On the complexity of trial and error for constraint satisfaction problems

Gábor Ivanyos[1], Raghav Kulkarni[2], Youming Qiao[24], Miklos Santha[23], and Aarthi Sundaram[2]

[1] Institute for Computer Science and Control, Hungarian Academy of Sciences, Budapest, Hungary Gabor.Ivanyos@sztaki.mta.hu
[2] Centre for Quantum Technologies, National University of Singapore kulraghav@gmail.com, aarthims@nus.edu.sg
[3] LIAFA, Univ. Paris 7, CNRS, 75205 Paris, France miklos.santha@liafa.univ-paris-diderot.fr
[4] Centre for Quantum Computation and Intelligent Systems University of Technology, Sydney jimmyqiao86@gmail.com

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.

1 Introduction

In [2], 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 [3], and to study information diffusion in a social network in [1].

As mentioned, in [2] 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. The optimal algorithm minimizes 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 [2], 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. Based on this framework we define three types of revealing oracles to generalize the model of [2]. 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 [2], 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. 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. We omit 2SAT as its hardness is implied from that of 1SAT.

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 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 $\mathcal{U} = \emptyset$ 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 (note that the latter does not necessarily imply that S is 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 include as possibilities for the specified clause, all those relations which were violated at the specified indices of the proposed assignment, and remove 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 all its possibilities. This naturally brings us to consider the CSP $\bigcup \mathrm{S}$, 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 distinct 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–S$_{\{\mathcal{V}\}}$ (resp. H–S$_{\{\mathcal{R}\}}$). In fact, with $\bigcup$E–S we can solve H–S$_\emptyset$. 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–S$_{\{\mathcal{V}\}}$ *and* $\bigcup$S*, (b)* H–S$_{\{\mathcal{R}\}}$ *and* E–S*, (c)* H–S$_\emptyset$ *and* $\bigcup$E–S*.*

The precise dependence on the parameters can be found in the theorems of Section 3 and Corollary 1 highlights the conditions for polynomial equivalence.

Example 1 continued. Since $\bigcup\{\mathsf{Id}, \mathsf{Neg}\} = \{\emptyset, \mathsf{Id}, \mathsf{Neg}, \{0,1\}\}$, $\bigcup$1SAT has only the two trivial (always false or always true) relations in addition to the relations in 1SAT. Therefore it can be solved in polynomial time, and by the the Transfer Theorem H–1SAT$_{\{\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 [6], and therefore the Transfer Theorem implies that H–1SAT$_{\{\mathcal{R}\}}$ and H–1SAT$_\emptyset$ 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 S$^{\mathsf{PROM}}$ the promise problem whose instances are instances of S satisfying PROM. The problem H–S$^{\mathsf{PROM}}_{\{\mathcal{U}\}}$ is defined in an analogous way from H–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* $\bigcup$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* $\text{H–S}^{\mathsf{PROM}}_{\{\mathcal{V}\}}$ *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 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 $\text{H–1SAT}^{\mathsf{RF}}_{\emptyset}$, hidden repetition-free 1SAT with $\emptyset$-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 $\emptyset$-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 discussed as above. Interestingly, many of the graph properties mentioned in the last paragraph are no longer NP-hard but 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 [2, Theorem 13], whose proof, with the help of the Transfer Theorem, becomes very simple.

With constraint and relation index revealing oracles, we show a dichotomy theorem similar to results obtained in [4, 5] for any CSP with constant arity and alphabet size: if some string of the form $(\alpha, \ldots, \alpha)$ satisfies all the non-empty relations then the problem is in P, otherwise it is 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, in this setting 2SAT is still NP-hard. The group isomorphism problem can also be cast in this framework, and we give a simplified proof of [2, 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 $\emptyset$-revealing, $\{\mathcal{V}\}$-revealing and $\{\mathcal{R}\}$-revealing oracles respectively. Finally in Section 7 we present the results for hidden promise CSPs. Most proofs are omitted from this version of the paper due to space constraints.

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 $\forall \pi \in S_{\ell(n)}$, 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 except the number of relations which is at most 2^s, and whose type is $\bigcup\mathcal{R}$. We remark that $\dim(\bigcup\mathcal{R}) \leq \min\{|\mathcal{R}|, |W_q|\}$.

