From Backdoor Key to Backdoor Completability: Improving a Known Measure of Hardness for the Satisfiable CSP

Guillaume Escamocher, Mohamed Siala, and Barry O'Sullivan

Insight Centre for Data Analytics
Department of Computer Science, University College Cork, Ireland
{guillaume.escamocher, mohamed.siala, barry.osullivan}@insight-centre.org

Abstract. Many studies have been conducted on the complexity of Constraint Satisfaction Problem (CSP) classes. However, there exists little theoretical work on the hardness of *individual* CSP instances. In this context, the backdoor key fraction (BKF) [17] was introduced as a quantifier of problem hardness for individual satisfiable instances with regard to backtracking search. In our paper, after highlighting the weaknesses of the BKF, we propose a better characterization of the hardness of an individual satisfiable CSP instance based on the ratio between the size of the solution space and that of the search space. We formally show that our measure is negatively correlated with instance hardness. We also show through experiments that this measure evaluates more accurately the hardness of individual instances than the BKF.

1 Introduction

Finding a solution to a CSP instance is well known to be NP-hard, even when considering satisfiable instances [3]. The complexity of CSP instances has been extensively cataloged in the framework of complexity theory. However, attempts to formally define instance hardness, to find out what makes some CSP instances difficult to solve have been scarcer. A number of studies have been proposed based, in particular, on the notion of constrainedness [9, 10], to predict the behavior of large sets of instances. Constrainedness compares the expected number of solutions of constraint instances to their average size. It is straightforward to compute, and is well-suited for large classes of problems, but it does not establish a distinction between individual instances that have the same average tightness but different solution spaces. On the other hand, only considering the solution space is not enough to accurately predict instance complexity [18].

The *Backdoor Key Fraction* was proposed in [17] to characterize the hardness of a given satisfiable CSP instance with respect to backtracking search based on the notion of *backdoor*. A backdoor is a set of variables that, when properly assigned, allows us to decide the remainder of the problem in polynomial time using a given sub-solver [19]. The backdoor key fraction is based on the backdoor key set. A variable is in the backdoor key set if its value is logically determined

by the settings of other backdoor variables. The key fraction is the ratio of the backdoor key set size to the corresponding backdoor size. Unfortunately, as we explain in Section 2, there are many classes of instances for which the backdoor key fraction is ill-fitted. The main motivation behind our paper is to revisit this measure by proposing a better characterization of instance hardness.

In this paper, we propose an improvement over the backdoor key fraction. Intuitively, a solver finds an instance difficult if it contains many paths that do not lead to a solution, where a path is a possible sequence of choices made by the solver. Therefore, we define our *completability* measure as the number of paths that are completable, meaning that they lead to a solution, divided by the number of paths actually explored by the solver. This can be viewed as the ratio between the solution space and the search space. If the ratio is close to 1, the solver mostly branches on completable paths and can easily solve the instance. If, however, the ratio is low, the solver explores a lot of dead-ends and finds the instance hard. Completability can be seen as an improvement over the backdoor key set. Indeed, as we explain in Section 2, both our metric and the backdoor key fraction are composed of a ratio between a numerator that takes into account the global interactions between the backdoor and the rest of the instance, and a denominator that only relies on the internal structure of the backdoor.

The notion of completability has previously been used by [7], albeit in a completely different way. The author proposed to add constraints to CSP instances to transform them into equivalent minimal instances, where an instance is minimal if any partial solution of size bounded by some predefined constant can be extended to a solution [16]. On the other hand, we are computing a theoretical measure and are not modifying any part of the observed instances. Nonetheless, it is interesting to note how his intention was to make CSP instances easier by, in essence, increasing the completability ratio of small subsets of variables. We show that when some particular small subsets of variables, namely backdoors, have a high completability ratio, then backtracking solvers have an easier time finding a solution. His paper and ours are, therefore, consistent with each other in their approach of completability. Another work closely related to the idea of completable partial solutions can be found in [4]. The authors also consider the ratio between the solution space and the search space within small sets of variables (although not backdoors), but their measure is restricted to minimal CSP instances, while our metric can measure any satisfiable CSP instance.

In the next section, we give the different notions used throughout the paper and we highlight the limitations of the backdoor key fraction. In Section 3 we adopt a theoretical approach to justify our measure. Finally, we present our experimental study in Section 4.

2 CSP, Backdoor Key Fraction, and Backdoor Completability

A CSP instance is a triplet $\langle \mathcal{X}, \mathcal{D}, \mathcal{C} \rangle$ where $\mathcal{X}$ is a set of variables $\{v_1, v_2, \ldots, v_n\}$, $\mathcal{D}$ is a set of domains $\{D_{v_1}, D_{v_2}, \ldots, D_{v_n}\}$, and $\mathcal{C}$ is a set of constraints. A domain

D_v is a set of integers (values) associated with the variable v. A constraint C of arity $k \geq 1$ is a pair $(\mathcal{X}(C), \mathcal{R}(C))$, where $\mathcal{X}(C)$ is a sequence of k variables, and $\mathcal{R}(C) \subseteq \mathbb{Z}^k$. The set of variables in $\mathcal{X}(C)$ is called the scope of C. The constraint C is universal if every k-tuple of integers is in $\mathcal{R}(C)$. An assignment is a pair $\langle v, a \rangle$ where v is a variable and $a \in D_v$. A value a is said to be assigned to a variable v if $D_v = \{a\}$. An *instantiation* is a set of assignments where each variable appears at most once. The scope of an instantiation S is the set of variables $\{v \mid \exists \langle v, a \rangle \in S\}$. Let S be an instantiation and C be a constraint such that $\mathcal{X}(C) = [v_{i_1}, v_{i_2}, \ldots, v_{i_k}]$. We say that S violates C if $\forall l \in [1, k]$, there exists a_l such that $\langle v_{i_l}, a_l \rangle \in S$ and $\langle a_1, a_2, \ldots a_k \rangle \notin \mathcal{R}(C)$. The instantiation S is said to satisfy C if S does not violate C. An instantiation that does not violate any constraint is called a *partial solution*. A *solution* to a CSP instance $\langle \mathcal{X}, \mathcal{D}, \mathcal{C} \rangle$ is a partial solution with a scope equal to $\mathcal{X}$. Not every partial solution can be extended to a solution. A CSP instance that admits a solution is called *satisfiable*. The arity of a CSP instance is the greatest arity of its constraints. When an instance is binary (i.e., of arity 2), two assignments $\langle v, a \rangle$ and $\langle v', a' \rangle$ are incompatible if there exists a constraint C such that $\mathcal{X}(C) = [v, v']$ such that $\langle a, a' \rangle \notin \mathcal{R}(C)$. Two assignments are compatible if they are not incompatible.

