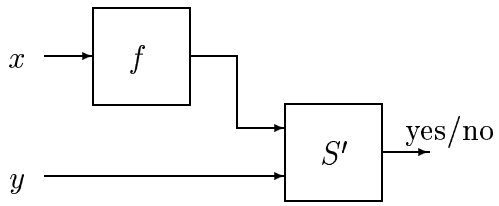


Figure 3: The compC machine.

In order to prove the non-compilability of problems, the concepts of nucomp reduction and hardness are defined.

Definition 15 *A nucomp reduction (non-uniform compilability reduction) between two problems of pairs A and B is a triple $\langle f_1, f_2, g\rangle$ such that f_1 and f_2 are polysize functions, g is a polynomial function, and for each pair $\langle x, y\rangle$ it holds*

$$\langle x, y\rangle \in A \text{ iff } \langle f_1(x, ||y||), g(f_2(x, ||y||), y)\rangle \in B$$

We say that A is *nucomp reducible* to B if there exists a nucomp reduction from A to B. The definition of hardness we include in this paper is a simplification of the original definition. See [Cadoli *et al.*, 1997] for details.

Definition 16 *A problem of pairs B is said to be nucompC-hard if any problem in compC is nucomp reducible to it B.*

Now, from [Cadoli *et al.*, 1996] it follows that a nucompNP-hard problem cannot be compiled into P, unless $\Pi_2^p = \Sigma_2^p$ (that is, unless the polynomial hierarchy collapses).

In many cases, nucompNP-hard problems are the variant (with fixed/varying part) of well-known NP-hard problems. For instance, let us consider the problem 3SAT, defined as follows.

Definition 17 *The problem* 3SAT *is defined as:*

INSTANCE: *a set* Π *of propositional clauses, each composed of three literals*

QUESTION: *is* Π *satisfiable?*

This problem can be easily converted into an "artificial" nucompNP-hard problem, that will be useful to prove the hardness of our problems.

Definition 18 *The problem* $*$3SAT *is defined as follows:*

INSTANCE: *a pair of strings* $\langle x, y \rangle$*, in which* y *denotes a* 3CNF *formula*

QUESTION: *is* y *satisfiable?*

In other words, a pair $\langle x, y \rangle$ is in $*$3SAT if y represents a satisfiable set of clauses, each composed of three literals, while x can be any string.

Theorem 19 The problem $*$3SAT is nucompNP-hard.

Proof: Let A be an arbitrarily chosen problem in compNP. We prove that A is nucomp reducible to $*$3SAT. Since A is in compNP, it follows that there exists a polysize function f and a polynomial function p such that

$$\langle x, y \rangle \in A \text{ iff } p(f(x), y) \in S$$

where S is a problem in NP. Since 3SAT is NP-hard, it follows that there exists a polynomial reduction f' from S to 3SAT:

$$z \in S \text{ iff } f'(z) \in \text{3SAT}$$

The nucomp reduction from A to $*3\text{SAT}$ is defined as $\langle f_1, f_2, g\rangle$, where

$$\begin{aligned} f_1(x,k) &= x \\ f_2(x,k) &= f(x) \\ g(z,y) &= f'(p(z,y)) \end{aligned}$$

We can now prove that $\langle x, y\rangle \in A$ if and only if $\langle f_1(x, ||y||), g(f_2(x, ||y||), y)\rangle \in *3\text{SAT}$:

$$\begin{aligned} \langle f_1(x, ||y||), g(f_2(x, ||y||), y)\rangle \in *3\text{SAT} \quad &\text{iff} \quad \langle x, g(f(x), y)\rangle \in *3\text{SAT} \\ &\text{iff} \quad \langle x, f'(p(f(x), y))\rangle \in *3\text{SAT} \\ &\text{iff} \quad f'(p(f(x), y)) \in 3\text{SAT} \\ &\text{iff} \quad p(f(x), y) \in S \\ &\text{iff} \quad \langle x, y\rangle \in A \end{aligned}$$

Since f_1 and f_2 are polysize (actually, they are computable in polynomial time) and g is polynomial (since f is), this is a nucomp reduction. □

The nucompNP hardness of a problem is a way of proving that a problem cannot be compiled into P. However, this requires to prove that there is a nucomp reduction from a previously proved nucompNP-hard problem, which can be a highly nontrivial task.