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–$\mathcal{R} = \{R^{(j_1,\ldots,j_q)} \ : \ R \in \mathcal{R}$ 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, and the number of relations which becomes at most $s\frac{\ell!}{(\ell-q)!}$. 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 $\emptyset$-revealing oracle $\mathsf{V}_\emptyset$ 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–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–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. We will only give the proof for the case of the $\{\mathcal{V}\}$-revealing oracle below owing to space constraints.

Theorem 1. *(a) If* $\bigcup \mathrm{S}$ *is solvable in time* T *then* H–$\mathrm{S}_{\{\mathcal{V}\}}$ *is solvable in time* $O((T + s \times \mathrm{comp}(\mathcal{R})) \times m \times \dim(\bigcup \mathcal{R})$. *(b) If* H–$\mathrm{S}_{\{\mathcal{V}\}}$ *is solvable in time* T *then* $\bigcup \mathrm{S}$ *is solvable in time* $O(T \times m \times \mathrm{comp}(\bigcup \mathcal{R}))$.

Theorem 2. *(a) If* E–S *is solvable in time* T *then* H–$\mathrm{S}_{\{\mathcal{R}\}}$ *is solvable in time* $O((T + |[\ell]^{(q)}| \times \mathrm{comp}(\mathcal{R})) \times m \times |[\ell]^{(q)}|)$. *(b) If* H–$\mathrm{S}_{\{\mathcal{R}\}}$ *is solvable in time* T *then* E–S *is solvable in time* $O(T \times m \times \mathrm{comp}(\mathrm{E}\text{–}\mathcal{R}))$.

Theorem 3. *(a) If* $\bigcup$ E–S *is solvable in time* T *then* H–$\mathrm{S}_{\emptyset}$ *is solvable in time* $O((T + s \times |[\ell]^{(q)}| \times \mathrm{comp}(\mathcal{R})) \times m \times \dim(\bigcup \mathrm{E}\text{–}\mathcal{R}))$. *(b) If* H–$\mathrm{S}_{\emptyset}$ *is solvable in time* T *then* $\bigcup$ E–S *is solvable in time* $O(T \times m \times \mathrm{comp}(\bigcup \mathrm{E}\text{–}\mathcal{R}))$.

Proof of Theorem 1. 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 $C_j^t = \bigcup_{R \in \mathcal{R}: R \cap A_j^t = \emptyset} R(x_{j_1^t}, \ldots, x_{j_q^t})$ 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 $A_j^1 = \emptyset$ 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 $R \cap A_j^t \neq \emptyset$ 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 $m \times \dim(\mathcal{R})$. 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–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 $\bigcup$S. Let $\mathcal{C} = \{C_1, \ldots, C_m\}$ be an instance of $\bigcup$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 $C_j^t = \bigcup_{R \in \mathcal{R}: R \cap A_j^t = \emptyset} R(x_{j_1^t}, \ldots, x_{j_q^t})$, where the sets $A_j^t \subseteq W_q$ are determined by the result of the tth call to the oracle. Initially $A_j^0 = \emptyset$. 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 $A_j^t \bigcap C_j = \emptyset$ 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. □

Corollary 1. *Let* 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$_{\emptyset}$ *and* $\bigcup$E–S *if both s and q are constant.*

The polynomial time equivalence of Theorems 1, 2, 3 and Corollary 1 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 graph is a monotone Boolean function $\mathcal{P}$ on $\binom{n}{2}$ variables invariant under relabeling of vertices. The CSP $\mathrm{S}_{\mathcal{P}}$ associated with $\mathcal{P}$ has parameters $w = 2,\ q = 1,\ \ell = \binom{n}{2}$, $W_{\mathcal{P}} = \{A \mid A$ is a graph with minimal number of edges satisfying $\mathcal{P}\}$, and $\mathcal{R} = \{\mathsf{Neg}\}$. 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 and to bipartite graphs.