Definition 1. *Let I be a CSP instance with n variables and let p be an integer such that $1 \leq p < n$. We say that I is* $(p, 1)$-consistent *if for any partial solution S of size p and for any variable v not in the scope of S, there is a value $a \in D_v$ such that $S \cup \{\langle v, a \rangle\}$ is a partial solution. We also say that I is* strongly $(p, 1)$-consistent *if it is $(q, 1)$-consistent for all q such that $1 \leq q \leq p$.*

A backdoor [19] is defined with regard to a particular sub-solver. We define a sub-solver and the other notions the same way that [17] did.

Definition 2. *An algorithm A that takes a CSP instance as input is a* sub-solver *if:*

1. *For any CSP instance I, either A rejects I or A correctly recognizes I as satisfiable or unsatisfiable. If I is recognized as satisfiable, then A also returns a solution to I.*
2. *A runs in time polynomial in the size of I.*

Now that we have explained the concept of sub-solvers, we can properly define the notion of a backdoor to a CSP.

Definition 3. *Let A be a sub-solver. Let I be a CSP instance. Let V be a subset of the variables of I. We say that V is a* backdoor *for A of I if there exists a partial solution S_p of scope V such that the instance I' obtained from I after assigning the value a to the variable v for each assignment $\langle v, a \rangle \in S_p$ is recognized as satisfiable by A.*

Informally, a backdoor is a (small) set of variables that, when properly assigned, makes the rest of the instance easy. When the sub-solver A is clear from the context, we shall use "backdoor" instead of "backdoor for A". Technically, the set of all variables in an instance is always a trivial backdoor. Therefore, we mainly focus on backdoors of minimal size.

Definition 4. *Let A be a sub-solver and let I be a satisfiable CSP instance. Let B be a set of variables of I. We say that a backdoor B for A of I is a* minimal backdoor *if $\forall v \in B$, $B \backslash \{v\}$ is not a backdoor for A of I.*

Before presenting our backdoor completability measure, we describe the existing metric that is closest to our own. This is the backdoor key fraction, introduced in [17]. Backdoor keys are sets of dependent variables, where a dependent variable is defined as follows:

Definition 5. *Let I be a satisfiable CSP instance, let B be a subset of variables of I, and let $v \in B$ be a variable. Let S be a solution to I, and let $S_p \subset S$ be a partial solution of scope $B \backslash \{v\}$. We say that v is a* dependent variable with respect to S_p *if there is exactly one value a in D_v such that $S_p \cup \{\langle v, a \rangle\}$ can be extended to a solution to I.*

Definition 6. *Let I be a satisfiable CSP instance, let B be a backdoor for I, and let $v \in B$ be a variable. Let S be a solution to I, and let $S_B \subset S$ be a partial solution of scope B and let S_v be equal to S_B restricted to $B \backslash \{v\}$. We say that v is in the* backdoor key *set of B with respect to S_B if v is a dependent variable with respect to S_v.*

The backdoor key fraction can now be defined.

Definition 7. *Let B be a backdoor for a satisfiable CSP instance I, and let S_B be a partial solution to B. The* backdoor key fraction *of B with respect to S_B is the ratio between the number of variables in the backdoor key set of B with respect to S_B and the total number of variables in B. If B is empty, we say that the backdoor key fraction of B is 0.*

The last sentence is our own addition to account for the cases when the backdoor is empty. Note that this follows the intuition of their paper. Indeed, empty backdoors are associated with very easy instances, and their intention was for the backdoor key fraction to be positively correlated with the hardness.

There are many cases where the backdoor key fraction is not useful. The authors of [17] mentioned the case where given any backdoor and its corresponding solution, one can always flip the truth assignment of any variable in the backdoor and still extend the backdoor to a solution. In such instances, the backdoor key fraction is equal to 0 for any backdoor. Another issue arises when a given CSP instance I only has one solution, the backdoor key fraction of any non-empty backdoor of I is by Definition 7 always 1. More generally, any backdoor variable which is also part of the instance backbone (the set of variables that are assigned the same values in all solutions [15]) is a dependent variable, and therefore is in the backdoor key.

In general, hard instances with regard to backtracking algorithms are the ones that offer many potentially wrong choices and few potentially right ones to solvers. Therefore, if we want to quantify hardness, it makes sense to build a measure that keeps track of both the size of the search space (the choices that the solver can make) and the size of the solution space (the right choices).

What we define as the search space is the set of partial, local solutions while the solution space is simply the set of global solutions.

Definition 8. *Let I be a CSP instance and let B be a non-empty set of variables of I. We say that the* completability ratio *of B is the ratio $\frac{\#completable}{\#partial}$ where:*

- *$\#partial$ is the number of partial solutions of scope B.*
- *$\#completable$ is the number of partial solutions of scope B that can be extended to a solution to I.*

We also say that the completability ratio of an empty set of variables is 1, and that the completability ratio of a set of variables with no partial solution is 0.

From now on, we only apply the notion of completability ratio on minimal backdoors within satisfiable CSP instances. However, this concept is general and could be applied to any set of variables in any CSP instance (although the ratio is trivially always 0 in unsatisfiable instances). In particular, the completability ratio is not dependent on a particular sub-solver, and is unique for each set of variables within a particular CSP instance. We define now our measure for a whole instance.