For this reason, the monotonic polynomial reductions have been introduced [Liberatore, 1998]. Let us assume that we want to prove the noncompilability to P of the problem S. Further assume that we have proven the NP-hardness of S by means of a polynomial reduction from the problem 3SAT. This means that, given an instance of 3SAT (i.e. a set of clauses, each composed of three literals), there exists exactly one instance of S associated with it. Now, each instance of S is composed of a fixed and a varying part. Let us denote r the function that gives the fixed part of S corresponding to a set of clauses, and h the function that gives the varying part. In other words, given a set of clauses Π, the corresponding instance of S has $r(\Pi)$ has fixed part, and $h(\Pi)$ has varying part. A polynomial reduction from an arbitrary problem to a problem with fixed and varying part can thus be seen as a pair of polynomial functions $\langle r, h\rangle$.

Definition 20 *A polynomial reduction $\langle r, h\rangle$ from 3SAT to a language of pairs S is called monotonic if, for any two sets of clauses Π_1 and Π_2 over*

the same alphabet, with $\Pi_1 \subseteq \Pi_2$*, it holds*

$$\langle r(\Pi_1), h(\Pi_1)\rangle \in S \;\Leftrightarrow\; \langle r(\Pi_2), h(\Pi_1)\rangle \in S \tag{1}$$

We can now prove that, given a monotonic polynomial reduction from 3SAT to S, one can build a proof of nucompNP hardness for the problem S.

Theorem 21 If there exists a monotonic polynomial reduction from 3SAT to a problem of pairs S, then S is nucompNP-hard.

Proof: Suppose there exists a monotonic polynomial reduction $\langle r, h\rangle$ from 3SAT to B. We prove that there exists a nucomp reduction f_1, f_2, g from the problem $*$3SAT (which is nucompNP-hard by Theorem 19) to S.

Let $Var(\Pi)$ be the number of atoms in Π: notice that $|Var(\Pi)| \leq ||\Pi||$. Furthermore, let Π_m denote the set of all the clauses of three literals over a set of variables $\{x_1, \ldots, x_m\}$.

Define f_1, f_2, g as follows.

$$\begin{aligned} f_1(s,m) &= r(\Pi_m) \\ f_2(s,m) &= \epsilon \\ g(a,\Pi) &= h(\Pi \cup \{x_{Var(\Pi)+1} \vee \neg x_{Var(\Pi)+1}, \ldots, x_{||\Pi||} \vee \neg x_{||\Pi||}\}) \end{aligned}$$

Let $\Pi' = \Pi \cup \{x_{Var(\Pi)+1} \vee \neg x_{Var(\Pi)+1}, \ldots, x_{||\Pi||} \vee \neg x_{||\Pi||}\}$. Using the fact that $\langle r, h\rangle$ is a monotonic polynomial reduction we have that

$$\begin{aligned} \langle x, \Pi\rangle \in *3\text{SAT} \quad &\text{iff} \quad \Pi \text{ is satisfiable} \\ &\text{iff} \quad \Pi' \text{ is satisfiable} \\ &\text{iff} \quad \langle r(\Pi'), h(\Pi')\rangle \in S \\ &\text{iff} \quad \langle r(\Pi_{Var(\Pi')}), h(\Pi')\rangle \in S \\ &\text{iff} \quad \langle r(\Pi_{||\Pi||}), h(\Pi')\rangle \in S \end{aligned}$$

But $r(\Pi_{||\Pi||}) = f_1(x, ||\Pi||)$ and $h(\Pi') = g(a, \Pi)$, thus $\langle x, \Pi\rangle \in *$3SAT if and only if $\langle f_1(x, ||\Pi||), g(f_2(x, ||\Pi||), \Pi)\rangle \in S$. □

4.2 Compilability of Constraint Satisfaction

As shown in the previous section, MAX-IND-CSP(F) is computationally hard for every set F of nontrivial functions. In this section, we investigate the possibility of compiling part of a MAX-IND-CSP(F) instance in order to obtain a faster on-line solving process.

