Theoretical Computer Science ••• (••••) •••–•••

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

Pushing the frontier of minimality

Guillaume Escamocher[*], Barry O'Sullivan

Insight Centre for Data Analytics, University College Cork, Ireland

ARTICLE INFO

Article history:
Received 20 June 2017
Received in revised form 5 December 2017
Accepted 4 June 2018
Available online xxxx
Communicated by V.Th. Paschos

Keywords:
Constraint satisfaction
Minimal CSP
NP-hardness result

ABSTRACT

The Minimal Constraint Satisfaction Problem, or Minimal CSP for short, arises in a number of real-world applications, most notably in constraint-based product configuration. It is composed of the set of CSP problems where every allowed tuple can be extended to a solution. Despite the very restrictive structure, computing a solution to a Minimal CSP instance is NP-hard in the general case. In this paper, we look at three independent ways to add further restrictions to the problem. First, we bound the size of the domains. Second, we define the arity as a function on the number of variables. Finally we study the complexity of computing a solution to a Minimal CSP instance when not just every allowed tuple, but every partial solution smaller than a given size, can be extended to a solution. In all three cases, we show that finding a solution remains NP-hard. All these results reveal that the hardness of minimality is very robust.

1. Introduction

An instance of the Minimal Constraint Satisfaction Problem, or Minimal CSP for short, is a CSP instance where each tuple allowed in a constraint relation is part of at least one solution [9]. Since all Minimal CSP instances are satisfiable, solving such an instance does not refer to the decision problem of determining whether it has a solution, but to the exemplification of a solution.

Minimal CSP is often found 'naturally' in configuration problems [8]. A seller might want to offer its customers a large degree of customization. If, for example, the product sold is a car, some possible options might be the color of the vehicle and whether it is automatic or manual. If after choosing "automatic", "red" remains a valid option for the color parameter, then it is preferable that at least one red automatic car can be configured. The Minimal CSP can answer a number of queries relevant to product configuration in polynomial time [6], such as whether a solution exists that satisfies a given unary constraint, or whether an assignment to k variables is consistent in a Minimal CSP where all constraints are defined over k-tuples of the variables. These queries can be answered simply by inspecting the constraints of the problem instance. However, answering queries over arbitrary assignments to the variables remains hard, which has given rise to many studies of the use of automata and decision diagrams to reason about the solution sets of complex configuration problems [2].

The notion of minimality is related to that of robustness [1,7]. Robust CSP is the problem of determining whether every partial solution of a given size can be extended to a full solution, in effect checking the minimality of an instance. On the other hand, Minimal CSP already assumes that this condition is fulfilled and instead requires to find a solution.

The restrictions defining minimality can be viewed as extreme forms of consistency. The concept of minimality is that all values not belonging to a solution have been pruned, all constraints allowing values that cannot possibly appear in a same

[*] Corresponding author.
E-mail addresses: guillaume.escamocher@insight-centre.org (G. Escamocher), barry.osullivan@insight-centre.org (B. O'Sullivan).

https://doi.org/10.1016/j.tcs.2018.06.008
0304-3975/© 2018 Elsevier B.V. All rights reserved.

solution have been adjusted. Yet, even though minimality offers an abundance of data about the CSP instances it applies to, it turns out that algorithms cannot use this information to significantly distinguish Minimal CSP instances from general CSP ones. Indeed, we prove that when bounding the size of the domains by a constant d and the arity of the constraints by a constant k, the Minimal CSP and the general CSP are NP-hard for the exact same values of d and k.

We also expand the concept of minimality, to study if hardness is conserved. We present the different directions that we considered. Our main result is the one revolving around what seems like the most natural expansion. In our new class of Minimal CSP instances, we significantly increase the number of sets of compatible values that can be extended to a solution. While one may think this leads to triviality, or at least tractability, we show that again no algorithm can exploit this new information in a useful way, unless $P = NP$. The long-term objective of this work is to identify the frontier of intractability for Minimal CSP.