Definition 9. *Let A be a sub-solver and let I be a satisfiable CSP instance. The* backdoor completability for *A of I is the average of the completability ratios of all minimal backdoors for A of I.*

Observe that backdoor completability can be used to study satisfiable CSP instances of any arity. This is also the case for backdoor key fraction. Recall that finding a solution to a satisfiable CSP instance is an NP-hard problem [3], so such a restriction does not diminish the usefulness of either measure. It should be noted also that we are not limited to binary instances. Indeed, in Section 4, we present an experimental study on both binary and non-binary instances.

3 Theoretical Justification

In order to be a valid measure of hardness, backdoor completability needs to correctly recognize both easy and hard classes. In the former case, this is done by returning a high value for tractable classes. In the latter case, this is done by returning a low value for a subset of decent size in each non-tractable class. However, both tractability and backdoors are defined with regard to a specific algorithm. So ideally backdoor completability should tag a class as tractable if and only if the sub-solver used to define a backdoor solves the class.

In this section, we present an example to further explain what result we are aiming for, then we state our main Theorem. We refer to *(primal) constraint graphs*, *tree decompositions* and *treewidth*. We now recall the definitions of these four concepts.

Definition 10. *Let I be a CSP instance. The* primal constraint graph *of I is the graph G such that:*

Data: A satisfiable instance I with n variables $v_1, v_2, \ldots, v_n$.
Result: Either REJECT or SATISFIABLE.
Build primal constraint graph G of I;
if *G is a tree* **then**
  Sort the n vertices of G to get an ordering $v'_1, v'_2, \ldots, v'_n$ such that for each i there is at most one j such that $j < i$ and v'_i is connected to v'_j;
else
  return REJECT;
end
Establish $(1, 1)$-consistency on I;
for $i \leftarrow 1$ **to** n **do**
  Assign to the variable v'_i the lowest value a_i left in $D_{v'_i}$ such that $\{\langle v'_1, a_1\rangle, \langle v'_2, a_2\rangle, \ldots, \langle v'_i, a_i\rangle\}$ is a partial solution;
  if *no such value exists* **then**
    return REJECT;
  end
end
return SATISFIABLE;

Algorithm 1: A simple sub-solver based on $(1, 1)$-consistency.

- *The vertices of G are the variables of I.*
- *There is a an edge between two vertices v_i and v_j of G if and only if there is a non-universal constraint C of I such that $\mathcal{X}(C)$ contains both v_i and v_j.*

In the case of binary instances, the primal constraint graph is called the constraint graph. Part of the algorithms that we present is to build the *tree decomposition* of some (primal) constraint graph.

Definition 11. *Let G be a graph. Let T be a tree such that each vertex of T is a set of vertices of G. We say that T is a* tree decomposition *of G if:*

1. *Each vertex of G belongs to at least one vertex of T.*
2. *If two vertices v_1 and v_2 are connected in G, then there is a vertex of T containing both v_1 and v_2.*
3. *If two vertices t_1 and t_2 in T both contain some vertex v of G, then all the vertices of T in the path between t_1 and t_2 also contain v.*

Definition 12. *Let G be a graph. The* width *of a given tree decomposition of G is the number of vertices of G in the largest vertex of this tree decomposition, minus one. The* treewidth *of G is the lowest width of all possible tree decompositions of G.*

To illustrate the validity of our measure on one very specific example, consider the class $\mathcal{C}_{\text{tree}}$ composed of the satisfiable binary CSP instances whose constraint graph is a tree, the class $\mathcal{C}_{\text{all}}$ composed of all satisfiable CSP instances of any arity and with any (primal) constraint graph, and the sub-solver described by Algorithm 1.

Algorithm 1 builds a solution to instances with a tree as a primal constraint graph by starting from a random root after establishing $(1,1)$-consistency and then following along the branches of the tree. It correctly solves the class $\mathcal{C}_{\text{tree}}$ [5], but not the class $\mathcal{C}_{\text{all}}$. Therefore, in order to be a valid measure of hardness for these two classes and Algorithm 1, backdoor completability needs to be high for all instances of $\mathcal{C}_{\text{tree}}$ and very low for at least some instances of $\mathcal{C}_{\text{all}}$. As we show in a generalized version of this example in Theorem 1, this is what indeed happens.

Note that we do not require backdoor completability to return a high value for *all* instances of a given hard CSP class. A CSP class does not need to have all of its instances hard to be considered hard. The general CSP is hard for all solvers, even though most CSP instances are easy in practice.

Our Theorem covers the set of CSP classes $\{\mathcal{C}_1, \mathcal{C}_2, \dots\}$, such that each $\mathcal{C}_p$ is the set of satisfiable CSP instances whose primal constraint graph treewidth is upper bounded by p. These classes are hierarchically ordered by inclusion: for each p, $\mathcal{C}_p \subset \mathcal{C}_{p+1}$. Therefore, any given sub-solver finds all classes up to some p easy, meaning that it returns a solution to all instances from these classes, and all larger classes hard, meaning that it rejects at least some instances from each one of these subsequent classes. Note that the union of all the $\mathcal{C}_p$ is equal to the NP-hard satisfiable CSP, so no sub-solver can find all the classes easy, unless P=NP.

We are going to prove that for any sub-solver A belonging to a specific set of algorithms based on local consistency, and for any aforementioned class $\mathcal{C}_p$, backdoor completability returns a very low value with regard to A for some of the instances in $\mathcal{C}_p$ if and only if $\mathcal{C}_p$ is a hard class for A. We define "very low" as exponentially inverse to the number of variables.