We show that the MAX-IND-CSP(F) problem is nucompNP-hard when formalized as a fixed/varying part problem. The proof is composed of two parts: first, we show that a variant of INDEPENDENT SET (COMP-INDEP-SET) is nucompNP-hard by exhibiting a monotonic polynomial reduction from 3SAT. Then, we show that there exists a nucomp reduction from COMP-INDEP-SET to the compilation version of MAX-IND-CSP(F). Since nucomp reduction are transitive [Cadoli *et al.*, 1996, Cadoli *et al.*, 1997], the result follows.

First of all, we have to define the "compilable version" of INDEPENDENT SET and MAX-IND-CSP. Indeed, in order to determine the compilability of those problems, we have to specify which part of the instances is fixed and which part is varying. As a result, we have to re-define those problems specifying such a partition of the input.

Definition 22 *The* MAXIMAL INDUCED INDEPENDENT SET *problem is defined as:*

INSTANCE: *a pair* $\langle G, (V', k)\rangle$*, where* G *is a graph,* V' *is a subset of its nodes, and* k *is an integer. The fixed part is the graph* G*.*

QUESTION: *does* $G|V'$ *have an independent set of* k *nodes?*

Definition 23 *The problem* COMPILABLE MAXIMAL INDUCED CSP *is defined as:*

INSTANCE: *a pair* $\langle \Pi, \langle V', k\rangle\rangle$*, where* Π *is a CSP(F) instance,* V' *a subset of its variables, and* k *an integer. The fixed part is* Π*.*

QUESTION: *is there a solution with at least* k *variables for the instance* $\Pi|V'$*?*

We begin by proving that independent set is not compilable, unless the polynomial hierarchy collapses.

Theorem 24 INDEPENDENT SET is nucompNP-hard.

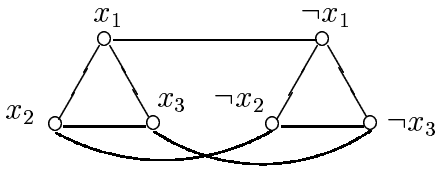
Proof: We prove that there exists a monotonic reduction from 3SAT to COMP-INDEP-SET. Let Π be a an arbitrary instance of 3SAT. For each clause, introduce a triplet of nodes in the graph. Each node of a triple is used to represent a literal of the corresponding clause. There is an edge between any two nodes of the same triplet, and also an edge between two nodes of different triplets if the two nodes represent opposite literals (that is, the first node is associated to a variable x_i and the second one is associated to its negation $\neg x_i$).

It has been proved [Papadimitriou, 1994] that a set of clauses of three literals is satisfiable if and only if the corresponding graph has an independent set of size equal to the number of clauses in Π. Thus, we can take the resulting graph to be the fixed part and $\langle V, k\rangle$ (where V are the nodes of the graph and k is the number of clauses in Π) to be the varying part of the COMP-INDEP-SET instance.

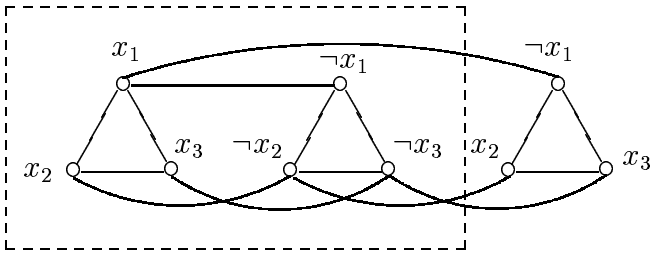
We continue by showing that this reduction is monotonic. Let Π_1 and Π_2 be two sets of clauses (of length 3) over the same alphabet and assume that $\Pi_1 \subseteq \Pi_2$. Let $r(\Pi_i)$, with $1 \leq i \leq 2$, denote the graph associated with the clauses in Π_i. Let V be the set of nodes of $r(\Pi_1)$ and let k be the number of clauses in Π_1.

