Electronic Colloquium on Computational Complexity, Report No. 34 (2014)

On the complexity of trial and error for constraint satisfaction problems

Gábor Ivanyos* Raghav Kulkarni† Youming Qiao‡ Miklos Santha§
Aarthi Sundaram¶

Abstract

In a recent work of Bei, Chen and Zhang (STOC 2013), a trial and error model of computing was introduced, and applied to some constraint satisfaction problems. In this model the input is hidden by an oracle which, for a candidate assignment, reveals some information about a violated constraint if the assignment is not satisfying. In this paper we initiate a *systematic* study of constraint satisfaction problems in the trial and error model. To achieve this, we first adopt a formal framework for CSPs, and based on this framework we define several types of revealing oracles. Our main contribution is to develop a *transfer theorem* for each type of the revealing oracle, under a broad class of parameters. To any hidden CSP with a specific type of revealing oracle, the transfer theorem associates another, potentially harder CSP in the normal setting, such that their complexities are polynomial time equivalent. This in principle transfers the study of a large class of hidden CSPs, possibly with a promise on the instances, to the study of CSPs in the normal setting. We then apply the transfer theorems to get polynomial-time algorithms or hardness results for hidden CSPs, including satisfaction problems, monotone graph properties, isomorphism problems, and the exact version of the Unique Games problem. Most of the proofs of these results are short and straightforward, which exhibits the power of the transfer theorems.

*Institute for Computer Science and Control, Hungarian Academy of Sciences, Budapest, Hungary (Gabor.Ivanyos@sztaki.mta.hu).

†Centre for Quantum Technologies, National University of Singapore, Singapore 117543 (kulraghav@gmail.com).

‡Centre for Quantum Technologies, National University of Singapore, Singapore 117543 (cqtqy@nus.edu.sg).

§LIAFA, Univ. Paris 7, CNRS, 75205 Paris, France; and Centre for Quantum Technologies, National University of Singapore, Singapore 117543 (miklos.santha@liafa.jussieu.fr).

¶Centre for Quantum Technologies, National University of Singapore, Singapore 117543 (aarthims@nus.edu.sg).

ISSN 1433-8092

1 Introduction

In [BCZ13a], Bei, Chen and Zhang proposed a *trial and error* model to study algorithmic problems when some input information is lacking. As argued in their paper, the lack of input information can happen when we have only limited knowledge of and access to the problem. They also described several realistic scenarios where the inputs are actually unknown. Then, they formalized this methodology in the complexity-theoretic setting, and proposed a trial and error model for constraint satisfaction problems. They further applied this idea to investigate the information needed to solve linear programming in [BCZ13b], and to study information diffusion in a social network in [BCD+13].

As mentioned, in [BCZ13a] the authors focused on the hidden versions of some specific constraint satisfaction problems (H–CSPs), whose instances could only be accessed via a *revealing* oracle. An algorithm in this setting interacts with this revealing oracle to get information about the input instance. Each time, the algorithm proposes a candidate solution, a *trial*, and the validity of this trial is checked by the oracle. If the trial succeeds, the algorithm is notified that the proposed trial is already a solution. Otherwise, the algorithm obtains as an *error*, a violation of some property corresponding to the instance. The algorithm aims to make effective use of these errors to propose new trials, and the goal is to minimize the number of trials while keeping in mind the cost for proposing new trials. When the CSP is already difficult, a computation oracle that solves the original problem might be allowed. Its use is justified as we are interested in the *extra difficulty* caused by the lack of information. Bei, Chen and Zhang considered several natural CSPs in the trial and error setting, including SAT, Stable Matching, Graph Isomorphism and Group Isomorphism. While the former two problems in the hidden setting are shown to be of the same difficulty as in the normal one, the last two cases have substantially increased complexities in the unknown-input model. They also studied more problems, as well as various aspects of this model, like the query complexity.

In this paper, following [BCZ13a], we initiate a *systematic* study of the constraint satisfaction problems in the trial and error model. To achieve this, we first adopt a formal framework for CSPs, and based on this framework we define three types of revealing oracles. This framework also helps to clarify and enrich the model of [BCZ13a]. Our main contribution is to develop a *transfer theorem* for each type of the revealing oracle, under a broad class of parameters. For any hidden CSP with a specific type of revealing oracle, the transfer theorem associates another CSP in the normal (unhidden) setting, such that their difficulties are roughly the same. This in principle transfers the study of hidden CSPs to the study of CSPs in the normal setting. We also apply transfer theorems to get results for concrete CSPs, including some problems considered in [BCZ13a], for which we usually get much shorter and easier proofs.

The framework for CSPs, and hidden CSPs. To state our results we describe informally the framework of CSPs. A CSP S is defined by a finite alphabet $[\![w]\!] = \{0, 1, \ldots, w-1\}$ and by $\mathcal{R} = \{R_1, \ldots, R_s\}$, a set of relations over $[\![w]\!]$ of some fixed arity q. For a set of variables $\mathcal{V} = \{x_1, \ldots, x_\ell\}$, an instance of S is a set of constraints $\mathcal{C} = \{C_1, \ldots, C_m\}$, where $C_j = R(x_{j_1}, \ldots, x_{j_q})$ for some relation $R \in \mathcal{R}$ and some q-tuple of variables. An assignment $a \in [\![w]\!]^\ell$ satisfies $\mathcal{C}$ if it satisfies every constraint in it.

Example 1.1. 1SAT: Here $w = 2$, $q = 1$, and $\mathcal{R} = \{\mathsf{Id}, \mathsf{Neg}\}$, where $\mathsf{Id} = \{1\}$ is the identity relation, and $\mathsf{Neg} = \{0\}$ is its complement. Thus a constraint is a literal x_i or $\bar{x}_i$, and an instance is just a collection of literals. *In case of* 3SAT *the parameters are* $w = 2$, $q = 3$ *and* $|\mathcal{R}| = 8$. *We will keep for*

further illustrations 1SAT *which is a problem in polynomial time.* 3SAT *would be a less illustrative example since the standard problem is already* NP*-complete.*

To allow for more versatility, we often consider some *promise* $W \subseteq [\![w]\!]^\ell$ on the assignments, and only look for a satisfying assignment within this promise. This case happens, say when we look for permutations in isomorphism problems.

Recall that in the hidden setting, the algorithm interacts with some revealing oracle by repeatedly proposing assignments. If the proposed assignment is not satisfying then the revealing oracle discloses certain information about some violated constraint. This can be in principle an index of such a constraint, (the index of) the relation in it, the indices of the variables where this relation is applied, or any subset of the above. Here we will require that the oracle always reveals the index of a violated constraint from $\mathcal{C}$. To characterize the choices for the additional information, for any subset $\mathcal{U} \subseteq \{\mathcal{R}, \mathcal{V}\}$ we say that an oracle is $\mathcal{U}$*-revealing* if it also gives out the information corresponding to $\mathcal{U}$. For a CSP problem S we use H–$\mathrm{S}_{\mathcal{U}}$ to denote the corresponding hidden problem in the trial and error model with $\mathcal{U}$-revealing oracle.

Example 1.1 continued. Let us suppose that we present an assignment $a \in \{0,1\}^\ell$ for an instance of the hidden version H–$\mathsf{1SAT}_{\mathcal{U}}$ of 1SAT to the $\mathcal{U}$-revealing oracle. If $\mathcal{U} = \{\mathcal{V}\}$ and the oracle reveals j and i respectively for the violated constraint and the variable in it then we learn that the jth literal is x_i if $a_i = 0$, and $\bar{x}_i$ otherwise. If $\mathcal{U} = \{\mathcal{R}\}$ and say the oracle reveals j and Id then we learn that the jth literal is positive. If $\mathcal{U} = \varnothing$ and the oracle reveals j then we only learn that the jth literal is either a positive literal corresponding to one of the indices where a is 0, or a negative literal corresponding to an index where a is 1.

In order to explain the transfer theorem and motivate the operations which create richer CSPs, we first make a simple observation that H–$\mathrm{S}_{\{\mathcal{R},\mathcal{V}\}}$ and S are polynomial time equivalent, when the relations of S are in P. Indeed, an algorithm for H–$\mathrm{S}_{\{\mathcal{R},\mathcal{V}\}}$ can solve S, as the answers of the oracle can be given by directly checking if the proposed assignment is satisfying. In the other direction, we repeatedly submit assignments to the oracle. The answer of the oracle fully reveals a (violated) constraint. Given some subset of constraints we already know, to find a new constraint, we submit an assignment which satisfies all the known constraints. Such an assignment can be found by the algorithm for S.

With a weaker oracle this procedure clearly does not work and to compensate, we need stronger CSPs. In the case of $\{\mathcal{V}\}$-revealing oracles an answer helps us exclude, for the specified clause, all those relations which were violated at the specified indices of the proposed assignment, but keep as possibilities all the relations which were satisfied at those indices. Therefore, to find out more information about the input, we would like to find a satisfying assignment for a CSP instance whose corresponding constraint is the union of the satisfied relations. This naturally brings us to consider the CSP $\bigcup$S, the *closure by union* of S whose relations are from $\bigcup\mathcal{R}$, the *closure by union* of $\mathcal{R}$, which contains relations by taking union over any subset of $\mathcal{R}$.

The situation with the $\{\mathcal{R}\}$-revealing oracle is analogous, but here we have to compensate, in the stronger CSP, for the lack of revealed information about the variable indices. For a relation R and q-tuple of indices $(j_1, \ldots, j_q)$, we define the ℓ-ary relation $R^{(j_1,\ldots,j_q)} = \{a \in W \,:\, (a_{j_1}, \ldots, a_{j_q}) \in R\}$, and for a set I of q-tuples of indices, we set $R^I = \bigcup_{(j_1,\ldots,j_q)\in I} R^{(j_1,\ldots,j_q)}$. The *arity extension* of S is the constraint satisfaction problem E–S whose relations are from arity extension E–$\mathcal{R}$ $= \bigcup_I \{R^I \,:\, R \in \mathcal{R}\}$ of $\mathcal{R}$.

The transfer theorem first says that with $\bigcup$S (resp. E–S) we can compensate the information

hidden by a $\{\mathcal{V}\}$-revealing (resp. $\{\mathcal{R}\}$-revealing) oracle, that is we can solve H–$\mathrm{S}_{\{\mathcal{V}\}}$ (resp. H–$\mathrm{S}_{\{\mathcal{R}\}}$). In fact, with $\bigcup$E–S we can solve H–$\mathrm{S}_{\varnothing}$. Moreover, perhaps more surprisingly, it says that these statements also hold in the reverse direction: if we can solve the hidden CSP, we can also solve the corresponding extended CSP.

Transfer Theorem (informal statement) *Let* S *be a CSP whose parameters are "reasonable" and whose relations are in* P*. Then for any promise W on the assignments, the complexities of the following problems are polynomial time equivalent: (a)* H–$\mathrm{S}_{\{\mathcal{V}\}}$ *and* $\bigcup$S*, (b)* H–$\mathrm{S}_{\{\mathcal{R}\}}$ *and* E–S*, (c)* H–$\mathrm{S}_{\varnothing}$ *and* $\bigcup$E–S*.*

The precise dependence on the parameters can be found in the theorems of Section 3 and Corollary 3.5 highlights the conditions for polynomial equivalence. **Example 1.1 continued.** Since $\bigcup\{\mathsf{Id}, \mathsf{Neg}\} = \{\varnothing, \mathsf{Id}, \mathsf{Neg}, \{0,1\}\}$, $\bigcup$1SAT has only the two trivial (always false or always true) relations in addition to the relations in 1SAT. Therefore it can be solved in polynomial time, and by the the Transfer Theorem H–$\mathsf{1SAT}_{\{\mathcal{V}\}}$ is also in P. On the other hand, for any index set $I \subseteq [\ell]$, Id^I is a disjunct of positive literals with variables from I, and similarly Neg^I is a disjunct of negative literals with variables from I. Thus E–1SAT includes MONSAT, which consists of those instances of SAT where in each clause either every variable is positive, or every variable is negated. The problem MONSAT is NP-hard by Schaefer's characterization [Sch78], and therefore the Transfer Theorem implies that H–$\mathsf{1SAT}_{\{\mathcal{R}\}}$ and H–$\mathsf{1SAT}_{\varnothing}$ are also NP-hard.