Theorem 1. *Let p and p' be such that $p, p' > 0$. Let $A_{p,p'}$ be the sub-solver described by Algorithm 2 and let $\mathcal{C}_{p'}$ be the set of satisfiable CSP instances whose primal constraint graph treewidth is upper bounded by p'. Then exactly one of the two following statements is true:*

- *$p \geq p'$ and for every instance $I \in \mathcal{C}_{p'}$, the backdoor completability for $A_{p,p'}$ of I is equal to 1.*
- *$p < p'$ and for every integer N there is an instance I in $\mathcal{C}_{p'}$ with n variables such that $n > N$ and the backdoor completability for $A_{p,p'}$ of I is equal to $O(\frac{1}{2^{n/p'}})$.*

To prove the second point of the Theorem, we shall build instances with a low enough backdoor completability for $A_{p,p'}$. Using binary constraints is enough to do so, however we emphasize that instances in the class $\mathcal{C}_{p'}$ can be of any arity, so the scope of our result is not restricted to binary instances.

Definition 13. *Let $N > 1$ and $p' > 1$ be two integers. Then we call $I_{N,p'}$ the binary CSP instance defined in the following way:*

1. *Variables:* $1+p'N$ *variables* $v_0, v_{1,1}, v_{1,2}, \dots, v_{1,p'}, v_{2,1}, \dots, v_{2,p'}, v_{3,1}, \dots, v_{N,p'}$.

Data: A satisfiable instance I with n variables $v_1, v_2, \ldots, v_n$.
Result: Either REJECT or SATISFIABLE.
Build primal constraint graph G of I;
if *(treewidth of G)*$\leq p'$ **then**
  Build a p'-wide tree decomposition T of G;
else
  return REJECT;
end
Sort the n' vertices of T to get an ordering $v'_1, v'_2, \ldots, v'_{n'}$ such that for each i there is at most one j such that $j < i$ and v'_i is connected to v'_j;
Establish strong $(p, 1)$-consistency on all sets of variables that are entirely contained within a single vertex v_i;
for $i \leftarrow 1$ **to** n' **do**
  for *each variable v_j in the vertex v'_i of T* **do**
    Assign to v_j the lowest value a_j left in D_{v_j} such that $\{\langle v_1, a_1\rangle, \langle v_2, a_2\rangle, \ldots, \langle v_j, a_j\rangle\}$ is a partial solution;
    if *no such value exists* **then**
      return REJECT;
    end
  end
end
return SATISFIABLE;

Algorithm 2: A sub-solver based on $(p, 1)$-consistency.

2. *Domains: For each $1 \leq i \leq N$ and each $1 \leq j \leq p'$, $D_{v_{i,j}} = \{1, 2, \ldots, p'\}$. Furthermore, $D_{v_0} = \{1, 2, 3, \ldots, N+1\}$.*
3. *Constraints: For each $1 \leq i \leq N$, for all $1 \leq i_1, i_2 \leq p'$ such that $i_1 \neq i_2$, for each $1 \leq a < p'$, the assignments $\langle v_{i,i_1}, a\rangle$ and $\langle v_{i,i_2}, a\rangle$ are incompatible.*
4. *Constraints: For each $1 \leq a \leq N$, for each $1 \leq i \leq p'$, the assignments $\langle v_0, a\rangle$ and $\langle v_{a,i}, p'\rangle$ are incompatible.*
5. *Constraints: For each $1 \leq i \leq N$, for each $1 \leq j \leq p'$, for each $1 \leq a < p'$, the assignments $\langle v_0, N+1\rangle$ and $\langle v_{i,j}, a\rangle$ are incompatible.*
6. *Constraints: All pairs of assignments that have not been mentioned yet are compatible.*

In addition, for each $1 \leq i \leq N$, we call V_i the set of variables $\{v_{i,1}, v_{i,2}, \ldots, v_{i,p'}\}$ and we call t_i the set of variables $\{v_0\} \cup V_i$.

Note that any non-universal constraint can only be between two variables of a same set t_i for some i. We give two figures to illustrate the constraints within the variables of a set t_i in an instance $I_{N,p'}$, in this case t_2 in the instance $I_{7,4}$. Figure 1 illustrates points 3 and 6 from Definition 13, while Figure 2 illustrates points 4, 5 and 6. In both figures, a circle represents the domains of a variable, a dot represents a value in a domain and a dashed line connects two incompatible assignments. In Figure 1, a continuous line connects two compatible assignments. In order to not clutter the figure, only the most representative pairs of (in)compatible assignments are connected. For the same reason, in Figure 2

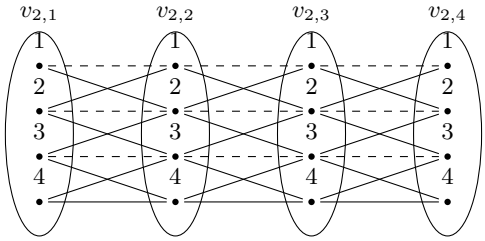


Fig. 1: The variables of V_2 in $I_{7,4}$, their domains and the related constraints.

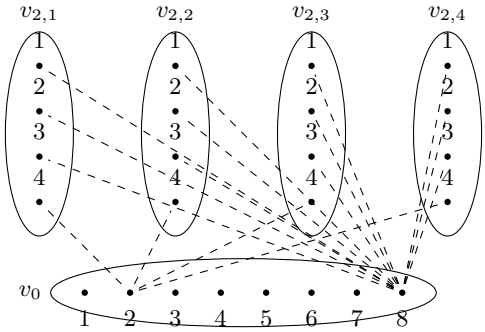


Fig. 2: The constraints between the variable v_0 and the set of variables V_2 in $I_{7,4}$.

pairs of compatible assignments are not shown, and only pairs of incompatible assignments involving a value from the domain of v_0 are connected.

To simplify the proof of the theorem, we first give some preliminary results concerning the instances $I_{N,p'}$.