To prove that the above stated reduction is monotonic, we have to show that $r(\Pi_1)|V$ contains an independent set of size k if and only if $r(\Pi_2)|V$ contains an independent set of the same size. However, this is easy by noting that $r(\Pi_2)|V = r(\Pi_1)|V = r(\Pi_1)$ by the construction of $r(\Pi_2)$. □

The following picture shows the graph associated to the set of clauses $\{x_1 \vee x_2 \vee x_3, \neg x_1 \vee \neg x_2 \vee \neg x_3\}$.



Let now show what is the graph associated to a larger set of clauses, for instance $\{x_1 \vee x_2 \vee x_3, \neg x_1 \vee \neg x_2 \vee \neg x_3, \neg x_1 \vee x_2 \vee x_3\}$.



The dashed box encloses exactly the nodes of V'. This picture makes it clear that the graph enclosed in the box is exactly the previous one.

Now we prove that the problem of finding a maximal satisfiable set of variables is nucompNP-complete. This is done by showing that the reduction used in Theorem 11 is indeed a nucomp reductions.

Theorem 25 COMP-MAX-IND-CSP(F) is nucompNP-hard, for any nontrivial set of functions of arbitrary arity greater or equal than two.

Proof: We show that the reduction from INDEPENDENT SET to MAX-IND-CSP given in Theorem 11 is a nucomp reduction. Let us consider how the reduction of Theorem 11 works. The instance of INDEPENDENT SET is composed of a graph G, a subset of nodes V' and an integer k. The corresponding instance of COMP-MAX-IND-CSP is as follows: there is one variable for each node of the graph, plus a number of other nodes. There are some constraints over the variables, depending on the edges of the graph. This part of the instance can thus be determined knowing G only, so there is a polysize function f_1 that gives the set of constraints corresponding to a graph G.

The varying part of the constraint satisfaction problem (the set of variables and the integer) can be determined in polynomial time. Let $f_2(G, n) = G$ and $g(G, \langle V', k\rangle)$ be the function that takes G, V', and k and produces the varying part of the constraint satisfaction instance. Clearly, $\langle f_1, f_2, g\rangle$ is a nucomp reduction, since f_1 and f_2 are polysize and g is polynomial. □

If the domain is finite, COMP-MAX-IND-CSP(F) is in NP, thus is also in nucompNP.

Corollary 26 COMP-MAX-IND-CSP(F) is nucompNP-complete, for any nontrivial set of functions of arbitrary arity greater or equal than two, if the domain is finite.

5 Approximability

In this section, we study the approximability of MAX-IND-CSP. We have earlier completely classified when the decision problem is polynomial and NP-hard, respectively, and shown that certain compilation techniques cannot simplify the problem. Unfortunately, we are not able to give such a general result about the approximability of MAX-IND-CSP. However, a strong dichotomy holds in two special cases:

- when all functions have arity 2; and
- when the intersection of the functions equals 0 for all possible arguments.

In these cases, the problem is either trivial or cannot be approximated within n^δ for some $\delta > 0$, unless $P = NP$. In the second case, we also show that the problem is not approximable within $n^{1-\epsilon}$ for any $\epsilon > 0$, unless $ZPP = NP$.

We begin by providing some definitions and results concerning approximability. Solving a maximization problem A given the input instance x means finding a solution y such that the value of the objective function $m_A(x, y)$ is maximum. Let the *optimal value function* opt_A return the optimal value of m_A for arbitrary instances x of A.

Given a maximization problem A and a function $\alpha : \mathbb{N} \to (1, \infty)$, we say that a polynomial-time algorithm P is an *α-approximation algorithm* for A iff for every instance x of A of size n, P produces a solution in the range $[opt_A(x)/\alpha(n), opt_A(x)]$. We say that A is *approximable within a factor α* if such an algorithm exists. We measure instances as follows: the size of a graph equals the number of nodes in it and the size of a CSP instance is given by the number of variables.