In a further generalization, we will also consider CSPs and H–CSPs whose instances satisfy some property. One such property can be *repetition freeness* meaning that the constraints of an instance are pairwise distinct. The promise H–CSPs could also be a suitable framework for discussing certain graph problems on special classes of graphs. For a promise PROM on instances of S we denote by $\mathrm{S}^{\mathsf{PROM}}$ the promise problem whose instances are instances of S satisfying PROM. The problem H–$\mathrm{S}^{\mathsf{PROM}}_{\{\mathcal{U}\}}$ is defined in an analogous way from H–$\mathrm{S}_{\{\mathcal{U}\}}$.

It turns out that we can generalize the Transfer Theorem for CSPs with promises on the instances. We describe this in broad lines for the case of $\{\mathcal{V}\}$-revealing oracles. Given a promise PROM on S, the corresponding promise $\bigcup$PROM for $\bigcup$S is defined in a natural way. We say that a $\bigcup$S-instance $\mathcal{C}'$ *includes* an S-instance $\mathcal{C}$ if for every $j \in [m]$, the constraint C'_j in $\mathcal{C}'$ and the constraint C_j in $\mathcal{C}$ are defined on the same variables, and seen as relations, $C_j \subseteq C'_j$. Then $\bigcup$PROM is the set of instances $\mathcal{C}'$ of $\bigcup$S which include some $\mathcal{C} \in$ PROM. The concept of an algorithm *solving* $\bigcup \mathrm{S}^{\bigcup\mathsf{PROM}}$ has to be relaxed: while we search for a satisfying assignment for those instances which include a satisfiable instance of PROM, when this is not the case, the algorithm can abort even if the instance is satisfiable. With this we have:

Transfer Theorem for promise problems (informal statement) *Let* S *be a constraint satisfaction problem with promise* PROM*. Then the complexities of* H–$\mathrm{S}^{\mathsf{PROM}}_{\{\mathcal{V}\}}$ *and* $\bigcup S^{\bigcup\mathsf{PROM}}$ *are polynomial time equivalent when the parameters are "reasonable" and the relations of* S *are in* P*.*

Example 1.1 continued. Let RF denote the property of being repetition free, in the case of 1SAT this just means that no literal can appear twice in the formula. Then H–$\mathsf{1SAT}^{\mathsf{RF}}_{\varnothing}$, hidden repetition-free 1SAT with $\varnothing$-revealing oracle, is solved in polynomial time. To see this we first consider X–1SAT, the constraint satisfaction problem whose relations are all ℓ-ary extensions of Id and Neg. (See Section 2 for a formal definition.) It is quite easy to see that hidden 1SAT with $\varnothing$-revealing oracle is essentially the same problem as hidden X–1SAT with $\{\mathcal{V}\}$-revealing oracle. Therefore, by the Transfer Theorem we are concerned with $\bigcup$X–1SAT with promise $\bigcup$RF. The instances satisfying the promise are $\{C_1, \ldots, C_m\}$, where C_j is a disjunction of literals such that

there exist distinct literals $z_1, \ldots, z_m$, with $z_j \in C_j$. It turns out that these specific instances of SAT can be solved in polynomial time. The basic idea is that we can apply a maximum matching algorithm, and only output a solution if we can select m pairwise different variables $x_{i_1}, \ldots, x_{i_m}$ such that either x_{i_j} or $\overline{x}_{i_j}$ is in C_j.

Applications of transfer theorems. Since NP-hard problems obviously remain NP-hard in the hidden setting (without access to an NP oracle), we investigate the complexity of various polynomial-time solvable CSPs. We first apply the Transfer Theorem when there is no promise on the instances. We categorize the hidden CSPs depending on the type of the revealing oracle.

With constraint index revealing oracles, we focus on various monotone graph properties like Spanning Tree, Cycle Cover, etc.. We define a general framework to represent monotone graph property problems as H–CSPs and show that they become NP-hard. This framework also naturally extends to directed graphs.

With constraint and variable index revealing oracles, we obtain results on several interesting families of CSPs including the exact-Unique Games Problem (cf. Section 5), equality to a member of a fixed class of graphs, and graph properties as discussed above. Interestingly, many of the graph properties mentioned in the last paragraph are no longer NP-hard but are in P, as well as some other CSPs like 2SAT and the exact-Unique Game problem on alphabet size 2. Still, there are some NP-hard CSPs, like the exact-Unique Game problem on alphabet size ≥ 3, and equality to some specific graph, such as k-cliques. The latter problem is just the Graph Isomorphism problem considered in [BCZ13a, Theorem 13], whose proof, with the help of the Transfer Theorem, becomes very simple.

With constraint and relation index revealing oracles, we show that if the arity and the alphabet size are constant, any CSP satisfying certain modest requirements becomes NP-hard.

Finally, we investigate hidden CSPs with promises on the instances. We first consider the repetition freeness promise, as exhibited by the 1SAT example as above. Though the hidden repetition free 1SAT problem becomes solvable in polynomial time, 2SAT is still NP-hard. The group isomorphism problem can also be cast in this framework, and we give a simplified proof of [BCZ13a, Theorem 11]: to compute an explicit isomorphism of the hidden group with $\mathbb{Z}_p$ is NP-hard.

Organization. In Section 2 we formally describe the model of CSPs, and hidden CSPs. In Section 3, the transfer theorems are stated and proved. Section 4, 5, and 6 contain the applications of the main theorems in the case of $\varnothing$-revealing, $\{\mathcal{V}\}$-revealing and $\{\mathcal{R}\}$-revealing oracles respectively. Finally in Section 7 we present the results for hidden promise CSPs.

2 Preliminaries

The model of constraint satisfaction problems. For a positive integer k, let $[k]$ denote the set $\{1, \ldots, k\}$. (Recall that $[\![k]\!] = \{0, 1, \ldots, k-1\}$.) A *constraint satisfaction problem*, (CSP) S, is specified by its set of parameters and its type, both defined for every positive integer n.

The *parameters* are the alphabet size $w(n)$, the assignment length $\ell(n)$, the set of (admissible) assignments $W(n) \subseteq [\![w(n)]\!]^{\ell(n)}$, the arity $q(n)$, and the number of relations $s(n)$. We suppose that $W(n)$ is symmetric, that is for every permutation $\pi \in S_{\ell(n)}$, if $a_1 \ldots a_{\ell(n)} \in W(n)$ then $a_{\pi(1)} \ldots a_{\pi(\ell(n))} \in W(n)$. To simplify notations, we often omit n from the parameters, and just write w, ℓ, W, q and s.

We denote by W_q the projection of W to q coordinates, i.e. $W_q = \{u \in [\![w]\!]^q : uv \in W \text{ for some } v \in [\![w]\!]^{\ell-q}\}$. A q-ary *relation* is $R \subseteq W_q$. For b in W_q, if $b \in R$, we sometimes write $R(b) = \mathrm{T}$, and similarly for $b \notin R$ we write $R(b) = \mathrm{F}$. The *type* of S is a set of q-ary relations $\mathcal{R}_n = \{R_1, \ldots, R_s\}$, where $R_k \subseteq W_q$, for every $k \in [s]$. As for the parameters, we usually just write $\mathcal{R}$.

We set $[\ell]^{(q)} = \{(j_1, \ldots, j_q) \in [\ell]^q : |\{j_1, \ldots, j_q\}| = q\}$, that is $[\ell]^{(q)}$ denotes the set of distinct q-tuples from $[\ell]$. An *instance* of S is given by a set of m (m may depend on n) constraints $\mathcal{C} = \{C_1, \ldots, C_m\}$ over a set $\mathcal{V} = \{x_1, \ldots, x_\ell\}$ of variables, where a *constraint* is $R_k(x_{j_1}, \ldots, x_{j_q})$ for some $k \in [s]$ and $(j_1, \ldots, j_q) \in [\ell]^{(q)}$. We say that an assignment $a \in W$ satisfies $C_j = R_k(x_{j_1}, \ldots, x_{j_q})$ if $R_k(a_{j_1}, \ldots, a_{j_q}) = \mathrm{T}$. An assignment *satisfies* $\mathcal{C}$ if it satisfies all its constraints. The *size* of an instance is $n + m(\log s + q \log \ell) + \ell \log w$ which includes the length of the description of $\mathcal{C}$ and the length of the assignments. In all our applications the instance size will be polynomial in n. A *solution* of $\mathcal{C}$ is a satisfying assignment if there exists any, and NO otherwise.

We further introduce the following notations. For a relation R let $\mathrm{comp}(R)$ be the time complexity of deciding the membership of a tuple in R, and for a set of relations $\mathcal{R}$ let $\mathrm{comp}(\mathcal{R})$ be $\max_{R \in \mathcal{R}} \mathrm{comp}(R)$. We denote by $\dim(\mathcal{R})$ the *dimension* of $\mathcal{R}$ which is defined as the length of the longest chain of relations (for inclusion) in $\mathcal{R}$.

We also introduce two new operations which create richer sets of relations from a relation set. For a given CSP S, these richer sets of relations derived from the type of S, will be the types of harder CSPs which turn out to be equivalent to various hidden variants of S. The first operation is standard. We denote by $\bigcup \mathcal{R}$ the closure of $\mathcal{R}$ by the union operation, that is $\bigcup \mathcal{R} = \{\bigcup_{R \in \mathcal{R}'} R : \mathcal{R}' \subseteq \mathcal{R}\}$. We define the (*closure by*) *union* of S as the constraint satisfaction problem $\bigcup$S whose parameters are the same as those of S, and whose type is $\bigcup \mathcal{R}$. We remark that $\dim(\bigcup \mathcal{R}) \leq |\mathcal{R}|$.

For a relation $R \in \mathcal{R}$ and for $(j_1, \ldots, j_q) \in [\ell]^{(q)}$ we define the ℓ-ary relation $R^{(j_1,\ldots,j_q)} = \{a \in W : (a_{j_1}, \ldots, a_{j_q}) \in R\}$, and X–$\mathcal{R}$ $= \{R^{(j_1,\ldots,j_q)} : R \in \mathcal{R} \text{ and } (j_1, \ldots, j_q) \in [\ell]^{(q)}\}$. The set X–$\mathcal{R}$ contains the natural extension of relations in $\mathcal{R}$ from arbitrary coordinates. If we want to consider unions of the same relation from arbitrary coordinates, then for $I \subseteq [\ell]^{(q)}$, we set $R^I = \bigcup_{(j_1,\ldots,j_q) \in I} R^{(j_1,\ldots,j_q)}$, and define the *arity extension* of $\mathcal{R}$, as E–$\mathcal{R}$ $= \bigcup_{R \in \mathcal{R}} \{R^I : I \subseteq [\ell]^{(q)}\}$. Observe that E–$\mathcal{R} \subseteq \bigcup$ X–$\mathcal{R} = \bigcup$ E–$\mathcal{R}$. The *arity extension* of S is the constraint satisfaction problem E–S whose parameters are the same as those of S except for the arity which becomes ℓ. The type of E–S is E–$\mathcal{R}$. The problem X–S is defined similarly, but with type X–$\mathcal{R}$.