By Theorem 3, $\mathrm{H}\text{–}\mathrm{S}_{\mathcal{P}}$ can be analyzed by considering $\bigcup\mathrm{X}\text{–}\mathrm{S}_{\mathcal{P}}$. We do this for the following: Spanning Tree (ST, the property of being connected), Undirected Cycle Cover (UCC, containing an undirected cycle cover), Undirected Path (UPATH, containing an undirected path between s and t), Bipartite Perfect Matching (BPM, having a perfect matching in a bipartite graph), Directed Spanning Tree (DST), Directed Cycle Cover (DCC), and Directed Path (DPATH).

Theorem 4. *In the monotone graph property framework for the hidden model using constraint-index revealing oracle the following properties are* NP*-hard:* ST, DST, UCC, DCC, BPM, DPATH, UPATH.

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: **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 $R_{abc}(x, y, z) := (x = a) \wedge (y = b) \wedge (z = c)$; **Hyperplane Non-cover** ($\mathsf{HYP-NC}$): Given a group Z_p^N, the hyperplane non-cover problem is the solvability of a system of homogeneous linear in-equations in Z_p^N; **Arbitrary sets of binary relations on Boolean alphabet,** in particular, the 2-SAT Problem ($\mathsf{2SAT}$); **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 $\pi_e(\alpha_u) = \alpha_v$; **$k$-Clique Isomorphism** ($k\mathsf{CLQ}$–$\mathsf{ISO}$): Given an undirected graph $G = (V, E)$, does there exist a permutation π on $[n]$ such that: (a) $\forall(i, j) \in E,\ \ \pi(i), \pi(j) \leq k$, (b) $\forall(i, j) \notin E,\ \ \pi(i) > k$ or $\pi(j) > k$; **Polynomial time Solvable Graph Properties** ($\mathcal{P}^{poly}$): The framework for graph properties as defined in Section 4, but with $\{\mathcal{V}\}$-revealing oracle; **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 $\mathcal{K}$. For example, Equality to k-Clique ($\mathsf{EQ}_{k\mathsf{CLQ}}$), Equality to Hamiltonian Cycle ($\mathsf{EQ}_{\mathsf{HAMC}}$), and Equality to Spanning Tree ($\mathsf{EQ}_{\mathsf{ST}}$).

Theorem 5. *(a) The following problems in the hidden setting with constraint-index and variable-index revealing oracle are in polynomial time:* 2SAT, UG[2], $\mathcal{P}^{poly}$, $\mathsf{EQ}_{\mathsf{ST}}$. *(b) The following problems in the hidden setting with constraint-index and variable-index revealing oracle are* NP*-hard:* Δ, HYP$-$NC, UG[k] *for* $k \geq 3$, kCLQ–ISO, $\mathsf{EQ}_{k\mathsf{CLQ}}$, $\mathsf{EQ}_{\mathsf{HAMC}}$.
Remarks. (1) Polynomial-time solvable graph properties are in P this time, in contrast to the NP-hardness result when only constraint index is revealed (Theorem 4). (2) UG[k] for $k = 2$ is in P, while for $k \geq 3$ it is NP-hard.

6 Constraint-index and Relation-index Revealing Oracle

Theorem 6. *Let* S *be a* CSP *with constant arity and alphabet size* w. *If for every* $\alpha \in [\![w]\!]$, *there is a non-empty relation* $R \in \mathcal{R}$ *such that* $(\alpha, \ldots, \alpha) \notin R$, *then* H–$\mathrm{S}_{\{\mathcal{R}\}}$ *is* NP*-hard; otherwise* H–$\mathrm{S}_{\{\mathcal{R}\}}$ *is (trivially) in* P.
Remark. Under the same conditions H–$\mathrm{S}_{\emptyset}$ is NP-hard. As an application, let LINEQ be the CSP in which that alphabet is identified with a finite field F and the ℓ-ary constraints are linear equations over F. Then H–$\mathsf{LINEQ}_{\emptyset}$ is 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 $\bigcup$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 $j = 1, \ldots, m$ $C'_j = R'_j(x_{j_1}, \ldots, x_{j_q})$ for $R'_j \in \bigcup\mathcal{R}$ such that $R_{k_j} \subseteq R'_j$. Then $\bigcup$PROM is defined as the set of instances in $\mathcal{C}' \in \bigcup$S which includes some $\mathcal{C} \in$ PROM. In order for the transfer theorem to work, we relax the notion of a solution. A *solution under promise* for $\mathcal{C}' \in \bigcup$PROM has to satisfy two criteria: it is a satisfying assignment when $\mathcal{C}'$ includes a satisfiable instance $\mathcal{C} \in$ PROM, and it is EXCEPTION when $\mathcal{C}'$ is unsatisfiable. However, when all the instances $\mathcal{C} \in$ 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 $\forall \mathcal{C}' \in \bigcup$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 1 goes through and we obtain the following.