As our starting point, we use the problem MAXIMUM SUBGRAPH PROBLEM FOR PROPERTY π, denoted by MSP(π).

Definition 27 *The* MAXIMUM SUBGRAPH PROBLEM FOR PROPERTY π *problem (denoted by* MSP(π)*) is the problem defined as:*
INSTANCE: *Undirected graph $G = \langle V, E \rangle$, positive integer $k \leq |V|$.*
QUESTION: *Is there a subset $V' \subseteq V$ with $|V'| \geq k$ such that the graph $G|V'$ has property π?*

A property π of graphs is said to be *hereditary* iff it holds for all induced subgraphs of G whenever it holds for G. The property π is *nontrivial* iff π holds for infinitely many graphs and does not hold for infinitely many graphs.

Theorem 28 [Lund and Yannakakis, 1993] There exists an $\delta > 0$ such that $\text{MSP}(\pi)$ cannot be approximated with ratio n^δ for any nontrivial hereditary property that is false for some clique, unless P = NP.

Given an arbitrary instance Π of $\text{CSP}(\{f\})$ (where f is symmetric), we define the graph G_Π as follows: each node in G_Π corresponds to a variable in Π and an edge joins nodes v, w iff Π contains the constraint (f, v, w) and/or (f, w, v). Since f is symmetric, the exact order of the arguments does not matter.

Lemma 29 Let $f : D^2 \to \{0, 1\}$ be a symmetric function such that $\{f\}$ is nontrivial. Define $\mathcal{F}$ to be the set

$$\{G_\Pi \mid \Pi \text{ is a satisfiable instance of } \text{CSP}(\{f\})\}.$$

and let the graph property π hold for a graph G iff $G \in \mathcal{F}$. Then,

1. π is hereditary;
2. π is nontrivial; and
3. there exists a clique not having property π.

Proof:

1. π is hereditary since if $\Psi = \langle V, C \rangle$ is a satisfiable instance of $\text{CSP}(\{f\})$, then $\Psi|V'$ is satisfiable for every choice of $V' \subseteq V$.
2. To see that π is nontrivial, first note that any completely disconnected graph has property π since the domain D is required to be nonempty. Thus, π holds for infinitely many graphs. Continue by arbitrarily choosing an unsatisfiable instance Σ of $\text{CSP}(\{f\})$ and assume that Σ contains c variables. Let Σ' be an instance of $\text{CSP}(\{f\})$ containing k variables and satisfying $K_c = G_{\Sigma'}$. Clearly, Σ' is not satisfiable since Σ is not satisfiable. Consequently, $K_k \notin \mathcal{F}$ for any $k \geq c$ and there exists infinitely many graphs not having property π.

3. The clique K_c does not have property π, as was shown above.

□

From now on, let $F = \{f_1, \ldots, f_n\}$ be a nontrivial set of functions from D^2 to $\{0,1\}$. Let $f : D^2 \to \{0,1\}$ denote the intersection of the functions in F.

Theorem 30 MAX-IND-CSP(F) cannot be approximated within n^δ for some $\delta > 0$, unless $P = NP$. Furthermore, if $f(x,y) = 0$ for all $x, y \in D$, then

1. MAX-IND-CSP(F) cannot be approximated within $n^{1-\epsilon}$ for any $\epsilon > 0$, unless $ZPP = NP$; and
2. MAX-IND-CSP(F) cannot be approximated within $n^{1/2-\epsilon}$ for any $\epsilon > 0$, unless $P = NP$.

Proof: By definition, f is symmetric. Define $\mathcal{F}$ to be the set

$$\{G_\Pi \mid \Pi \text{ is a satisfiable instance of CSP}(\{f\})$$

and let the graph property π hold for a graph G iff $G \in \mathcal{F}$. By Lemma 29, π is hereditary and nontrivial and there exists a clique not having property π.