Hidden CSP **in the trial and error model.** Suppose that we want to solve a CSP problem S whose parameters and type are known to us, but for the instance $\mathcal{C}$, we are explicitly given only n and the number of constraints m. The instance is otherwise specified by a *revealing* oracle V for $\mathcal{C}$ which can be used by an algorithm to receive information about the constraints in $\mathcal{C}$. The algorithm can propose $a \in W$ to the oracle which is conceived as its guess for a satisfying assignment. If a indeed satisfies $\mathcal{C}$ then V answers YES. Otherwise there exists some violated constraint $C_j = R_k(x_{j_1}, \ldots, x_{j_q})$, and the oracle has to reveal some information about that. We will require that the oracle always reveals j, the index of the constraint C_j in $\mathcal{C}$, but in addition, it can also make further disclosures. These can be k, the index of the relation R_k in $\mathcal{R}$; $(j_1, \ldots, j_q)$, the q-tuple of indices of the ordered variables $x_{j_1}, \ldots, x_{j_q}$ in $\mathcal{V}$; or both of these. To characterize the choices for the additional information, for any subset $\mathcal{U} \subseteq \{\mathcal{R}, \mathcal{V}\}$, we require that a $\mathcal{U}$-*revealing* oracle $\mathsf{V}_\mathcal{U}$ give out the information corresponding to $\{\mathcal{C}\} \bigcup \mathcal{U} \subseteq \{\mathcal{C}, \mathcal{R}, \mathcal{V}\}$. Thus for example a $\varnothing$-revealing oracle $\mathsf{V}_\varnothing$ reveals the index j of some violated constraint but nothing else, whereas a $\mathcal{V}$-revealing oracle $\mathsf{V}_{\{\mathcal{V}\}}$ also reveals the indices $(j_1, \ldots, j_q)$ of the variables of the relation in the clause C_j, but not the name of the relation.

Analogously, for every CSP S, and for every $\mathcal{U} \subseteq \{\mathcal{R}, \mathcal{V}\}$, we define the *hidden constraint satisfaction problem* (H–CSP) with *$\mathcal{U}$-revealing oracle* H–$\mathrm{S}_{\mathcal{U}}$ whose parameters and type are those of S, but whose instances are specified by a $\mathcal{U}$-revealing oracle. An algorithm *solves* the problem H–$\mathrm{S}_{\mathcal{U}}$ if for all n, m, for every instance $\mathcal{C}$ for S, specified by any $\mathcal{U}$-revealing oracle for $\mathcal{C}$, it outputs a satisfying assignment if there exists any, and NO otherwise. The complexity of an algorithm for H–$\mathrm{S}_{\mathcal{U}}$ is the number of steps in the worst case over all inputs and all $\mathcal{U}$-revealing oracles, where a query to the oracle is counted as one step.

3 Transfer Theorems for Hidden CSPs

In this section we precisely state our transfer theorems between H–CSPs and CSPs with extended types.

Theorem 3.1. *(a) If* $\bigcup \mathrm{S}$ *is solvable in time* T *then* H–$\mathrm{S}_{\{\mathcal{V}\}}$ *is solvable in time* $O((T+s\times\mathrm{comp}(\mathcal{R}))\times m \times \min\{\dim(\bigcup\mathcal{R}), |W_q|\})$.
(b) If H–$\mathrm{S}_{\{\mathcal{V}\}}$ *is solvable in time* T *then* $\bigcup \mathrm{S}$ *is solvable in time* $O(T \times m \times \mathrm{comp}(\bigcup\mathcal{R}))$.

Proof. We first prove (a). Let $\mathcal{A}$ be an algorithm which solves $\bigcup \mathrm{S}$ in time T. We define an algorithm $\mathcal{B}$ for H–$\mathrm{S}_{\{\mathcal{V}\}}$. The algorithm will repeatedly call $\mathcal{A}$, until it finds a satisfying assignment or reaches the conclusion NO. The instance $\mathcal{C}^t = \{C_1^t, \ldots, C_m^t\}$ of the tth call is defined as $C_j^t = \bigcup_{R\in\mathcal{R}: R\cap A_j^t = \varnothing} R(x_{j_1^t}, \ldots, x_{j_q^t})$ where $A_j^t \subseteq W_q$ and $(j_1^t, \ldots, j_q^t) \in [\ell]^{(q)}$, for $j \in [m]$, are determined successively by $\mathcal{B}$. Initially $A_j^1 = \varnothing$ and $(j_1^1, \ldots, j_q^1)$ is arbitrary. If the output of $\mathcal{A}$ for $\mathcal{C}^t$ is NO then $\mathcal{B}$ outputs NO. If the output of $\mathcal{A}$ for $\mathcal{C}^t$ is $a \in W$ then $\mathcal{B}$ submits a to the $\{\mathcal{V}\}$-revealing oracle V. If V answers YES then $\mathcal{B}$ outputs a. If the oracle does not find a satisfying, and reveals j and $(j_1, \ldots, j_q)$ about the violated constraint, then $\mathcal{B}$ does not change A_i^t and $(i_1^1, \ldots, i_q^1)$ for $i \neq j$, but sets $A_j^{t+1} = A_j^t \bigcup \{(a_{j_1}, \ldots, a_{j_q})\}$, and $(j_1^{t+1}, \ldots, j_q^{t+1}) = (j_1, \ldots, j_q)$. Observe that the q-tuple for the jth constraint is changed at most once, the first time when the revealing oracle gives the index of the jth constraint.

To prove that the algorithm correctly solves H–$\mathrm{S}_{\{\mathcal{V}\}}$, let $\mathcal{C} = \{C_1, \ldots, C_m\}$ be an instance of S and let V be any $\{\mathcal{V}\}$-revealing oracle for $\mathcal{C}$. We have to show that if $\mathcal{B}$ answers NO then $\mathcal{C}$ is unsatisfiable. If $\mathcal{B}$ answers NO, then for some t, the tth call of $\mathcal{A}$ resulted in output NO. By construction A_j^t and $(j_1^t, \ldots, j_q^t)$, for every $j \in [m]$, are such that if $R \cap A_j^t \neq \varnothing$ then C_j can't be $R(x_{j_1}, \ldots, x_{j_q})$. Indeed, if $C_j = R(x_{j_1}, \ldots, x_{j_q})$ and $b \in R \cap A_j^t$ then at the call when b was added to A_j^t the oracle's answer is incorrect. Therefore all possible remaining R_js are included in C_j^t, and since $\mathcal{C}^t$ is unsatisfiable, so is $\mathcal{C}$.

For the complexity of the algorithm let us remark that if for some j and t, the constraint C_j^t is the empty relation then $\mathcal{B}$ stops since $\mathcal{C}^t$ becomes unsatisfiable. This happens in particular if $A_j^t = W_q$. Since for every call to $\mathcal{A}$ one new element is added to one of the A_j^t and at least one new relation in $\mathcal{R}$ is excluded from C_j^t, the number of calls is upper bounded by $m \times \min\{\dim(\mathcal{R}), |W_q|\}$. To compute a new constraint, some number of relations in $\mathcal{R}$ have to be computed on a new argument, which can be done in time $s \times \mathrm{comp}(\mathcal{R})$.

We now prove (b). Let $\mathcal{A}$ be an algorithm which solves H–$\mathrm{S}_{\{\mathcal{V}\}}$ in time T. Without loss of generality we suppose that $\mathcal{A}$ only outputs a satisfying assignment a after submitting it to the verifying oracle. We define an algorithm $\mathcal{B}$ for $\bigcup \mathrm{S}$. Let $\mathcal{C} = \{C_1, \ldots, C_m\}$ be an instance of $\bigcup \mathrm{S}$ where for $j \in [m]$, $C_j = \bigcup_{R\in\mathcal{R}_j} R(x_{j_1}, \ldots, x_{j_q})$, for some $\mathcal{R}_j \subseteq \mathcal{R}$ and $(j_1, \ldots, j_q) \in [\ell]^{(q)}$. The

algorithm $\mathcal{B}$ runs $\mathcal{A}$, and outputs NO whenever $\mathcal{A}$ outputs NO. During $\mathcal{A}$'s run $\mathcal{B}$ simulates a $\{\mathcal{V}\}$-revealing oracle V for $\mathcal{A}$ which we describe now. Simultaneously with V's description we also specify instances $\mathcal{C}^t = \{C_1^t, \ldots, C_m^t\}$ of $\bigcup$S which will be used in the proof of correctness of the algorithm. For $j \in [m]$, the constraints of $\mathcal{C}^t$ are defined as $C_j^t = \bigcup_{R \in \mathcal{R}: R \cap A_j^t = \varnothing} R(x_{j_1^t}, \ldots, x_{j_q^t})$, where the sets $A_j^t \subseteq W_q$ are determined by the result of the tth call to the oracle. Initially $A_j^0 = \varnothing$. For every request $a \in W$, the algorithm $\mathcal{B}$ checks if a satisfies $\mathcal{C}$. If it is the case then V returns a and $\mathcal{B}$ outputs a. Otherwise there exists $j \in [m]$ such that a violates C_j, and the answer of the oracle is j and $(j_1, \ldots, j_q)$ (where j can be chosen arbitrarily among the violated constraints, if there are several). Observe that this is a legitimate oracle for any instance of H–S$_{\{\mathcal{V}\}}$ whose jth constraint is arbitrarily chosen from $\mathcal{R}_j$. We define $A_j^{t+1} = A_j^t \bigcup \{(a_{j_1}, \ldots, a_{j_q})\}$, and for $i \neq j$ we set $A_i^{t+1} = A_i^t$.

To show the correctness of $\mathcal{B}$, we prove that whenever $\mathcal{A}$ outputs NO, the instance $\mathcal{C}$ is unsatisfiable. Let us suppose that $\mathcal{A}$ made t queries before outputting NO. An algorithm for H–S$_{\{\mathcal{V}\}}$ can output NO only if all possible instances of S which are compatible with the answers received from the oracle are unsatisfiable. In such an instance the jth constraint has necessarily empty intersection with A_j^t, therefore we can deduce that the $\bigcup$S instance $\mathcal{C}^t$ is unsatisfiable. It also holds that $A_j^t \bigcap C_j = \varnothing$ for every $j \in [m]$, since if $b \in A_j^t \bigcap C_j$ then the request to the oracle because of which b was added to A_j^t wouldn't violate the jth constraint. Thus $C_j \subseteq C_j^t$, and $\mathcal{C}$ is unsatisfiable.

For the complexity analysis we observe that during the algorithm, for every query to the oracle and for every constraint, one relation in $\bigcup\mathcal{R}$ is evaluated. □

Theorem 3.2. *(a) If* E–S *is solvable in time T then* H–S$_{\{\mathcal{R}\}}$ *is solvable in time* $O((T + |[\ell]^{(q)}| \times \mathrm{comp}(\mathcal{R})) \times m \times |[\ell]^{(q)}|)$.
(b) If H–S$_{\{\mathcal{R}\}}$ *is solvable in time T then* E–S *is solvable in time* $O(T \times m \times \mathrm{comp}(\text{E–}\mathcal{R}))$.

Proof. The proof is very similar to the proof of Theorem 3.1. We first prove (a). Let $\mathcal{A}$ be an algorithm which solves E–S in time T. We define an algorithm $\mathcal{B}$ for H–S$_{\{\mathcal{R}\}}$. The algorithm will repeatedly call $\mathcal{A}$, until it finds a satisfying assignment or reaches the conclusion NO. Initially, in the first call there is no constraint, which we formally describe by $\mathcal{C}^1 = \{C_1^1, \ldots, C_m^1\}$ where $C_j^1 = W$. For $t > 1$, the instance of the tth call will be described via $A_j^t \subseteq W$ and $I_j^t \subseteq [\ell]^{(q)}$, for $j \in [m]$, where initially $A_1^1 = \ldots = A_m^1 = \varnothing$ and $I_1^1 = \ldots = I_m^1 = [\ell]^{(q)}$. If the output of $\mathcal{A}$ for $\mathcal{C}^{t-1}$ is NO then $\mathcal{B}$ outputs NO. If the output of $\mathcal{A}$ for $\mathcal{C}^{t-1}$ is $a \in W$ then $\mathcal{B}$ submits a to the $\{\mathcal{R}\}$-revealing oracle V. If V answers YES then $\mathcal{B}$ outputs a. If the oracle does not find a satisfying, and reveals j and $R \in \mathcal{R}$ about the violated constraint, then $\mathcal{B}$ does not change A_i^{t-1} and I_i^{t-1} for $i \neq j$, but sets $A_j^t = A_j \bigcup \{(a_{j_1}, \ldots, a_{j_q})\}$, and $I_j^t = \{(j_1, \ldots, j_q) \ : \ A_j^t \bigcap R^{(j_1, \ldots, j_q)} = \varnothing\}$. Then the instance $\mathcal{C}^t = \{C_1^t, \ldots, C_m^t\}$ of the tth call is defined by $C_j^t = R^{I_j^t}$.