Lemma 1. *Let $N > 1$ and $p' > 1$ be two integers. Then the instance $I_{N,p'}$ has exactly one solution, consisting of the assignment $\langle v_0, N+1\rangle$ and of the assignments $\langle v_{i,j}, p'\rangle$ for all $1 \leq i \leq N$ and $1 \leq j \leq p'$.*

Proof. It is easy to check that the set of assignments described in the statement of the Lemma is indeed a solution. We prove that it is the only one. Let S be a solution to I. let s_0 be the value such that $\langle v_0, s\rangle \in S$ and for all i and j such that $1 \leq i \leq N$ and $1 \leq j \leq p'$, let $s_{i,j}$ be the value such that $\langle v_{i,j}, s_{i,j}\rangle \in S$. For each $1 \leq i \leq N$, we know from Definition 13.3 that at least one of the $s_{i,1}, s_{i,2}, \ldots, s_{i,p'}$ is equal to p'. So from Definition 13.4 we know that for each $1 \leq i \leq N$, $s_0 \neq i$. So $s_0 = N+1$. So from Definition 13.5 we know that $s_{i,j} = p'$ for all i and j such that $1 \leq i \leq N$ and $1 \leq j \leq p'$. So we have shown that the one and only solution to I is the one described in the Lemma. □

Lemma 2. *Let $N > 1$ and $p' > 1$ be two integers. Then the instance $I_{N,p'}$ belongs to $\mathcal{C}_{p'}$.*

Proof. $\mathcal{C}_{p'}$ is the set of satisfiable CSP instances whose primal constraint graph treewidth is upper bounded by p'. From Lemma 1, we know that $I_{N,p'}$ is satisfiable, so it only remains to prove that the treewidth of its constraint graph is upper bounded by p'.

Let T be a graph with N vertices $t_1, t_2, \ldots, t_N$ and $N-1$ edges, such that:

- Each vertex t_i is as defined in Definition 13: the set of the $p'+1$ variables $\{v_0, v_{i,1}, v_{i,2}, \ldots, v_{i,p'}\}$.
- For each $1 \leq i < N$, the pair (t_i, t_{i+1}) is an edge of T.

From the second point, T is a tree. Each variable $v_{i,j}$ of $I_{N,p'}$ is in the vertex t_i of T and v_0 is in every vertex of T, so the first condition in the definition of a tree decomposition (Definition 11) is fulfilled. Since each non-universal constraint of $I_{N,p'}$ either involves v_0 or is between two variables of a same set V_i, each edge in the constraint graph of $I_{N,p'}$ is contained in a vertex of T and the second condition of Definition 11 is fulfilled. Furthermore, the only variable of $I_{N,p'}$ that appears in several vertices of T is v_0, which appears in all vertices of T, so the third condition in Definition 11 is fulfilled. So T is a tree decomposition of the constraint graph of $I_{N,p'}$. Since each vertex of T contains $p'+1$ variables of I, the treewidth of the constraint graph of $I_{N,p'}$ is (at most) p'. □

Lemma 3. *Let N, p and p' be three integers such that $N > 1$ and $0 < p < p'$. Let B be a set of variables of $I_{N,p'}$ such that $v_0 \notin B$. Then B is a backdoor for $A_{p,p'}$ of $I_{N,p'}$ if and only if B contains a variable from each set $V_i = \{v_{i,1}, v_{i,2}, \ldots, v_{i,p'}\}$, with at most one exception.*

Proof. – B is a backdoor for $A_{p,p'}$ of $I_{N,p'} \Rightarrow B$ contains a variable from each set V_i, with at most one exception:
We first show that for each $1 \leq i \leq N$, t_i is strongly $(p,1)$-consistent. Let i be an integer such that $1 \leq i \leq N$ and let S be a partial solution of scope $W = \{w_1, w_2, \ldots, w_q\}$, with $q \leq p$ and $W \subset t_i$. Let v be a variable from $t_i \backslash W$. We need to show that there is a value $a \in D_v$ such that $S \cup \{\langle v, a\rangle\}$ is a partial solution. There are three cases to consider:
- v is v_0: from Definition 13, $\langle v_0, j\rangle$ is compatible with all assignments on variables from V_i if $i \neq j$. So $S \cup \{\langle v, 2\rangle\}$ is a partial solution if $i = 1$ and $S \cup \{\langle v, 1\rangle\}$ is a partial solution otherwise.
- One of the variables from W is v_0: let s_0 be such that $\langle v_0, s_0\rangle \in S$. There are three possibilities for the value of s_0. First possibility, $s_0 = N+1$. In this case, we know from Definition 13.5 that all the other assignments in S are of the form $\langle w_j, p'\rangle$, so $S \cup \{\langle v, p'\rangle\}$ is a partial solution. Second possibility, $s_0 = i$. In this case, we know from Definition 13.4 that none of the other assignments in S is of the form $\langle w_j, p'\rangle$, and $S \cup \{\langle v, a\rangle\}$ is a partial solution, with a a value such that $a \neq p'$ and $a \neq b$ for each assignment $\langle w_j, b\rangle \in S$ such that $w_j \neq v_0$. There is always such a value a, because W has at most p variables, so $W \backslash \{v_0\}$ has at most $p-1 \leq p'-2$ variables. Third and last possibility, either $s_0 < i$ or $i < s_0 < N+1$. In this case, $S \cup \{\langle v, p'\rangle\}$ is a partial solution.

- Neither v nor any of the variables from W is v_0: we know from Definition 13 that $S \cup \{\langle v, p' \rangle\}$ is a partial solution.

So for each variable $v \in t_i \backslash W$, there is a value $a \in D_v$ such that $S \cup \{\langle v, a \rangle\}$ is a partial solution. So t_i is strongly $(p, 1)$-consistent.