Theorem 7. *Let* $\mathrm{S}^{\mathsf{PROM}}$ *be a promise* CSP. *(a) If* $\bigcup S^{\bigcup \mathsf{PROM}}$ *is solvable under promise in time* T *then* $\mathrm{H\text{–}S}^{\mathsf{PROM}}_{\{\mathcal{V}\}}$ *is solvable in time* $O((T+s\times\mathrm{comp}(\mathcal{R}))\times m\times \min\{\dim(\bigcup\mathcal{R}),|W_q|\})$. *(b) If* $\mathrm{H\text{–}S}^{\mathsf{PROM}}_{\{\mathcal{V}\}}$ *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 to the following problems: $\mathrm{H\text{–}1SAT}^{\mathsf{RF}}_{\emptyset}$, $\mathrm{H\text{–}2SAT}^{\mathsf{RF}}_{\emptyset}$, $\mathrm{H\text{–}2COL}^{\mathsf{RF}}_{\emptyset}$, and $\mathrm{H\text{–}}k\mathsf{WEIGHT}^{\mathsf{RF}}_{\emptyset}$. Informally, the problem $k\mathsf{WEIGHT}$ decides if a 0-1 string has Hamming weight at least k, and $\mathrm{H\text{–}}k\mathsf{WEIGHT}_{\emptyset}$ is NP-hard under the constraint index revealing oracle. Interestingly, in the repetition-free setting, $\mathrm{H\text{–}1SAT}^{\mathsf{RF}}_{\emptyset}$ and $\mathrm{H\text{–}}k\mathsf{WEIGHT}^{\mathsf{RF}}_{\emptyset}$ are in P. On the other hand, $\mathrm{H\text{–}2SAT}^{\mathsf{RF}}_{\emptyset}$ and $\mathrm{H\text{–}2COL}^{\mathsf{RF}}_{\emptyset}$ are still NP-hard. Finally, we give an alternative proof, via Theorem 7, for [2, Theorem 11], showing NP-hardness of the isomorphism problem of a hidden group (specified by its multiplication table) with a given group.

Acknowledgements. We are grateful to an anonymous referee for various insightful comments and improvements. Most of this work was conducted when the authors were at the Centre for Quantum Technologies (CQT) in Singapore, and partially funded by the Singapore Ministry of Education and the National Research Foundation, also through the Tier 3 Grant "Random numbers from quantum processes". Research partially supported by the European Commission IST STREP project Quantum Algorithms (QALGO) 600700, by the French ANR Blanc program under contract ANR-12-BS02-005 (RDAM project), and by the Hungarian Scientific Research Fund (OTKA), Grant NK105645.

References

1. Xiaohui Bei, Ning Chen, Liyu Dou, Xiangru Huang, and Ruixin Qiang. Trial and error in influential social networks. In *KDD*, pages 1016–1024, 2013.
2. Xiaohui Bei, Ning Chen, and Shengyu Zhang. On the complexity of trial and error. In *STOC*, pages 31–40, 2013.
3. Xiaohui Bei, Ning Chen, and Shengyu Zhang. Solving linear programming with constraints unknown. *CoRR*, abs/1304.1247, 2013.
4. Tomás Feder and Moshe Y. Vardi. The computational structure of monotone monadic snp and constraint satisfaction: A study through datalog and group theory. *SIAM J. Comput.*, 28(1):57–104, February 1999.
5. Barnaby Martin. First-order model checking problems parameterized by the model. In *CiE*, pages 417–427, 2008.
6. Thomas J. Schaefer. The complexity of satisfiability problems. In *STOC*, pages 216–226, 1978.