To prove that the algorithm correctly solves H–S$_{\{\mathcal{R}\}}$, let $\mathcal{C} = \{C_1, \ldots, C_m\}$ be an instance of S and let V be any $\{\mathcal{R}\}$-revealing oracle for $\mathcal{C}$. We have to show that if $\mathcal{B}$ answers NO then $\mathcal{C}$ is unsatisfiable. If $\mathcal{B}$ answers NO then for some t, the tth call of $\mathcal{A}$ resulted in output NO. By construction A_j^t and I_j^t are such that for every constraint C_j whose relation R has been already revealed, if $R^{(j_1, \ldots, j_q)} \cap A_j^t \neq \varnothing$ then C_j can not be $R(x_{j_1}, \ldots, x_{j_q})$. Indeed, if $C_j = R(x_{j_1}, \ldots, x_{j_q})$ and $a \in R \cap A_j^t$ then at the call when a was added to A_j^t the oracle answer is incorrect. Therefore C_j^t is the union of all possible remaining $R^{(j_1, \ldots, j_q)}$, and since $\mathcal{C}^t$ is unsatisfiable, so is $\mathcal{C}$.

For the complexity of the algorithm let us remark that if for some j and t, the constraint C_j^t is the empty relation then $\mathcal{B}$ stops since $\mathcal{C}^t$ becomes unsatisfiable. This happens in particular if $I_j^t = \varnothing$.

Since for every call to $\mathcal{A}$, for some j the size of I_j^t decreases by at most one, the total number of calls is upper bounded by $m \times |[\ell]^{(q)}|$.To compute a new constraints, at most $|[\ell]^{(q)}|$ relations from $\mathcal{R}$ evaluated in a new argument. Therefore the overall complexity is as claimed

We now prove (b). Let $\mathcal{A}$ be an algorithm which solves H–S$_{\{\mathcal{R}\}}$ in time T. Without loss of generality we suppose that $\mathcal{A}$ only outputs a satisfying assignment a after submitting it to the verifying oracle. We define an algorithm $\mathcal{B}$ for E–S. Let $\mathcal{C} = \{C_1, \ldots, C_m\}$ be an instance of E–S where for $j \in [m]$, we have $C_j = R^{I_j}$ for some $R \in \mathcal{R}$ and $I_j \subseteq [\ell]^{(q)}$. The algorithm $\mathcal{B}$ runs $\mathcal{A}$, and outputs NO whenever $\mathcal{A}$ outputs NO. During $\mathcal{A}$'s run $\mathcal{B}$ simulates an $\{\mathcal{R}\}$-revealing oracle V for $\mathcal{A}$ which we describe now. Simultaneously with V's description we also specify instances $\mathcal{C}^t = \{C_1^t, \ldots, C_m^t\}$ of E–S which will be used in the proof of correctness of the algorithm. Initially $C_j^0 = W$ for every $j \in [m]$. For $t \geq 1$, the constraints of $\mathcal{C}^t$ are defined as $C_j^t = R^{I_j^t}$, where the sets $I_j^t \subseteq [\ell]^{(q)}$ are $I_j^t = \{(j_1, \ldots, j_q) \ : \ A_j^t \bigcap R^{(j_1,\ldots,j_q)} = \varnothing\}$, and the sets $A_j^t \subseteq W_q$ are determined by the result of the tth call to the oracle. Initially $A_j^0 = \varnothing$. For every request $a \in W$, the algorithm $\mathcal{B}$ checks if a satisfies $\mathcal{C}$. If it is the case then ((V returns a and)) $\mathcal{B}$ outputs a. Otherwise there exist $j \in [m]$ such that a violates C_j, and the answer of the oracle is j and $R \in \mathcal{R}$. Observe that this is a legitimate oracle for any instance of H–S$_{\{\mathcal{R}\}}$ whose jth constraint is arbitrarily chosen from $\{R^{(j_1,\ldots,j_q)} \ : \ (j_1, \ldots, j_q) \in I_j\}$. We define $A_j^{t+1} = A_j \bigcup \{(a_{j_1}, \ldots, a_{j_q})\}$, and for $i \neq j$ we set $A_i^{t+1} = A_i^t$.

To show the correctness of $\mathcal{B}$, we prove that whenever $\mathcal{A}$ outputs NO, the instance $\mathcal{C}$ is unsatisfiable. Let us suppose that $\mathcal{A}$ made t queries before outputting NO. An algorithm for H–S$_{\{\mathcal{R}\}}$ can output NO only of all possible instances of S which are compatible with the answers received from the oracle are unsatisfiable. In such an instance the jth constraint has necessarily empty intersection with A_j^t, therefore we can deduce that the E–S instance $\mathcal{C}^t$ is unsatisfiable. It also holds that $A_j^t \bigcap C_j = \varnothing$ for every $j \in [m]$, since if $a \in A_j^t \bigcap C_j$ then the request to the oracle because of which a was added to A_j^t wouldn't violate the jth constraint. Thus $C_j \subseteq C_j^t$, and $\mathcal{C}$ is unsatisfiable.

For the complexity analysis we just have to observe that during the algorithm, for every query to the oracle and for every constraint, one relation in E–$\mathcal{R}$ is evaluated. □

Theorem 3.3. *(a) If* $\bigcup$E–S *is solvable in time* T *then* H–S$_{\varnothing}$ *is solvable in time* $O((T + s \times |[\ell]^{(q)}| \times \mathrm{comp}(\mathcal{R})) \times m \times \dim(\bigcup \text{E–}\mathcal{R}))$.
(b) If H–S$_{\varnothing}$ *is solvable in time* T *then* $\bigcup$E–S *is solvable in time* $O(T \times m \times \mathrm{comp}(\bigcup \text{E–}\mathcal{R}))$.

Theorem 3.4. *(a) If* $\bigcup$X–S *is solvable in time* T *then* H–S$_{\varnothing}$ *is solvable in time* $O((T + s \times \frac{\ell!}{(\ell-q)!} \times \mathrm{comp}(\mathcal{R})) \times m \times \dim(\text{X–}\mathcal{R}))$.
(b) If H–S$_{\varnothing}$ *is solvable in time* T *then* $\bigcup$X–S *is solvable in time* $O(T \times m \times \mathrm{comp}(\bigcup \text{X–}\mathcal{R}))$.

Proof. Apply Theorem 3.1 to X–S and observe that H–X–S$_{\{\mathcal{V}\}}$ and H–S$_{\varnothing}$ are essentially the same in the sense that an algorithm solving one of the problems also solves the other one. Indeed, the variable index disclosure of the $\{\mathcal{V}\}$-revealing oracle is pointless since the relations in X–S involve all variables. Moreover, the map sending a constraint $R(x_{j_1}, \ldots x_{j_q})$ of S to the constraint $R^{(j_1,\ldots,j_q)}(x_1, \ldots x_\ell)$ of X–S is a bijection which preserves satisfying assignments. □

Corollary 3.5. *Let* $\mathrm{comp}(\mathcal{R})$ *be polynomial. Then the complexities of the following problems are polynomial time equivalent: (a)* H–S$_{\{\mathcal{V}\}}$ *and* $\bigcup$S *if the number of relations* s *is constant, (b)* H–S$_{\{\mathcal{R}\}}$ *and* E–S *if the arity* q *is constant, (c)* H–S$_{\varnothing}$ *and* $\bigcup$E–S *if both* s *and* q *are constant.*

The polynomial time equivalence of Theorems 3.1, 3.2, 3.4 and Corollary 3.5 remain true when the algorithms have access to the same computational oracle. Therefore, we get generic easiness results for H–CSPs under an NP oracle.

4 Constraint-index Revealing Oracle

In this section, we present some applications of our transfer theorems in the context of the constraint-index revealing oracle. Here we propose a framework for monotone graph properties to present our examples. Recall that a *monotone graph property* of an n-vertex graphs is a monotone Boolean function $\mathcal{P}$ on $\binom{n}{2}$ variables. The the CSP $\mathrm{S}_{\mathcal{P}}$ associated with $\mathcal{P}$ has parameters $w = 2$, $q = 1$, $\ell = \binom{n}{2}$, $W_{\mathcal{P}} = \{A \mid A \text{ is a graph with minimal number of edges satisfying } \mathcal{P}\}$, and $\mathcal{R} = \{\mathsf{Neg}\}$, where Neg is the negation function. The goal is to decide, given a graph $G = (V, E)$, whether there exists an $A \in W_{\mathcal{P}}$ such that $A \subseteq G$. The corresponding constraints are $e \notin A$ for every $e \notin E$. We have $\mathrm{X}\text{–}\mathcal{R} = \{\mathsf{Neg}^e \mid e \in \binom{n}{2}\}$, where $\mathsf{Neg}^e(\alpha_1, \ldots, \alpha_{\binom{n}{2}}) = \neg\alpha_e$. Thus, the $\bigcup\mathrm{X}\text{–}\mathrm{S}_{\mathcal{P}}$ problem becomes the following: given a graph $G = (V, E)$, and $E_1, \ldots, E_m \subseteq \binom{[n]}{2}$, does there exist an $A \in W_{\mathcal{P}}$ such that $A \subseteq E$ and A excludes at least one edge from each E_i? This framework naturally extends to directed graphs. Also monotone decreasing properties can be treated by replacing Neg with Id.

From Theorem 3.3, the complexity of $\mathrm{H}\text{–}\mathrm{S}_{\mathcal{P}}$ can be analyzed by considering the complexity of $\bigcup\mathrm{X}\text{–}\mathrm{S}_{\mathcal{P}}$. We do this for the following graph properties:
1. Spanning Tree (ST): the property of being connected
2. Directed Spanning Tree (DST): the property of containing a directed spanning tree rooted at vertex (say) 1 such that all the edges of the spanning tree are directed towards the root.
3. Undirected Cycle Cover (UCC): the property of containing an undirected cycle cover (union of vertex disjoint cycles such that every vertex belongs to some cycle)
4. Directed Cycle Cover (DCC): the property of containing a directed cycle cover (union of vertex disjoint directed cycles such that every vertex belongs to some cycle)
5. Bipartite Perfect Matching (BPM): the property of having a perfect matching in a bipartite graph
6. Directed Path (DPATH): the property of containing a directed path between two specified vertices s and t.
7. Undirected Path (UPATH): the property of containing an undirected path between two specified vertices s and t.

Theorem 4.1. *The following problems are* NP*-hard: (1)* $\mathrm{H}\text{–}\mathsf{ST}_{\varnothing}$*, (2)* $\mathrm{H}\text{–}\mathsf{DST}_{\varnothing}$*, (3)* $\mathrm{H}\text{–}\mathsf{UCC}_{\varnothing}$*, (4)* $\mathrm{H}\text{–}\mathsf{DCC}_{\varnothing}$*, (5)* $\mathrm{H}\text{–}\mathsf{BPM}_{\varnothing}$*, (6)* $\mathrm{H}\text{–}\mathsf{DPATH}_{\varnothing}$*, (7)* $\mathrm{H}\text{–}\mathsf{UPATH}_{\varnothing}$*.*

Proof. We show that the following problems in the hidden model with the constraint index revealing oracle are NP-hard.

(1) Spanning Tree ($\mathrm{H}\text{–}\mathsf{ST}_{\varnothing}$)

Here $\mathcal{P}$ is *connectedness* and $\mathcal{A}$ is the set of *Spanning Trees* on n-vertices.

Proof. Given $G = (V, E)$, for every vertex $v \in V$, we consider $\binom{n-1}{3}$ subsets $E_{vijk} : 1 \le i < j < k \le n$, where $E_{vijk} := \{\{v, i\}, \{v, j\}, \{v, k\}\}$. With this choice of E_{vijk}s the $\bigcup\mathrm{X}\text{–}\mathsf{ST}$ problem becomes: does there exist a spanning tree in G which avoids at least one edge from each

E_{vijk}. This is exactly the HAM−PATH problem in G, i.e., does G contain a Hamiltonian path. Hence from Theorem 3.3(b), the former is NP-hard. □

(2) Directed Spanning Tree (H–$\mathsf{DST}_\varnothing$)

Here $\mathcal{P}$ is the property that the graph contains a directed spanning tree rooted at vertex (say) 1 such that all the edges of the spanning tree are directed towards the root. Then $\mathcal{A}$ is the set of directed spanning trees rooted at vertex 1.