Let B' be a set of variables of $I_{N,p'}$ such that $v_0 \notin B'$. Suppose that there are some i and j with $i \neq j$ such that no variable from $V_i \cup V_j$ is in B'. We have just shown that t_i (which we recall is $V_i \cup \{v_0\}$) and t_j (which is $V_j \cup \{v_0\}$) are strongly $(p, 1)$-consistent. So establishing strong $(p, 1)$-consistency after assigning the value p' to the variables in B' leaves at least the three values i, j (because $\langle v_0, i \rangle$ and $\langle v_0, j \rangle$ are compatible with all assignments $\langle v, p' \rangle$ for all variables v not in t_i nor t_j) and $N+1$ (because $\langle v_0, N+1 \rangle$ is compatible with all assignments $\langle v, p' \rangle$ for all variables v of $I_{N,p'}$) in D_{v_0}. However, the only assignments in the unique solution to I are of the form $\langle v, a \rangle$, with a the highest value in D_v, and $A_{p,p'}$ picks the lowest available value in each domain. So even after assigning the correct values to B' and establishing strong $(p, 1)$-consistency, and whatever the order in which the sub-solver sorts the variables, $A_{p,p'}$ will pick a wrong value when making its first choice within $t_1 \cup t_2$, and will eventually reject I. So B' is not a backdoor for $A_{p,p'}$ of $I_{N,p'}$. So any backdoor B for $A_{p,p'}$ of $I_{N,p'}$ not containing v_0 contains at least one variable in every set V_i, with at most one exception.

– B contains a variable from each set V_i, with at most one exception $\Rightarrow$ B is a backdoor for $A_{p,p'}$ of $I_{N,p'}$:
Let B be a set of $N-1$ variables, each in a different set V_i, and none being v_0. Let i be the integer such that no variable from V_i is in B. Once we have assigned the value p' to all variables of B' and established strong $(p, 1)$-consistency, we know from Definition 13.4 that all the values $1, 2, 3, \ldots, i-1, i+1, \ldots, N-1, N$ will be removed from D_{v_0}. Let $a \in D_v$ and $b \in D_w$ be two values from the domains of two different variables v and w of $t_i \backslash \{v_0\}$, such that $a < p'$ and $b = p'$. Since the only two remaining values in D_{v_0} are i and $N+1$, we know from Definition 13.4 and 13.5 respectively that neither $\langle v_0, i \rangle$ nor $\langle v_0, N+1 \rangle$ is compatible with both $\langle v, a \rangle$ and $\langle w, b \rangle$. So establishing strong $(p, 1)$-consistency will make incompatible all such pairs of assignments $\langle v, a \rangle$ and $\langle w, b \rangle$ with v and w in $t_i \backslash \{v_0\}$. From Definition 13.3, we know that there is no partial solution to V_i that contains an assignment $\langle v, a \rangle$ with $a \neq p'$. So establishing strong $(p, 1)$-consistency will make incompatible all pairs of assignments $\langle v, a \rangle$ and $\langle w, b \rangle$ with v and w from $t_i \backslash \{v_0\}$ and either $a \neq p'$ or $b \neq p'$, and will eventually remove all values other than p' from the domains of $t_i \backslash \{v_0\}$. So from Definition 13.4, the value i will also be removed from the domain of v_0, leaving only the value $N+1$ in this domain. Lastly, from Definition 13.5, all values other than p' will be removed from all other domains, leaving only one value in each domain after establishing strong $(p, 1)$-consistency. So B is a backdoor for $A_{p,p'}$ of $I_{N,p'}$. □

Now that we have the results we need, we can prove the main theorem.

Proof of Theorem 1: Through the proof, we assume that "backdoor" and "backdoor completability" are implicitly "backdoor for $A_{p,p'}$" and "backdoor completability for $A_{p,p'}$" respectively.

- Suppose that $p \geq p'$. Since p' is a constant, building a tree decomposition of width p' is polynomial [2]. In general, it is well-known that using strong $(p,1)$-consistency alongside a p-wide tree decomposition solves the CSP restricted to instances whose primal constraint graph has a treewidth bounded by p [6]. Therefore, the empty set is a minimal backdoor of I, and from Definition 9 the backdoor completability of I is 1.
- Suppose that $p < p'$. It is enough to show that for each integer $N > 1$, there is an instance $I \in \mathcal{C}_{p'}$ with $n = p'N + 1$ variables such that the backdoor completability of I is equal to $O(\frac{1}{2^{n/p'}})$. Let $N > 0$ be an integer and let I be the instance $I_{N,p'}$ defined in Definition 13.
  - I is in $\mathcal{C}_{p'}$: from Lemma 2.
  - I has n variables: from Definition 13.1.
  - The backdoor completability of I is equal to $O(\frac{1}{2^{n/p'}})$: let B be a minimal backdoor of I. There are two possibilities for B:
    - B contains v_0. If we assign the correct value $N+1$ to the backdoor variable v_0, then we know from Definition 13.5 that after $A_{p,p'}$ establishes strong $(p,1)$-consistency, only one value will remain in every other domain. So any set of variables containing v_0 is a backdoor of I, but among them only $\{v_0\}$ is a minimal backdoor of I. In this case, there are $N+1$ partial solutions of scope B (one for each value in D_{v_0}), and exactly one of them is a subset of a solution ($\{\langle v_0, N+1\rangle\}$). Therefore the completability ratio of B is $\frac{1}{N+1}$.
    - B does not contain v_0. From Lemma 3, we know that B contains at least one variable from each set $V_i = \{v_{i,1}, v_{i,2}, \ldots, v_{i,p'}\}$, with at most one exception. From the same Lemma we also know that containing one variable from each set V_i except one is a sufficient condition for a backdoor. So, since B is minimal, B contains exactly one variable from each set V_i except one. Note that all constraints between any two variables of B are universal and that every domain of B contains exactly p' values, so there are p'^{N-1} possible partial solutions of scope B. From Lemma 1, we know that there is only one solution to I, so only one partial solution of scope B can be extended to a solution and therefore the completability ratio of B is $\frac{1}{p'^{N-1}}$.