Each of the three next sections of the paper presents a particular way to further restrict the Minimal CSP. In Section 2, we start by formally defining both the general CSP and the Minimal CSP, then proceed to study the complexity of the Minimal CSP when bounding the arity of the constraints and the size of the domains. In particular, we present a complexity classification over these two parameters that extends Gottlob's complexity result [6] to instances with very small domain sizes. The contents of this section have been previously published [5]. In Section 3 we provide a look into the behavior of the Minimal CSP with global constraints. Section 4 focuses on generalizing the core notion of extendable tuple in the definition of Minimality to extendable partial solution. We begin in Section 4.1 by formalizing and illustrating the new concepts that we introduce. Then in Section 4.2 we present the main result of the paper, showing that the inherent hardness of minimality is conserved even with considerable additional restrictions. Finally, we conclude in Section 5 by summarizing our contributions and outlining some future work in this area.

2. Bounding the size of the domains and the arity

The first of our three generalizations of minimality deals with the arity and size parameters. Before presenting our complexity proofs, we start by formally defining the core notions of the Constraint Satisfaction Problem. In particular, we highlight the fact that we do not view the constraints of a CSP instance as a list of forbidden tuples, as is standard in the constraint literature, but instead as the complete specification of the value of every tuple of size smaller or equal than the arity, where the value here means either allowed or forbidden. Our main reason for doing so is to emphasize the role of allowed tuples, which are central to the notion of minimality but mostly ignored by the conventional CSP definition.

2.1. Definitions

We recall the definition of the Constraint Satisfaction Problem, or CSP.

Definition 1 *(CSP).* A *CSP instance* I comprises:

1. A set $V = \{v_1, \dots, v_n\}$ of n variables.
2. A set $A = \{A_{v_1}, \dots, A_{v_n}\}$ of n domains. For all $i \in [1, n]$, $A_{v_i} = \{a_1, \dots, a_{d_i}\}$ contains the d_i possible values for the variable v_i.
3. An integer k and a set $C = \{C_1, \dots, C_m\}$ of m constraints. To each constraint C_i is associated a different *scope* $W_i = \{w_1, \dots, w_{k_i}\} \subseteq V$, with $2 \le k_i \le k$, and a set U_i of k_i-tuples from $A_{w_1} \times A_{w_2} \times \dots \times A_{w_{k_i}}$. We say that these tuples are *allowed*, that the tuples from $A_{w_1} \times A_{w_2} \times \dots \times A_{w_{k_i}}$ that are not in U_i are *forbidden* and that k_i is the *arity* of the constraint C_i.
   For each set $V' \subseteq V$ containing k' variables with $2 \le k' \le k$, there is exactly one constraint in C whose scope is exactly V' (so $m = \sum_{k'=2}^{k} \binom{n}{k'}$). We say that k is the arity of the instance.

Note that since the scopes of the constraints cover all possible sets of variables of size between 2 and the arity of the instance, defining the constraints for a given k-ary CSP instance I is equivalent to specifying whether each tuple of k' values, with $2 \le k' \le k$, is allowed or forbidden.

Throughout the paper, and as long as the context is clear, we will associate a tuple of assignments with the corresponding set of values. For example, we will associate the 3-tuple composed of the value a_1 assigned to the variable v_1, the value a_2 assigned to the variable v_2 and the value a_3 assigned to the variable v_3 with the set $B = \{a_1, a_2, a_3\}$, as long as it is clear that the value $a_i \in B$ is the same as the value $a_i \in A_{v_i}$, for $i = 1, 2, 3$.

A *compatible* tuple B, or *partial solution*, is a set of assignments to variables of I such that no subset of B is a forbidden tuple. Similarly, an *incompatible* tuple B is a set of assignments to variables of I such that there is a subset of B which is a forbidden tuple. We also say that values in a compatible (respectively incompatible) tuple are compatible (respectively incompatible) with each other. In particular, any value in a forbidden tuple B is incompatible with the other values in B.

A *solution*, or *full solution*, to I is a partial solution on V. We now formally define the Minimal Constraint Satisfaction Problem, or Minimal CSP.