Proof. Let $G = (V, E)$, be a directed planar graph such that the indegree and the outdegree for every vertex is at most 2. The DHAM−PATH problem in such a G, i.e, does G contain a directed Hamiltonian path ending at node 1, is NP-hard [GJ79]. Our goal is to reduce the DHAM−PATH problem in G to the H–$\mathsf{DST}_\varnothing$ problem in G.

For every vertex $v \in V$, let $E_v := \{(i, v) \mid (i, v) \in E\}$, where $|E_v| \leq 2$ by our choice of G.

With this choice of E_vs the $\bigcup$X–DST problem becomes the DHAM−PATH problem in G. Hence from Theorem 3.3(b), the former is NP-hard. □

(3) Undirected Cycle Cover (H–$\mathsf{UCC}_\varnothing$)

Here $\mathcal{P}$ is the property of containing an undirected cycle cover (union of vertex disjoint cycles such that every vertex belongs to some cycle) and $\mathcal{A}$ is the set of *undirected cycle covers* on n-vertices.

Proof. From Hell et al. [HKKK88] we know that the problem of deciding whether a graph has a UCC that does not use the cycles of length (say) 5 is NP-hard. This problem can be expressed as $\bigcup$X–UCC by choosing the subsets $E_C := \{e \mid e \in C\}$ for every length 5 cycle C in G. □

(4) Directed Cycle Cover (H–$\mathsf{DCC}_\varnothing$)

Here $\mathcal{P}$ is the property of containing a directed cycle cover (union of vertex disjoint directed cycles such that every vertex belongs to some cycle) and $\mathcal{A}$ is the set of *directed cycle covers* on n-vertices.

Proof. From [GJ79] we know that the problem of deciding whether a graph has a DCC that does not use cycles of length 1 and 2 is NP-hard. This problem can be expressed as $\bigcup$X–DCC by choosing the subsets $E_C := \{e \mid e \in C\}$ for every length 1 and length 2 cycle C in G. □

(5) Bipartite Perfect Matching (H–$\mathsf{BPM}_\varnothing$)

Here $\mathcal{P}$ is the property of containing a perfect matching in a bipartite graph and $\mathcal{A}$ is the set of *perfect matchings* in a complete bipartite graph with n-vertices on each side.

Proof. There is a one to one correspondence between perfect matchings in a bipartite graph $G = (A \cup B, E)$ with n vertices on each side and the directed cycle covers in graph $G' = (V', E')$ on n vertices. Every edge $(i, j) \in E'$ corresponds to an undirected edge $\{i_A, j_B\} \in E$. With this correspondence H–$\mathsf{BPM}_\varnothing$ in G becomes H–$\mathsf{DCC}_\varnothing$ in G'. Thus from Theorem 4.1(4) the former becomes NP-hard. □

(6) Directed Path (H–$\mathsf{DPATH}_{\varnothing}$)

Here $\mathcal{P}$ is the property of containing a directed path between two specified vertices s and t. $\mathcal{A}$ is the set of *directed paths* from s to t.

Proof. It is known that given a layout of a graph on a plane possibly containing crossings, the problem of deciding whether there is a crossing-free path from s to t is NP-hard [KLN91]. One can express this condition by picking E_is to be the set of pairs of edges that cross. □

(7) Undirected Path (H–$\mathsf{UPATH}_{\varnothing}$)

Here $\mathcal{P}$ is the property of containing a undirected path between two specified vertices s and t. $\mathcal{A}$ is the set of *undirected paths* from s to t.

Proof. Apply the same proof method as the one used for the H–$\mathsf{DPATH}_{\varnothing}$ problem on an undirected graph. □

This completes the proof for Theorem 4.1 □

5 Constraint-index and Variable-index Revealing Oracle

In this section, we present some applications of our transfer theorem when the index of the constraint and the indices of the variables participating in that constraint are revealed. We consider following CSPs:

1. Deltas on Triplets (Δ): $w = 2$, $q = 3$, and $\mathcal{R} = \{R_{abc} : \{0,1\}^3 \to \{\mathrm{T},\mathrm{F}\} \mid a,b,c \in \{0,1\}\}$, where $R_{abc}(x,y,z) := (x = a) \wedge (y = b) \wedge (z = c)$.

2. Subgroup Non-cover ($\mathsf{SUBGRP-NC}[S]$): Given a group G of size n and a class, S, of subgroups of G, i.e. $S \subseteq \{H \mid H \leq G\}$, the subgroup non-cover problem asks if $\exists\, g \in G$ such that $g \notin \bigcup_{H \in S'} H$ where $S' \subseteq S$. Formally, $\ell = 1$, $w = n$, $q = 1$, $W = G$ and $\mathcal{R}_S = \{R_H \mid H \in S\}$ where $R_H(a)$ evaluates to T if and only if $a \notin H$. A special case is Hyperplane Non-Cover ($\mathsf{HYP-NC}$): $G = Z_p^N$ and $S = \{\text{all hyperplanes in } Z_p^N\}$.

3. Arbitrary sets of binary relations on Boolean alphabet, in particular, the 2-SAT Problem ($\mathsf{2SAT}$): $w = 2$, $q = 2$, and $\mathcal{R} = \{R_T, R_F, R_a, R_b, R_{\neg a}, R_{\neg b}, R_{a\vee b}, R_{a\vee\neg b}, R_{\neg a\vee b}, R_{\neg a\vee\neg b}\}$, where for $(\alpha,\beta) \in \{\mathrm{T},\mathrm{F}\}^q$, $R_T(\alpha,\beta) := \mathrm{T}, R_F(\alpha,\beta) := \mathrm{F}, R_a(\alpha,\beta) := \alpha, R_b(\alpha,\beta) := \beta, R_{\neg a}(\alpha,\beta) := \neg\alpha$, $R_{\neg b}(\alpha,\beta) := \neg\beta, R_{a\vee b}(\alpha,\beta) := \alpha \vee \beta, R_{a\vee\neg b}(\alpha,\beta) := \alpha \vee \neg\beta, R_{\neg a\vee b}(\alpha,\beta) := \neg\alpha \vee \beta, R_{\neg a\vee\neg b}(\alpha,\beta) := \neg\alpha \vee \beta$.

4. Exact-Unique Game Problem ($\mathsf{UG}[k]$): Given an undirected graph $G = (V,E)$ and given a permutation $\pi_e : [\![k]\!] \to [\![k]\!]$, for every edge $e \in E$, the goal is to decide if one can assign labels $\alpha_v \in [\![k]\!]$ for every vertex $v \in V$ such that for every edge $e = \{u,v\} \in E$ with $u < v$ we have $\pi_e(\alpha_u) = \alpha_v$. Formally: $w = k$, $q = 2$ and $\mathcal{R} = \{\pi : [\![k]\!] \to [\![k]\!] \mid \pi \text{ is a permutation}\}$.

5. k-Clique Isomorphism ($k\mathsf{CLQ}$–ISO): Given an undirected graph $G = (V,E)$, does there exist a permutation π on $[n]$ such that: (a) $\forall (i,j) \in E,\ R_{\leq k}(\pi(i),\pi(j))$; (b) $\forall (i,j) \notin E,\ \neg R_{\leq k}(\pi(i),\pi(j))$. Formally: $W =$ set of permutations on $[n]$, $w = n$, $q = 2$, and $\mathcal{R} = \{R_{\leq k}, \neg R_{\leq k}\}$, where $R_{\leq k}(\alpha,\beta) := \mathrm{T} \iff \alpha \leq k \,\&\, \beta \leq k$.

6. Polynomial time Solvable Graph Properties ($\mathcal{P}^{poly}$): The framework for graph properties is the same as defined in Section 4. Here we study them with $\{\mathcal{V}\}$-revealing oracle.

7. Equality to some member in a fixed class of graphs ($\mathsf{EQ}_{\mathcal{K}}$): For a fixed class $\mathcal{K}$ of graphs on n vertices, we denote by $\mathcal{P}_{\mathcal{K}} : \{0,1\}^{\binom{n}{2}} \to \{T, F\}$ the property of being equal to a graph from $\mathcal{K}$. Formally, $W = \mathcal{K}$, $w = 2$, $q = 1$, $\ell = \binom{n}{2}$, and $\mathcal{R} = \{\mathsf{Id}, \mathsf{Neg}\}$.

(a) Equality to k-Clique ($\mathsf{EQ}_{k\mathsf{CLQ}}$): Given a graph, decide if it is equal to a k-clique.

(b) Equality to Hamiltonian Cycle ($\mathsf{EQ}_{\mathsf{HAMC}}$): Decide if G is a cycle on all n vertices.

(c) Equality to Spanning Tree ($\mathsf{EQ}_{\mathsf{ST}}$): Given a graph, decide if it is a spanning tree.

Theorem 5.1. *The following problems are in polynomial time: (a)* H–$\mathsf{2SAT}_{\{\mathcal{V}\}}$*, (b)* H–$\mathsf{UG}[2]_{\{\mathcal{V}\}}$*, (c)* H–$\mathcal{P}^{poly}{}_{\{\mathcal{V}\}}$*, (d)* H–$\mathsf{EQ}_{\mathsf{ST}\{\mathcal{V}\}}$*.*

Theorem 5.2. *The following are* NP*-hard: (a)* H–$\Delta_{\{\mathcal{V}\}}$*, (b)* H–$\mathsf{HYP-NC}_{\{\mathcal{V}\}}$*, (c)* H–$\mathsf{UG}[k]_{\{\mathcal{V}\}}$ *for* $k \geq 3$*, (d)* H–$k\mathsf{CLQ}$–$\mathsf{ISO}_{\{\mathcal{V}\}}$*, (e)* H–$\mathsf{EQ}_{k\mathsf{CLQ}\{\mathcal{V}\}}$*, (f)* H–$\mathsf{EQ}_{\mathsf{HAMC}\{\mathcal{V}\}}$*.*

Proof of Theorem 5.1. We show that the following problems in the hidden model with the constraint and variable index revealing oracle are solvable in polynomial time.

(a) Arbitrary binary Boolean relations (H–$\mathsf{2SAT}_{\{\mathcal{V}\}}$)

Proof. We claim that $\bigcup\mathcal{R} = \mathcal{R}$. Hence, $\bigcup\mathsf{2SAT} = \mathsf{2SAT}$, which is in P.

Therefore, from Theorem 3.1(a), H–$\mathsf{2SAT}_{\{\mathcal{V}\}}$ is also in P. □

The above proof can be extended to an *arbitrary* set $\mathcal{R}'$ of binary relations as follows. Let $\mathcal{R}''$ stand for the set of *all* binary relations in Boolean variables. We trivially have $\bigcup\mathcal{R}' = \mathcal{R}''$, therefore an instance of H–$\mathsf{2SAT}_{\{\mathcal{V}\}}$ can actually be described by a conjunction of the form $\bigwedge_{k=1}^{m} R_k(x_{i_k}, x_{j_k})$ where R_k is a binary relation. Expressing each R_k by a Boolean formula in conjunctive normal form, we obtain an instance of $\mathsf{2SAT}$ consisting of $O(m)$ clauses, which can be solved in polynomial time.

(b) Unique Games (H–$\mathsf{UG}[2]_{\{\mathcal{V}\}}$)

Proof. Note that $\mathsf{UG}[2]$ is an instance of $\mathsf{2SAT}$, hence from Theorem 5.1(a), H–$\mathsf{UG}[2]_{\{\mathcal{V}\}}$ is in P. □

(c) Polynomial time solvable graph Properties (H–$\mathcal{P}^{poly}{}_{\{\mathcal{V}\}}$)

Proof sketch. Note that the variable indices being returned for a violated edge allow us to learn the exact endpoints of the edge. Since there are at most n^2 edges in a graph, we can learn which of those n^2 edges are absent in the graph using a polynomial number of queries. Hence, we can reconstruct the hidden graph and solve for the required graph property in polynomial time. □

(d) Equality/Isomorphism to a member in a fixed class of graphs

We define the H–$\mathsf{EQ}_{\mathcal{K}\{\mathcal{V}\}}$ problem in more detail. Let $\mathcal{K}$ be a class of graphs on n vertices. Similar to the definitions in Section 4, we define $\mathcal{P}_{\mathcal{K}} : \{0,1\}^{\binom{n}{2}} \to \{T, F\}$ as the graph property of being equal to a graph from $\mathcal{K}$. Correspondingly, $W = \mathcal{K}$.