    We have shown that either $B = \{v_0\}$ and has a completability ratio of $\frac{1}{N+1}$ or B is composed of $N-1$ variables from $N-1$ different sets V_i and has a completability ratio of $\frac{1}{p'^{N-1}}$. There are Np'^{N-1} possible backdoors of the latter kind, so the backdoor completability of I is lower than $\frac{2}{p'^{N-1}}$. Recall than $n = p'N + 1$, so $N = \frac{n-1}{p'}$ and therefore $\frac{2}{p'^{N-1}} = O(\frac{2}{p'^{n/p'}})$. Since $p' > p \geq 1$, $O(\frac{2}{p'^{n/p'}}) = O(\frac{1}{2^{n/p'}})$ and therefore the backdoor completability of I is equal to $O(\frac{1}{2^{n/p'}})$.

  We have exhibited a satisfiable instance $I \in \mathcal{C}_{p'}$ with n variables and a backdoor completability equal to $O(\frac{1}{2^{n/p'}})$. □

We have formally proved that backdoor completability correctly measures hardness for some precisely defined classes of instances and sub-solvers, namely satisfiable CSP instances with bounded treewidth and sub-solvers based on local consistency. Tractable classes relying on bounded treewidth are common [5, 8], while consistency is a ubiquitous tool in modern solvers [13], so our result shows that at least for some widely known algorithms and instance classes, backdoor completability is a valid measure.

4 Comparison with the Backdoor Key Fraction

We present an experimental comparison between the backdoor key fraction and the backdoor completability. Our experiments cover two different sets of problems: Quasigroup Completion With Holes (QWH) [1, 12], and random satisfiable CSP. We used the Mistral [11] solver for both experiments with default settings.

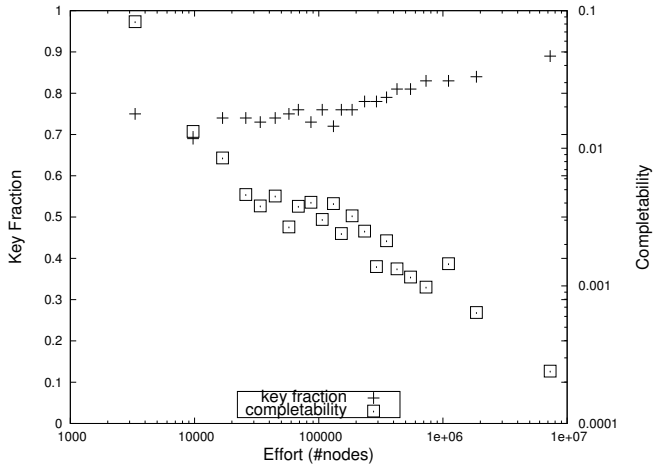
For QWH, we generated 1100 instances of order 22, with a number of holes equally spread over the range 192 to 222. This range was chosen to capture the instances at hand around the observed peak of difficulty at 204 holes. Note that [17] also used QWH instances to test the backdoor key fraction.

The inequality constraints are posted through the AllDifferent constraint with the bound consistency algorithm of [14]. The sub-solver that we chose has exactly the same configuration, however, with 500 failure limit. Note that the sub-solver is guaranteed to run in polynomial time since constraint propagation is polynomial as well. For each instance, we generated 100 minimal backdoors, using the methodology described in [17]. For each backdoor, we randomly sampled 20 partial solutions.

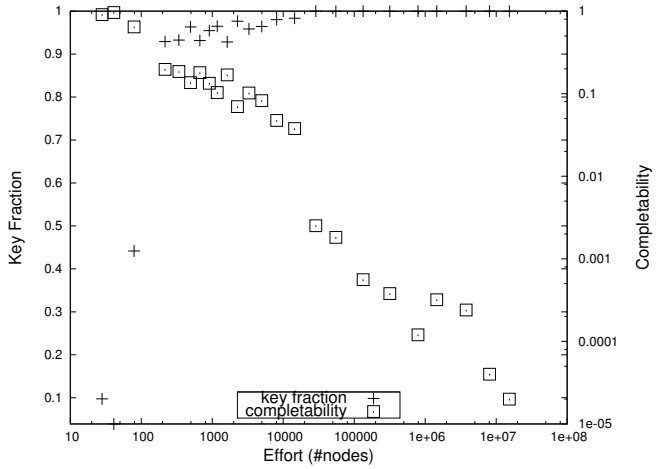
Figure 3a represents the results of our experiments on QWH instances. The x-axis represents the number of decisions required to find a solution, while the left y-axis represents the backdoor key fraction and the right y-axis represents backdoor completability. Each point represents an average of 50 instances, with the 50 easiest instances in one group, the 50 next easiest ones in a second group, and so on until the 50 hardest instances.

The correlation for the backdoor key fraction is good, with a Pearson coefficient of .876. This is consistent with the results from [17]. The correlation for backdoor completability is even better, with a coefficient of -.943; we recall that the further away from 0 the coefficient is, the more correlated the values are. This demonstrates that backdoor completability can prove a better quantifier of instance difficulty than the backdoor key fraction, even in problems where the latter has a good track record.

The second problem we studied is composed of random satisfiable CSP instances with 60 variables and 1770 constraints. We generated 1200 such instances, with an average tightness in each constraint equally spread over the range 5% to 16%, with an observed peak of difficulty at 8%. The correlations for backdoor key fraction and backdoor completability are presented in Figure 3b. As in Figure 3a, the instances are sorted by difficulty in groups of 50.



(a) QWH



(b) Random CSP

Fig. 3: Experimental results for QWH instances (top) and random CSP (bottom)

Table 1: Correlations between instance hardness and different measures.

