Journal of Computer and System Sciences 92 (2018) 48–64

Contents lists available at ScienceDirect

Journal of Computer and System Sciences

www.elsevier.com/locate/jcss

On the complexity of trial and error for constraint satisfaction problems[☆]

Gábor Ivanyos[a], Raghav Kulkarni[b], Youming Qiao[c,*], Miklos Santha[b,d], Aarthi Sundaram[b]

[a] Institute for Computer Science and Control, Hungarian Academy of Sciences, Budapest, Hungary
[b] Centre for Quantum Technologies, National University of Singapore, Singapore 117543, Singapore
[c] Centre or Quantum Software and Information, University of Technology Sydney, Australia
[d] IRIF, CNRS, Université Paris Diderot, 75205 Paris, France

A R T I C L E   I N F O

Article history:
Received 24 October 2016
Received in revised form 4 June 2017
Accepted 21 July 2017

Keywords:
Trial and error
Constraint satisfaction problems
Computational complexity

A B S T R A C T

In 2013 Bei, Chen and Zhang introduced a trial and error model of computing, 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, by adopting a formal framework for CSPs, and defining several types of revealing oracles. Our main contribution is to develop a *transfer theorem* for each type of the revealing oracle. To any hidden CSP with a specific type of revealing oracle, the transfer theorem associates another 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 to the study of normal CSPs. We apply the transfer theorems to get polynomial-time algorithms or hardness results for several families of concrete problems.

1. Introduction

In [3], 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 [4], and to study information diffusion in a social network in [1].

As mentioned, in [3] 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.

☆ A preliminary report on this work appeared in ICALP 14 as [7].
* Corresponding author.
E-mail addresses: Gabor.Ivanyos@sztaki.mta.hu (G. Ivanyos), kulraghav@gmail.com (R. Kulkarni), jimmyqiao86@gmail.com (Y. Qiao), santha@irif.fr (M. Santha), aarthims@u.nus.edu (A. Sundaram).

http://dx.doi.org/10.1016/j.jcss.2017.07.005
0022-0000/© 2017 Elsevier Inc. All rights reserved.

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 [3], 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 [3]. Let us make a quick remark that, our CSP model has a couple of features that may not be quite standard. We will mention some of these in the following, and discuss these in detail in Section 2.3.

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 [3], 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 $[\![w]\!] = \{0, 1, \dots, w-1\}$ and by $\mathcal{R} = \{R_1, \dots, R_s\}$, a set of relations over $[\![w]\!]$ of some fixed arity q. For a set of variables $\mathcal{V} = \{x_1, \dots, x_\ell\}$, an instance of S is a set of constraints $\mathcal{C} = \{C_1, \dots, C_m\}$, where $C_j = R(x_{j_1}, \dots, x_{j_q})$ 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 may only be interested in those assignments satisfying certain additional conditions that cannot be (easily) expressed in the framework of constraint satisfaction problems. This case happens, say when we look for permutations in isomorphism problems, or when we view monotone graph properties as CSP problems in Section 6. To cover these cases, our model will also include a subset $W \subseteq [\![w]\!]^\ell$ as a parameter and we will look for satisfying assignments from W, whose elements will be refereed to as *admissible assignments*. That these admissible assignments can play a notable role (as in Section 6) is a first feature that may be somewhat surprising.

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–$1\mathrm{SAT}_{\mathcal{U}}$ 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. 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 satisfied at the specified indices of the proposed assignment, but keep as possibilities all the relations which were violated at those indices. Therefore, to find out more information about the input, we would like to find a satisfying assignment for a CSP instance