Formally, for $\mathcal{P}_\mathcal{K}$ we consider the CSP $\mathsf{EQ}_\mathcal{K}$ with $w = 2$, $q = 1$, $W = \{\alpha \in \{0,1\}^{\binom{n}{2}} \mid \alpha \in \mathcal{K}\}$, $\ell = \binom{n}{2}$, and $\mathcal{R} = \{\mathsf{Id}, \neg\}$. Given a graph instance $G_1 = (V_1, E_1)$ in this model, the $\binom{n}{2}$ constraints for G_1 are such that $C_e = \mathsf{Id}(\alpha_e)$ for $e \in E$ and $C_e = \neg(\alpha_e)$ otherwise.

This implies that $\bigcup\mathcal{R} = \{\mathsf{Id}, \neg, \mathrm{T}\}$ and the $\bigcup\mathsf{EQ}_\mathcal{K}$ problem becomes: given sets E_1, E_2 such that $E_1 \subseteq E_2$, does there exist a graph $G \in \mathcal{K}$ such that $E_1 \subseteq G \subseteq E_2$?

From Theorem 3.1, the complexity of H–$\mathsf{EQ}_\mathcal{K}$, can be analyzed by considering the complexity of $\bigcup\mathsf{EQ}_\mathcal{K}$. Hence, we analyze the complexity of $\bigcup\mathsf{EQ}_\mathcal{K}$ for some examples below.

Remark 5.3. *For any $\mathcal{K}$, if we take $E_1 = \varnothing$, then solving $\mathsf{EQ}_\mathcal{K}$ becomes equivalent to finding out if there exists $G \in \mathcal{K}$ which is a subgraph of E_2.*

Remark 5.4. *Note that if we assume that $\mathcal{K}$ is the set of all graphs isomorphic to some G_0 and $E_1 = E_2$ as arbitrary graphs on n vertices, then solving $\mathsf{EQ}_\mathcal{K}$ becomes equivalent to finding out if E_2 is isomorphic to G_0.*

*Proof for Equality to a Spanning Tree (*H–$\mathsf{EQ}_{\mathsf{ST}\{\mathcal{V}\}}$*).* Here, $\mathcal{K}$ is the set of all possible Spanning Trees on n vertices and E_1 without loss of generality is a forest F. E_2 is any arbitrary graph on n vertices containing E_1. In this case, the $\bigcup\mathsf{EQ}_\mathcal{K}$ problem becomes equivalent to finding a spanning tree on E_2 which also contains the forest F. This problem is in P. □

This completes the proof for Theorem 5.1. □

Proof of Theorem 5.2. We show that the following problems in the hidden model with the constraint and variable index revealing oracle are NP-hard.

(a) Deltas on Triplets (H–$\Delta_{\{\mathcal{V}\}}$)

Proof. We claim that $\bigcup\mathcal{R}$ is the set of all Boolean predicates on 3 variables. Thus, $\mathsf{3SAT}$ can be expressed as the $\bigcup\Delta$ problem. Hence, from Theorem 3.1(b), H–$\Delta_{\{\mathcal{V}\}}$ is NP-hard. □

(b) Subgroup non-cover

Let $B = \{\bigcap_{H\in S'} H \mid S' \subseteq S\}$. Thus, $\bigcup\mathcal{R}_S = \{R'_D \mid D \in B\}$ where R'_D evaluates to T if $a \notin D$ and F otherwise. Then the $\bigcup\mathsf{SUBGRP-NC}[S]$ problem asks if $\exists\, g \in G$ such that $g \notin \bigcup_{D\in B} D$

Remark 5.5. *If S was just the set of all subgroups of G then any $B = \bigcap_{H\in H'} H$ where $H' \subseteq S$ is also a subgroup of G. Hence, the $\bigcup\mathsf{SUBGRP-NC}[S]$ problem is the same as the $\mathsf{SUBGRP-NC}[S]$ problem.*

*Proof for Hyperplane Non-cover (*H–$\mathsf{HYP-NC}_{\{\mathcal{V}\}}$*).* Consider the hyperplane non-cover problem ($\mathsf{HYP-NC}$) which is the solvability of homogeneous linear in-equations in Z_p^N. The $\mathsf{HYP-NC}$ problem over Z_3^N includes the $\mathsf{3COL}$ problem and is already NP-hard. Hence, we consider the H–$\mathsf{HYP-NC}_{\{\mathcal{V}\}}$ problem over Z_2^N. In this setting, $\bigcup\mathsf{HYP-NC}$ problem becomes equivalent to the $\mathsf{SUBGRP-NC}[S]$ problem where S is the set of all possible subgroups of Z_2^N. This problem becomes NP-hard as the it includes non-cover by subgroups of index 4 which encompasses the $\mathsf{4COL}$ problem. Hence, the former will be NP-hard using Theorem 3.1. □

(c) Unique games (H–$\mathsf{UG}[k]_{\{\mathcal{V}\}}$ for $k \geq 3$)

Proof. $\mathsf{UG}[3]$ is a CSP with $w = 3$, $q = 2$, and $\mathcal{R} = \{\pi : [\![3]\!] \to [\![3]\!] \mid \pi \text{ is a permutation}\}$. Let

$$R^{\circ} := \bigcup_{\pi:(\forall i)(\pi(i)\neq i)} \pi.$$

Note that $R^{\circ} \in \bigcup \mathcal{R}$. Choosing R° as the constraint for every edge gives us the $\mathsf{3COL}$ problem. Hence, from Theorem 3.1(b), H–$\mathsf{UG}[3]_{\{\mathcal{V}\}}$ is NP-hard. □

Remark 5.6. *Our proof method also shows that* H–$\mathsf{UG}[k]_{\{\mathcal{V}\}}$ *is* NP*-hard for any* $k > 2$.

(d) k-clique Isomorphism (H–$\mathsf{UG}[k]_{\{\mathcal{V}\}}$ for $k \geq 3$)

Proof. If we omit Constraint (1), i.e., replace it by the constraint $\forall (i,j) \in E,\ \ R_{\leq k} \vee \neg R_{\leq k}$, we obtain the $k\mathsf{CLQ}$ problem (deciding whether the graph contains a k-clique) which is NP-hard. Hence from Theorem 3.1(b), the H–$k\mathsf{CLQ}$–$\mathsf{ISO}_{\{\mathcal{V}\}}$ problem is NP-hard. □

(e) Equality to a k-Clique (H–$\mathsf{EQ}_{k\mathsf{CLQ}\{\mathcal{V}\}}$)
We use the framework defined in the previous proof for the H–$\mathsf{EQ}_{\mathsf{ST}\{\mathcal{V}\}}$ problem.

Proof. As mentioned in the previous remark, consider $\mathcal{K}$ to be the set of all possible k-cliques on n vertices and $E_1 = E_2$ parametrized by the hidden input. For this $\mathcal{K}$, the $\bigcup \mathsf{EQ}_{\mathcal{K}}$ problem now becomes equivalent to finding a k-clique on E_2 which is NP-hard. □

Remark 5.7. *The above proof also serves as an alternate proof for Theorem 5.2(d).*

(f) Equality to a Hamiltonian Cycle (H–$\mathsf{EQ}_{\mathsf{HAMC}\{\mathcal{V}\}}$)
We use the framework defined in the previous proof for the H–$\mathsf{EQ}_{\mathsf{ST}\{\mathcal{V}\}}$ problem.

Proof. Here, $\mathcal{K}$ is the set of all possible Hamiltonian Cycles on n vertices and E_1 without loss of generality is a Hamiltonian Cycle $\overline{C} = \{1, 2, \ldots, n\}$. E_2 is any arbitrary graph containing E_1. In this case, the $\bigcup \mathsf{EQ}_{\mathcal{K}}$ problem becomes equivalent to deciding if E_2 has a Hamiltonian Cycle, which is NP-hard. □

This completes the proof for Theorem 5.2 □

6 Constraint-index and Relation-index Revealing Oracle

Theorem 6.1. *Let* S *be a* CSP *with constant arity and alphabet size* w*. Assume that for every* $\alpha \in [\![w]\!]$*, there is a non-empty relation* $R_\alpha \in \mathcal{R}$ *such that* $(\alpha, \ldots, \alpha) \notin R$*. Then,* H–$\mathrm{S}_{\{\mathcal{R}\}}$ *is* NP*-hard.*

Proof. We show that E–S is NP-hard. We will reduce to it the problem E–3SAT which consists of those instances of 3SAT where in each clause either every variable is positive, or every variable is negated. That MONSAT is NP-complete can be deduced, for example, from Schaefer's characterization [Sch78].

Let $q' = (w-1)q+1$ and let $\mathcal{R}' \subseteq [\![w]\!]^{q'}$ be the set of q'-ary relations that can be obtained as an extension of an element of $\mathcal{R} \setminus \{\varnothing\}$ from any q coordinates. Since q and w are constant, the cardinality of $\mathcal{R}'$ is also constant. We claim that $\bigcap_{R\in\mathcal{R}'} R = \varnothing$. Indeed, every $a \in [\![w]\!]^{q'}$ has a subsequence $(\alpha,\dots,\alpha)$ of length q for some $\alpha \in [\![w]\!]$, therefore the extension of R_α from these q coordinates does not contain a. Let $\{R^0, R^1, \dots, R^h\}$ be a minimal subset of $\mathcal{R}'$ such that $\bigcap_{i=0}^{h} R^i = \varnothing$. Since the empty relation is not in $\mathcal{R}'$, we have $h \geq 1$. Let us set $A^0 = \bigcap_{i\neq 1} R^i$ and $A^1 = \bigcap_{i\neq 0} R^i$. Then $A^0 \bigcap A^1 = \varnothing$, and because of the minimality condition, $A^0 \neq \varnothing$ and $A^1 \neq \varnothing$. For a boolean variable x, we will use the notation $x^1 = x$ and $x^0 = \bar{x}$. The main idea of the proof is to encode a boolean variable x^1 by the relation A^1 and x^0 by A^0. We think about the elements of A^1 as satisfying x^1, and about the elements of A^0 as satisfying x^0. Then x^1 and x^0 can be both satisfied, but not simultaneously.

We suppose without loss of generality that ℓ is a multiple of q', and we set $\ell' = \ell/q'$. Since q' is constant, MONSAT on ℓ' variables is still NP-hard. We take ℓ' pairwise disjoint blocks of size q' of the index set $[\ell]$ and on each block we consider relations $R^0,\dots,R^h$. We denote by R^i_k the ℓ-ary relation which is obtained by extending R^i from the kth block. Observe that the relations R^i_k are just extensions of elements of $\mathcal{R}$.

Let $K = \bigwedge_{t=1}^{u} K_t$ be an instance of E–3SAT in ℓ' variables, with each 3-clause of the form $K_t = x_{t_1}^{b_t} \vee x_{t_2}^{b_t} \vee x_{t_3}^{b_t}$, where t_1, t_2, t_3 are indices from $[\ell']$ and b_t is either 0 or 1. Then we map K to the instance $\mathcal{C}$ whose constraints are

$$R_{t_1}^{b_t} \cup R_{t_2}^{b_t} \cup R_{t_3}^{b_t},$$

for each $t \in [u]$, and

$$C_k^i = R_k^i,$$

for each $k \in [\ell']$ and $i \in \{2,\dots,h\}$. This is an instance of MONSAT since the three relations $R_{t_1}^{b_t}$, $R_{t_2}^{b_t}$ and $R_{t_3}^{b_t}$ are the extensions of the same relation in $\mathcal{R}$. It is quite easy to see that K is satisfiable if and only if $\mathcal{C}$ is satisfiable. Indeed, a satisfying assignment a for the $\mathcal{C}$ can be translated to a satisfying assignment for K by assigning 0 or 1 to x_k according to whether the kth block of a was in A_k^0 or A_k^1 (taking an arbitrary value if it was in none of the two). Similarly, a satisfying assignment b for K can be translated to a satisfying assignment a for $\mathcal{C}$ by picking any element of $A_k^{b_k}$ for the kth block of a. □

An immediate consequence is that under the same conditions H–$S_\varnothing$ is NP-hard too. For an application of this consequence, let LINEQ stand for the CSP in which that alphabet is identified with a finite field F and the ℓ-ary constraints are linear equations over F.