	QWH			Random CSP		
	Pearson CC	RMSE	MAE	Pearson CC	RMSE	MAE
Backdoor key fraction	.876	.053	.032	.590	.186	.158
Backdoor completability	-.943	.037	.029	-.975	.051	.044

We can see from these results that the backdoor key fraction is not adequate for that type of instance. It grows with the hardness, as is expected, but does

not clearly distinguish between instances above a certain threshold of difficulty. Indeed, we observed that most of the hardest instances in this set have only a few variables that are not part of the backbone. In many cases all variables are in the backbone and the instance has exactly one solution; as explained in the remarks following Definition 7, the backdoor key fraction will always output 1 for instances with exactly one solution. This case (along with the other one mentioned in the last paragraph of Section 2) shows a limitation of the key fraction to capture hardness in some instances. On the other hand, backdoor completability does not study the variables separately, but examines the properties of the partial solutions to a whole backdoor. It is therefore more refined than the backdoor key fraction, in particular when looking at individual variables is not enough, for example when comparing instances that have a backbone of similar (large) size but different degrees of difficulty.

Table 1 contains the summary of our experiments. In addition to the Pearson correlation coefficients (Pearson CC), the table also includes the normalized values for the root mean square error (RMSE) and mean absolute error (MAE), two error measures for linear regression that [17] also reported. The results confirm that backdoor completability is negatively correlated with instance hardness, and that it measures the difficulty of instances in both sets more accurately than the backdoor key fraction does.

5 Conclusion

We have introduced a new measure, backdoor completability, that characterizes the hardness of an individual satisfiable CSP instance with regard to a given solver. Backdoor completability can be viewed as an index of hardness; the lower the value, the harder the instance. This measure is a crucial step towards the understanding of what makes a particular instance difficult.

We provided a theoretical justification of our measure. We proved that for some widespread classes of instances, namely CSP instances with bounded treewidth, backdoor completability captures exactly the limits of tractability. We also presented an empirical comparison between our metric and the existing backdoor key fraction, and showed that for some kinds of CSP instances, backdoor completability is reliable even though the backdoor key fraction is not.

The main motivation of our work is to revisit and to improve the backdoor key fraction measure. In the future, it would be interesting to study the practical usefulness of completability as it provides insights for designing search strategies. Moreover, we believe that completability could eventually be useful in generating hard instances since hard instances are the ones with low completability.

Acknowledgements

This research has been funded by Science Foundation Ireland (SFI) under Grant Number SFI/12/RC/2289.

References

1. Achlioptas, D., Gomes, C.P., Kautz, H.A., Selman, B.: Generating satisfiable problem instances. In: Proceedings of AAAI, IAAI, July 30 - August 3, 2000, Austin, Texas, USA. pp. 256–261 (2000)
2. Bodlaender, H.L.: A linear-time algorithm for finding tree-decompositions of small treewidth. SIAM J. Comput. 25(6), 1305–1317 (1996)
3. Dechter, R.: Constraint processing. Elsevier Morgan Kaufmann (2003)
4. Escamocher, G., O'Sullivan, B.: On the minimal constraint satisfaction problem: Complexity and generation. In: Proceedings of COCOA, Houston, TX, USA, December 18-20, 2015. pp. 731–745 (2015)
5. Freuder, E.C.: A sufficient condition for backtrack-free search. J. ACM 29(1), 24–32 (1982)
6. Freuder, E.C.: Complexity of k-tree structured constraint satisfaction problems. In: Proceedings of AAAI. Boston, Massachusetts, July 29 - August 3, 1990, 2 Volumes. pp. 4–9 (1990)
7. Freuder, E.C.: Completable representations of constraint satisfaction problems. In: Proceedings of KR. Cambridge, MA, USA, April 22-25, 1991. pp. 186–195 (1991)
8. Ganian, R., Ramanujan, M.S., Szeider, S.: Combining treewidth and backdoors for CSP. In: 34th Symposium on Theoretical Aspects of Computer Science, STACS 2017, March 8-11, 2017, Hannover, Germany. pp. 36:1–36:17 (2017)
9. Gent, I.P., MacIntyre, E., Prosser, P., Walsh, T.: The constrainedness of search. In: Proceedings of AAAI, IAAI, Portland, Oregon, August 4-8, 1996, Volume 1. pp. 246–252 (1996)
10. Gomes, C.P., Fernández, C., Selman, B., Bessière, C.: Statistical regimes across constrainedness regions. Constraints 10(4), 317–337 (2005)
11. Hebrard, E.: Mistral, a constraint satisfaction library. Proceedings of the Third International CSP Solver Competition 3, 3 (2008)
12. Kautz, H.A., Ruan, Y., Achlioptas, D., Gomes, C.P., Selman, B., Stickel, M.E.: Balance and filtering in structured satisfiable problems. In: Proceedings of IJCAI August 4-10, 2001, Seattle, Washington, USA. pp. 351–358 (2001)
13. Larrosa, J., Schiex, T.: Solving weighted CSP by maintaining arc consistency. Artif. Intell. 159(1-2), 1–26 (2004)
14. López-Ortiz, A., Quimper, C., Tromp, J., van Beek, P.: A fast and simple algorithm for bounds consistency of the alldifferent constraint. In: Proceedings of IJCAI, Acapulco, Mexico, August 9-15, 2003. pp. 245–250 (2003)
15. Monasson, R., Zecchina, R., Kirkpatrick, S., Selman, B., Troyansky, L.: Determining computational complexity from characteristic 'phase transitions'. Nature 400(8), 133–137 (1999)
16. Montanari, U.: Networks of constraints: Fundamental properties and applications to picture processing. Inf. Sci. 7, 95–132 (1974)
17. Ruan, Y., Kautz, H.A., Horvitz, E.: The backdoor key: A path to understanding problem hardness. In: Proceedings of AAAI, IAAI, July 25-29, 2004, San Jose, California, USA. pp. 124–130 (2004)
18. Valiant, L.G., Vazirani, V.V.: NP is as easy as detecting unique solutions. Theor. Comput. Sci. 47(3), 85–93 (1986), https://doi.org/10.1016/0304-3975(86)90135-0
19. Williams, R., Gomes, C.P., Selman, B.: Backdoors to typical case complexity. In: Proceedings of IJCAI, Acapulco, Mexico, August 9-15, 2003. pp. 1173–1178 (2003)