We show nonapproximability of MAX-IND-CSP(F) by a polynomial-time cost-preserving reduction from MSP(π). Let $G = \langle V, E \rangle$ be an arbitrary graph. Construct an instance Π of MAX-IND-CSP(F) as follows: For each $v \in V$, introduce a variable x. For each edge $(u,v) \in E$, introduce the constraint:

$$\{(f, x, y) \mid 1 \leq i \leq n\}$$

where x and y are the variables corresponding to u and v, respectively. As we have noted earlier, this transformation can be carried out in polynomial time. Furthermore, the size of the maximum satisfiable induced CSP of Π equals the size of the maximum induced subgraph of G having property π. By Theorem 28, this number cannot be approximated within n^δ for some δ.

Assume that $f(x,y) = 0$ for all $x, y \in D$. Then, the size of the maximum induced satisfiable CSP of Π equals the size of the largest independent set of G. Håstad [1996] has shown that the size of the independent set cannot be

approximated within $n^{1-\epsilon}$ for any $\epsilon > 0$, unless $ZPP = NP$, and not within $n^{1/2-\epsilon}$, unless $P = NP$. □

In the proof of Theorem 30, we used the nonapproximability of computing the size of independents sets for showing nonapproximability of certain MAX-IND-CSP(F) problems where F only contains binary functions. This idea can be extended to functions of higher arity by considering independent sets in hypergraphs. A *hypergraph* $\mathcal{H} = \langle V, \mathcal{E} \rangle$ is defined by a set V of *vertices* and a set $\mathcal{E}$ of *hyperedges* where $E \subseteq V$ for every $E \in \mathcal{E}$. A hypergraph is called *k-uniform* if $|E| = k$ for every $E \in \mathcal{E}$. A subset $I \subseteq V$ is said to be independent if I contains no hyperedges from $\mathcal{H}$, that is, $E \not\subseteq I$ for any $E \in \mathcal{E}$.

Hofmeister and Lefmann [1998] have shown the following result on the approximability of independent sets in hypergraphs.

Theorem 31 Let $k \geq 2$ be a fixed integer. The problem of determining the size of the largest independent set in k-uniform hypergraphs cannot be approximated within $n^{1-\epsilon}$ for any $\epsilon > 0$, unless $ZPP = NP$.

Let $F = \{f_1, \ldots, f_n\}$ be a nontrivial set of functions of arity ≥ 2 and let f denote the intersection of F. Let a be the arity of f.

Theorem 32 If $f(d_1, \ldots, d_a) = 0$ for all $d_1, \ldots, d_a \in D$, then MAX-IND-CSP(F) cannot be approximated within $n^{1-\epsilon}$ for any $\epsilon > 0$, unless $ZPP = NP$.

Proof: Arbitrarily choose an a-uniform hypergraph $\mathcal{H} = \langle V, \mathcal{E} \rangle$. Construct an instance of MAX-IND-CSP($\{f\}$) as follows: for each vertex v_k, introduce a variable v'_k and for each edge $\{v_{i_1}, \ldots, v_{i_a}\}$, introduce the constraint $(f, v'_{i_1}, \ldots, v'_{i_a})$. This transformation can be carried out in polynomial time since a is fixed.

The size of the maximum induced satisfiable CSP of the resulting instance equals the size of the largest independent set of $\mathcal{H}$. Thus, nonapproximability of MAX-IND-CSP(F) follows from Theorem 31. □

6 Conclusions

In this paper we studied the complexity of problems related to constraint satisfaction. Since a set of constraints may be unsatisfiable, we considered the problem of finding satisfiable subinstance of the problem. This is in line with the aim of satisfying as many constraints as possible.

In the case of constraints with arity greater than or equal to two, we gave a complete classification of the problem w.r.t. tractability. Indeed, we proved that, for any set of functions, either the problem is trivial (i.e. always satisfiable) or it is NP-hard.