Claim 6.2. H–$\mathsf{LINEQ}_\varnothing$ *is* NP*-hard.*

Proof. For each $i \in \ell$, we pick two equations: $x_i = 0$ and $x_i = 1$. Observe that $x_i = 0$ is the same as $\{0\}^i$, the ℓ-ary extension of the unary relation $\{0\}$ on the ith position and we have the same if we replace 0 by 1. By the above observation, the H–CSPs built from relations of these type are NP-hard. □

7 Hidden CSPs with Promise on Instances

In this section we consider an extension of the H–CSP framework where the instances satisfy some property. For the sake of simplicity, we develop this subject only for the constraint index revealing model. Formally, let S be a CSP, and let PROM be a subset of all instances. Then S with *promise* PROM is the CSP $\mathrm{S}^{\mathsf{PROM}}$ whose instances are only elements of PROM. One such property is *repetition freeness* where the constraints of an instance are pairwise distinct. We denote by RF the subset of instances satisfying this property. For example $\mathsf{1SAT}^{\mathsf{RF}}$, (as well as H–$\mathsf{1SAT}^{\mathsf{RF}}$) consists of pairwise distinct literals. Such a requirement is quite natural in the context of certain graph problems where the constraints are inclusion (or non-inclusion) of possible edges The promise H–CSPs framework could also be suitable for discussing certain graph problems on special classes of graphs (e.g, connected graphs, planar graphs, etc.).

We would like to prove an analog of the transfer theorem with promise. Let us be given a promise PROM for the CSP S of type $\mathcal{R} = \{R_1, \ldots, R_s\}$. The corresponding promise $\bigcup\mathsf{PROM}$ for $\bigcup$S is defined quite naturally as follows. We say that an instance $\mathcal{C} = (C_1, \ldots, C_m)$ of S, where $C_j = R_{k_j}(x_{j_1}, \ldots, x_{j_q})$, is *included* in an instance $\mathcal{C}' = (C'_1, \ldots, C'_m)$ of $\bigcup$S if for every $j = 1, \ldots, m$ $C'_j = Q_j(x_{j_1}, \ldots, x_{j_q})$ for $Q_j \in \bigcup\mathcal{R}$ such that $R_{k_j} \subseteq Q_j$. Then $\bigcup\mathsf{PROM}$ is defined as the set of instances in $\mathcal{C}' \in \bigcup$S which include $\mathcal{C} \in \mathsf{PROM}$. In order for the transfer theorem to work, we relax the notion of a solution. A *solution under promise* for $\mathcal{C}' \in \bigcup\mathsf{PROM}$ has to satisfy two criteria: it is a satisfying assignment when $\mathcal{C}'$ includes a satisfiable instance $\mathcal{C} \in \mathsf{PROM}$, and it is EXCEPTION when $\mathcal{C}'$ is unsatisfiable. However, when all the instances $\mathcal{C} \in \mathsf{PROM}$ included in $\mathcal{C}'$ are unsatisfiable but $\mathcal{C}'$ is still satisfiable, it can be either a satisfying assignment or EXCEPTION. We say that an algorithm *solves* $\bigcup \mathrm{S}^{\bigcup\mathsf{PROM}}$ *under promise* if $\forall \mathcal{C}' \in \bigcup\mathsf{PROM}$, it outputs a solution under promise.

Using the above definition in the transfer theorem's proof allows the algorithm for H–$\mathrm{S}_{\{\mathcal{V}\}}$ to terminate, at any moment of time, with the conclusion NO as soon as it gets enough information about the instance to exclude satisfiability and without making further calls to the revealing oracle. In some ambiguous cases, it can still call the oracle with an assignment which satisfies the $\bigcup$S-instance. Other cases when the satisfiability of a $\bigcup$S-instance with promise implies the existence of a satisfiable promise-included instance lack this ambiguity. With these notions the proof of Theorem 3.1 goes through and we obtain the following.

Theorem 7.1. *Let* $\mathrm{S}^{\mathsf{PROM}}$ *be a promise* CSP. *(a) If* $\bigcup S^{\bigcup\mathsf{PROM}}$ *is solvable under promise in time* T *then* H–$\mathrm{S}^{\mathsf{PROM}}_{\{\mathcal{V}\}}$ *is solvable in time* $O((T + s \times \mathrm{comp}(\mathcal{R})) \times m \times \min\{\dim(\bigcup\mathcal{R}), |W_q|\})$.
(b) If H–$\mathrm{S}^{\mathsf{PROM}}_{\{\mathcal{V}\}}$ *is solvable in time* T *then* $\bigcup S^{\bigcup\mathsf{PROM}}$ *is solvable under promise in time* $O(T \times m \times \mathrm{comp}(\bigcup\mathcal{R}))$.

We apply Theorem 7.1 to the following problems: (1) H–$\mathsf{1SAT}^{\mathsf{RF}}_{\varnothing}$, repetition free H–1SAT; (2) H–$\mathsf{2SAT}^{\mathsf{RF}}_{\varnothing}$, repetition free H–2SAT; (3) H–$\mathsf{2COL}^{\mathsf{RF}}_{\varnothing}$, repetition free H–2COL; (4) H–$k\mathsf{WEIGHT}^{\mathsf{RF}}_{\varnothing}$ the repetition free hidden version of the following problem. Informally, the problem kWEIGHT decides if a 0-1 string has Hamming weight at least k. The type of kWEIGHT can be described as follows. We have $w = 2$, $q = 1$ and $\mathcal{R} = \{\{0\}\}$ and W consists of words of length ℓ having Hamming weight k. An instance of kWEIGHT is a collection $(C_1, \ldots, C_m)$ of constraints of the form $x_{i_j} = 0$ (formally, $C_j = \{0\}^{i_j}$). (The string behind these constraints is b where $b_t = 0$ if and only if $t \in \{i_1, \ldots, i_m\}$.) In a repetition free instance we have $|\{i_1, \ldots, i_m\}| = m$.

Theorem 7.2. *(a) Repetition free* H–$\mathsf{1SAT}_{\varnothing}$ *with constraint index revealing oracle is easy, that is* H–$\mathsf{1SAT}^{\mathsf{RF}}_{\varnothing} \in$ P. *(b)* H–$k\mathsf{WEIGHT}_{\varnothing}$ *is* NP*-hard for certain* k*, but* H–$k\mathsf{WEIGHT}^{\mathsf{RF}}_{\varnothing} \in$ P *for every* k.

(c) Repetition free H–2SAT*, with constraint index revealing oracle is easy, that is,* $\text{H–2SAT}^{\mathsf{RF}}_{\varnothing}$ *is* NP*-hard. (d)Repetition free* H–2COL*, that is* $\text{H–2COL}^{\mathsf{RF}}_{\varnothing}$ *is* NP*-hard.*

Proof. We prove each part of the theorem separately:

(a) We consider every literal as its extended n-ary relation where n is the number of variables. This transforms the $\varnothing$-oracle into a $\{\mathcal{V}\}$-oracle. A repetition free instance of $\bigcup\mathsf{1SAT}$ is $\mathcal{C} = \{C_1, \ldots, C_m\}$, where each C_j is a disjunction of literals from $\{x_1, \overline{x}_1, \ldots, x_n, \overline{x}_n\}$ such that there exist m distinct literals $z_1, \ldots, z_m$ with z_j from C_j. A conjunction of literals is satisfiable, if for every $i \in [n]$, the literals x_i and $\overline{x}_i$ are not both among them. Hence an algorithm which solves $\text{H–1SAT}^{\mathsf{RF}}_{\varnothing}$ under promise can proceed as follows. Using a maximum matching algorithm it selects pairwise different variables $x_{i_1}, \ldots, x_{i_m}$ such that hat x_{i_j} or $\overline{x}_{i_j}$ is in C_j. If such a selection is not possible it returns EXCEPTION. Otherwise it can trivially find a satisfying assignment.

(b) Recall that an instance of $k\mathsf{WEIGHT}$ is a collection $\{C_1, \ldots, C_m\}$ of constraints of the form $x_{i_j} = 0$ (formally, $C_j = \{0\}^{i_j}$). (The string behind these constraints is b where $b_t = 0$ if and only if $t \in \{i_1, \ldots, i_m\}$.) Again, we consider the ℓ-ary relations so that the $\varnothing$-oracle is transformed into a $\{\mathcal{V}\}$-oracle.

An instance of $\bigcup k\mathsf{WEIGHT}$ is $\mathcal{C}' = \{C'_1, \ldots, C'_m\}$, where there exist subsets $S_1, \ldots, S_m$ of $[\ell]$ such that the relation for C_j is the set $\{a \in [\![w]\!]^\ell : a_i = 0 \text{ for some } i \in S_j\}$. Finding a satisfying instance of $\bigcup\mathcal{R}$ is therefore equivalent to finding a hitting set (a transversal) of size (at most) $\ell - k$ for the hypergraph $\{S_1, \ldots, S_m\}$. This problem is NP-hard for, say, $0.01\ell < k < 0.99\ell$.

A $k\mathsf{WEIGHT}^{\mathsf{RF}}$-instance included in an instance of $\bigcup k\mathsf{WEIGHT}^{\cup\mathsf{RF}}$ corresponding to subsets $S_1, \ldots, S_m$ consists of constraints $x_{i_j} \neq 0$ for m different indices $i_1, \ldots, i_m$ with $i_j \in S_j$. Obviously, such a set of constraints is satisfiable by an element of W if and only if $m \leq \ell - k$. These observations immediately give the following efficient solution under promise for $\bigcup k\mathsf{WEIGHT}^{\cup\mathsf{RF}}$. If $m > \ell - k$ we return EXCEPTION. Otherwise, using a maximum matching algorithm we find m different places $i_1, \ldots, i_m$ with $i_j \in S_j$ (which must exist by the promise) and return an assignment from W which can be found in an obvious way.

(c) Again, in order to work in the framework of a $\{\mathcal{V}\}$-oracle rather than a $\varnothing$-oracle, we consider every clause as its extended n-ary relation where n is the number of variables. This transforms the $\varnothing$-oracle into a $\{\mathcal{V}\}$-oracle. We reduce 3SAT to $\bigcup\mathsf{2SAT}^{\cup\mathsf{RF}}$ as follows. Let $\phi = \bigwedge_{t=1}^{m} C_j$ be a $3-\mathsf{CNF}$ where

$$C_j = x_{j_1(t)}^{b_1(t)} \vee x_{j_2(t)}^{b_2(t)} \vee x_{j_3(t)}^{b_3(t)}.$$

(Here $b_i(t) \in \{0,1\}$ and x^1 denotes x, x^0 stands for $\overline{x}$.) For each $t = 1, \ldots, m$ we introduce a new variable y_t. We will have $2m$ new clauses:

$$C'_j = x_{j_1(t)}^{b_1(t)} \vee x_{j_2(t)}^{b_2(t)} \vee x_{j_3(t)}^{b_3(t)} \vee y_t^0 \quad \text{and} \quad C''_j = x_{j_1(t)}^{b_1(t)} \vee x_{j_2(t)}^{b_2(t)} \vee x_{j_3(t)}^{b_3(t)} \vee y_t^1$$

for each t. Put $\phi' = \bigwedge_{t=1}^{m}(C'_j \wedge C''_j)$. Then ϕ is satisfiable if and only if ϕ' is satisfiable. In fact, there is a 1 to 2^m correspondence between the assignments satisfying ϕ and those satisfying ϕ': only the values assigned to the first ℓ variables matter. Also, the included constraints $(x_{j_1(t)}^{b_1(t)} \vee y_t^1)$ and $(x_{j_1(t)}^{b_1(t)} \vee y_t^0)$ for all $t = 1, \ldots, m$ form a system of $2m$ different $2-\mathsf{CNF}$s. Furthermore, if ϕ' is satisfied by an assignment then we can select a satisfiable system of $2m$

pairwise distinct sub-constraints: for each t we pick $s \in \{1,2,3\}$ such that $x^{b_s}_{j_s(t)}$ is evaluated to 1 and take $(x^{b_s}_{j_s(t)} \vee y^1_t)$ and $(x^{b_s}_{j_s(t)} \vee y^0_t)$ for $t = 1, \dots, m$.