Of course, the trivial case is not very interesting. As a result, we ended up with a NP-complete problem, thus intractable. In order to allow for fast solving, we considered two possible approaches for solving intractable problems: compilation and approximation.

With respect to compilation, we proved that the problem does not gain from compilation: the problem remains hard to solve even if we allow a time-consuming compilation phase. We also studied the approximability of the problem and found that it is hard to approximate in several cases.

At least two problems are left open by this work. As said in the beginning of this paper, we are concerned about problems of constraint satisfaction in which all constraints have arity greater or equal than two. This means that our results do not hold if unary constraints are present.

As an example, consider the domain $D = \{0, 1, 2\}$, and F being composed of two unary functions:

$$\begin{aligned} f_1(x) &= 1 \text{ iff } x = 0 \\ f_2(x) &= 1 \text{ iff } x > 0 \end{aligned}$$

Clearly, there are unsatisfiable instances using only these two functions, for instance $\{(f_1, x),\ (f_2, x)\}$. We proved that, in the case of binary functions, the existence of unsatisfiability instances implies the NP-hardness of the problem. This result cannot be extended to unary functions, as checking satisfiability of instances using f_1 and f_2 is polynomial, as it amounts to check whether there are two constraints having f_1 and f_2 as function and the same variable as argument. This can be generalized: checking satisfiability of instances using only unary functions is always a polynomial task. Finding maximal satisfiable subinstances is also polynomial.

The complexity of problems with both unary and binary constraints is harder to determine. The fact that unary functions alone are polynomial does not imply that we can neglect them. Let F be defined as $F = \{f_1, f_2, f_3\}$, where f_3 is defined as:

$$f_3(x, y) = \begin{cases} 1 & \text{iff } x = y = 0 \\ 0 & \text{otherwise} \end{cases}$$

Any instance using f_3 only is satisfiable, thus the problem is polynomial. However, using all three functions leads to NP-completeness. Indeed, intersecting the functions f_1 and f_3 we obtain a function that is never satisfiable, which implies that the problem of satisfiability and maximal induced satisfiable subinstance are NP-complete.

Another open question of this paper is whether the problem of maximal satisfiable induced subinstance is approximable in the case in which functions with arity greater than two are allowed. Indeed, we proved that the problem is not approximable only in the special case in which the intersection of all functions is always equal to 0. This case is useful: it is hard to imagine a set of meaningful constraints that are all satisfied by the same value. However, it would obviously be better to have a complete classification.

Acknowledgments

The first author was sponsored by the *Swdish Research Council for the Engineering Sciences* (TFR) under grant 97-301.

This work has been done while the second author was visiting Linköping University. He thanks Erik Sandewall for his hospitality and support.

References

[Cadoli *et al.*, 1996] M. Cadoli, F. M. Donini, P. Liberatore, and M. Schaerf. Feasibility and unfeasibility of off-line processing. In *Proceedings of the Fourth Israeli Symposium on Theory of Computing and Systems (ISTCS'96)*, pages 100–109. IEEE Computer Society Press, 1996. URL = `ftp://ftp.dis.uniroma1.it/PUB/AI/papers/cado-etal-96.ps.gz`.

[Cadoli *et al.*, 1997] M. Cadoli, F. M. Donini, P. Liberatore, and M. Schaerf. Preprocessing of intractable problems. Technical Report DIS 24-97, Dipartimento di Informatica e Sistemistica, Università di Roma "La Sapienza", November 1997. URL = `http://www.dis.uniroma1.it/PUB/AI/papers/cado-etal-97-d.ps.gz`.

[Cooper *et al.*, 1994] M. C. Cooper, D. A. Cohen, and P. G. Jeavons. Characterizing tractable constraints. *Artificial Intelligence*, 65:347–361, 1994.

[Creignou and Hermann, 1996] N. Creignou and M. Hermann. Complexity of generalized satisfiability counting problems. *Information and Computation*, 125:1–12, 1996.

[Creignou, 1995] N. Creignou. A dichotomy theorem for maximum generalized satisfiability problems. *Journal of Computer and System Science*, 51(3):511–522, 1995.