(d) Here the alphabet is $[\![2]\!] = \{0,1\}$, $q = 2$, $\mathcal{R}$ has one element "$\neq$", that is $\{(1,0),(0,1)\}$. An instance of $\mathsf{2COL}^{\mathsf{RF}}$ consists of a set of constraints of the form $x_u \neq x_v$ for m pairwise distinct unordered pairs $\{u,v\}$ from $\{1,\dots,\ell\}$ (corresponding to the edges of a graph). (Here we again work in the context of the extensions of the relation "$\neq$" to arity $\ell = n$.)

An instance of $\bigcup\mathsf{2COL}$ is a collection $\{C_1,\dots,C_m\}$, where each C_j is a disjunction of constraints of the form $x_u \neq x_v$. In an equivalent view, an instance of $\bigcup\mathsf{2COL}$ can be described by the collection of edge sets (graphs) $E_1,\dots,E_m$ on vertex set $[n]$ and a satisfying assignment can be described by a coloring $c : \{1,\dots,n\} \to \{0,1\}$ such that for every j there exists an edge $e_j \in E_j$ with endpoints having different colors. It is clear that if the edge sets $E_1,\dots,E_m$ are disjoint then the instance is repetition free and the solutions under promise coincide with the solutions in the normal sense.

Let $E_1,\dots,E_m$ be edge sets describing an instance of $\bigcup\mathsf{2COL}$. Put $s_j = |E_j|$. For each j we introduce $2s_j$ new vertices: $uvj1, uvj2$ for each $\{u,v\} \in E_j$, $2s_j$ new one-element edge sets $E_{uvj1} = \{\{u, uvj1\}\}$ and $E_{uvj2} = \{\{v, uvj2\}\}$; while E_j is replaced with an edge E'_j set consisting of s_j edges: $\{uvj1, uvj2\}$ for each $\{u,v\} \in E_j$. It turns out that the $\bigcup\mathsf{2COL}$ problem on the $n + 2\sum_{j=1}^m s_j$ vertices with the new $m + 2\sum_{j=1}^m s_j$ edge sets is equivalent to the original one and solutions of the two problems can be easily (and efficiently) mapped to each other. We have that the new edge sets are pairwise disjoint and hence the repetition free version of the new $\bigcup\mathsf{2COL}$ is the same as the the non-promise version.

Theorem 6.1 shows that non-promise $\bigcup\mathsf{2COL}$ is NP-hard. By the reduction above, so is its repetition free version.

□

On group isomorphisms

Isomorphism of a hidden multiplication table with a given group, a problem discussed in [BCZ13a], can also be cast in the framework of promise H–CSPs. We consider the following problem GROUPEQ (equality with a group from a class). Let $\mathcal{G}$ be a family of groups on the set $[k]$, that is, a set of multiplication tables on $[k]$ such that each multiplication defines a group. The task is to decide whether a hidden group structure $b(\,,\,)$ is equal to some $a(\,,\,)$ from $\mathcal{G}$ and if yes, find such an $a(\,,\,)$. (Note that a solution of the latter task will give the whole table for $b(\,,\,)$.) For instance, $\mathcal{G}$ may consist of all isomorphic copies on $[k]$ of a given group G having k elements. In that case, up to possibly an overhead of complexity $k^{O(\log k)}$, the search problem is equivalent to finding an isomorphism between G and the group H given by $[k]$. This is because given a mapping $\phi : G \to [k]$ we can easily compute the unique multiplication table on $[k]$ which defines a group such that ϕ becomes an isomorphism, and, conversely, given a group H with its multiplication table we can find an isomorphism $G \to H$ in time $k^{O(\log k)}$. (Of course, for abelian groups this overhead is polynomial.)

We define $\mathsf{GROUPEQ}(\mathcal{G})$ as a promise CSP as follows. First we consider the CSP $\mathsf{ENTRIES}(\mathcal{G})$ with the following parameters and type. We have $w = k$, $W = \mathcal{G}$, $\mathcal{R} = \{\{w\} : w \in [k]\}$, $\ell = k^2$. It will be convenient to consider assignments as $k \times k$ tables with entries from $[k]$, that is, functions $[k]^2 \to [k]$. (Implicitly, we use a bijection between the index set $\{1,\dots,\ell\}$ and $[k]^2$.) The number of

constrains is $m = k^2$ and an instance is a collection $x_{(u_h,v_h)} = b_h$ $(h = 1,\dots,m)$. Thus the assignment satisfying $\mathsf{ENTRIES}(\mathcal{G})$ are $k \times k$ multiplication tables from $\mathcal{G}$ which have prescribed values at k^2 (not necessarily distinct) places.

We say that an instance for $\mathsf{ENTRIES}(\mathcal{G})$ belongs to the promise GROUP if two conditions are satisfied. Firstly, there is one constraint for the value taken by each place. Formally, the map $\tau : h \mapsto (u_h, v_h)$ is a bijection between $\{1,\dots,m\}$ and $[k]^2$. As a consequence, by setting $b(u,v) := b_{\tau^{-1}(u,v)}$, we have a constraint $x_{u,v} = b(u,v)$ for pair $(u,v) \in [k]^2$. The second – essential – condition is that the multiplication given by $b(\,,\,)$ defines a group structure on $[k]$. The promise problem $\mathsf{GROUPEQ}(\mathcal{G})$ is the problem $\mathsf{ENTRIES}(\mathcal{G})^{\mathsf{GROUP}}$.

We consider the promise problem H–$\mathsf{ENTRIES}(\mathcal{G})^{\mathsf{GROUP}}_{\{V\}}$ (which we denote by H–$\mathsf{GROUPEQ}(\mathcal{G})$ for short) and the corresponding problem $\bigcup\mathsf{ENTRIES}(\mathcal{G})^{\bigcup\mathsf{GROUP}}_{\{V\}}$ (short notation: $\bigcup\mathsf{GROUPEQ}(\mathcal{G})$). In this H–CSP, if $a(\,,\,)$ is different from $b(\,,\,)$, the oracle reveals a pair (u,v) such that $a(u,v) \neq b(u,v)$.

We note here that the case of H–$\mathsf{GROUPEQ}(\mathcal{G})$ where $\mathcal{G}$ consists of all isomorphic copies of a group G in fact covers the problem of finding an isomorphism with G discussed in [BCZ13a] where input to the verification oracle is a bijection $\phi : [k] \to G$ and, in the case when ϕ is not an isomorphism, the oracle has to reveal $u, v \in [k]$ such that, given a binary operation g acting on elements of G, the product $g(\phi(u), \phi(b))$ does not equal $\phi(b(u,v))$ in G. Indeed, given ϕ we can define (and even compute) the multiplication $a_\phi(\,,\,)$ on $[k]$ – by taking $a_\phi(x,y) = \phi^{-1}(g(\phi(x), \phi(y))$ – such that ϕ becomes an isomorphism from the group given by $a_\phi(\,,\,)$ to G. Then ϕ is an isomorphism from the group given by $b(\,,\,)$ if and only if $a_\phi(\,,\,) = b(\,,\,)$ and if it is not the case then the oracle given in [BCZ13a] reveals a pair (u,v) such that $a_\phi(u,v) = b(u,v)$, exactly what is expected from a revealing oracle for H–$\mathsf{GROUPEQ}(\mathcal{G})$.

An instance of $\bigcup\mathsf{ENTRIES}(\mathcal{G})$ consists of k^2 constraints expressing that $a(u_h, v_h) \in S_h$ where $S_h \in 2^{[k]} \setminus \varnothing$ for $h = 1,\dots,m = k^2$. An instance of the promise version $\bigcup\mathsf{GROUPEQ}(\mathcal{G})$ $(= \bigcup\mathsf{ENTRIES}(\mathcal{G})^{\bigcup\mathsf{GROUP}}_{\{V\}})$ should satisfy that $\{(u_h, v_h) : h = 1,\dots,m\} = [k]^2$, that is, our constraints are actually $x_{(u,v)} \in S(u,v)$ for a map $S(\,,\,) : [k]^2 \to 2^{[k]}$, and, further, there is a map $b(\,,\,) : [k]^2 \to [k]$ with $b(u,v) \in S(u,v)$ for every $(u,v) \in [k]^2$ such that $b(\,,\,)$ gives a group structure.

Assume now that $k = p$, a prime. It is natural to choose $\mathcal{G}$ as the set of all group structures on $[k]$. As every group having p elements is isomorphic to Z_p, $\mathcal{G}$ coincides with the group structures on $[k]$ isomorphic to Z_p. We can translate the arguments given in [BCZ13a] to show that already this case of $\bigcup\mathsf{GROUPEQ}$ is NP-hard as follows. Let $([k], E)$ be a Hamiltonian directed graph (without loops) on $[k]$. Fix $z \in [k]$ and for $u \in [k]$ let $S(u,z) = \{v : (u,v) \in E\}$ and $S(u,v) = [k]$ for $v \neq z$. Let $\phi : [k] \to \{0,\dots,p-1\} = Z_p$ be a bijection that defines a Hamiltonian cycle in $([k], E)$. Then $b(x,y) = a_\phi(x,y) := \phi^{-1}(\phi(x) + \phi(y))$ gives a group structure on $[k]$ (isomorphic to Z_p via ϕ) consistent with the constraints given by $S(\,,\,)$. Conversely, if $b(\,,\,)$ gives a group structure (necessarily isomorphic to Z_p) consistent with $S(\,,\,)$ then the pairs $(u, b(u,z))$ $(u \in [k])$ form a Hamiltonian cycle in $([k], E)$. Thus finding Hamiltonian cycles in Hamiltonian digraphs on p points can be reduced to $\bigcup\mathsf{GROUPEQ}$ on p elements.

References

[BCD+13] Xiaohui Bei, Ning Chen, Liyu Dou, Xiangru Huang, and Ruixin Qiang. Trial and error in influential social networks. In Inderjit S. Dhillon, Yehuda Koren, Rayid Ghani, Ted E. Senator, Paul Bradley, Rajesh Parekh, Jingrui He, Robert L. Grossman, and Ramasamy Uthurusamy, editors, *KDD*, pages 1016–1024. ACM, 2013.

[BCZ13a] Xiaohui Bei, Ning Chen, and Shengyu Zhang. On the complexity of trial and error. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *STOC*, pages 31–40. ACM, 2013.

[BCZ13b] Xiaohui Bei, Ning Chen, and Shengyu Zhang. Solving linear programming with constraints unknown. *CoRR*, abs/1304.1247, 2013.

[GJ79] M. R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.

[HKKK88] Pavol Hell, David G. Kirkpatrick, Jan Kratochvíl, and Igor Kríz. On restricted two-factors. *SIAM J. Discrete Math.*, 1(4):472–484, 1988.

[KLN91] Jan Kratochvíl, Anna Lubiw, and Jaroslav Nešetřil. Noncrossing subgraphs in topological layouts. *SIAM J. Discret. Math.*, 4(2):223–244, March 1991.

[Sch78] Thomas J. Schaefer. The complexity of satisfiability problems. In Richard J. Lipton, Walter A. Burkhard, Walter J. Savitch, Emily P. Friedman, and Alfred V. Aho, editors, *STOC*, pages 216–226. ACM, 1978.

ECCC ISSN 1433-8092

http://eccc.hpi-web.de