[Feder and Vardi, 1998] T. Feder and M. Y. Vardi. The computational structure of monotone monadic SNP and constraint satisfaction: a study through datalog and group theory. *SIAM Journal of Computing*, 28(1):57–104, 1998.

[Håstad, 1996] J. Håstad. Clique is hard to approximate within $n^{1-\epsilon}$. In *Proceedings of the 37th IEEE Symposium on Foundations of Computer Science*, pages 627–636, 1996.

[Hell and Nešetřil, 1990] P. Hell and N. Nešetřil. On the complexity of H-coloring. *Journal of Combinatorial Theory, ser. B*, 48:92–110, 1990.

[Hofmeister and Lefmann, 1998] T. Hofmeister and H. Lefmann. Approximating maximum independent sets in uniform hypergraphs. In *Proceedings of the 23rd IEEE International Symposium on Mathematical Foundations of Computer Science*, pages 562–570, 1998.

[Jeavons and Cooper, 1996] P. G. Jeavons and M. C. Cooper. Tractable constraints on ordered domains. *Artificial Intelligence*, 79:327–339, 1996.

[Jeavons, 1998] P. G. Jeavons. On the algebraic structure of combinatorial problems. *Theoretical Computer Science*, 200(1–2):185–204, 1998.

[Jonsson, 1999] P. Jonsson. Boolean constraint satisfaction: complexity results for optimization problems with arbitrary weights. *Theoretical Computer Science*, 1999. To appear.

[Karp and Lipton, 1980] R. M. Karp and R. J. Lipton. Some connections between non-uniform and uniform complexity classes. In *Proceedings of the Twelfth ACM Symposium on Theory of Computing (STOC'80)*, pages 302–309, 1980.

[Kavvadias and Sideri, 1998] D. Kavvadias and M. Sideri. The inverse satisfiability problem. *SIAM Journal of Computing*, 28(1):152–163, 1998.

[Khanna *et al.*, 1997a] S. Khanna, M. Sudan, and L. Trevisan. Constraint satisfaction: The approximability of minimization problems. In *Proceedings of the 12th Annual IEEE Conference on Computational Complexity*, pages 282–296, 1997.

[Khanna *et al.*, 1997b] S. Khanna, M. Sudan, and D. P. Williamson. A complete classification of the approximability of maximization problems derived from boolean constraint satisfaction. In *Proceedings of the 29th ACM Symposium on Theory of Computing*, pages 11–20, 1997.

[Liberatore, 1998] P. Liberatore. On the compilability of diagnosis, planning, reasoning about actions, belief revision, etc. In *Proceedings of the Sixth International Conference on Principles of Knowledge Representation and Reasoning (KR'98)*, pages 144–155, 1998.

[Lund and Yannakakis, 1993] C. Lund and M. Yannakakis. The approximation of maximum subgraph problems. In *Proceedings of the 20th international colloquium on automata, languages and programming*, pages 40–51. Springer-Verlag, 1993. Lecture notes in computer science. 700.

[Papadimitriou, 1994] C. H. Papadimitriou. *Computational Complexity.* Addison Wesley, Reading, MA, 1994.

[Preparata and Shamos, 1985] F. P. Preparata and M. I. Shamos. *Computational Geometry: An introduction.* Springer-Verlag, 1985.

[Reith and Vollmer, 1998] S. Reith and H. Vollmer. The complexity of computing optimal assignments of generalized propositional formulae. Technical Report 22, Electronic Colloquium on Computational Complexity, 1998.

[Schaefer, 1978] T. J. Schaefer. The complexity of satisfiability problems. In *Proceedings of the 10th ACM Symposium on Theory of Computing*, pages 216–226, 1978.

ECCC ISSN 1433-8092
http://www.eccc.uni-trier.de/eccc
ftp://ftp.eccc.uni-trier.de/pub/eccc
ftpmail@ftp.eccc.uni-trier.de, subject 'help eccc'