https://doi.org/10.1007/s11704-025-50231-4

RESEARCH ARTICLE

SAT requires exhaustive search

Ke XU[1]✉, Guangyan ZHOU[2]

1. State Key Laboratory of Complex and Critical Software Environment, Beihang University, Beijing 100191, China
2. Department of Mathematics and Statistics, Beijing Technology and Business University, Beijing 100048, China

Received March 7, 2025; accepted May 18, 2025

E-mail: kexu@buaa.edu.cn

Abstract
In this paper, we identify the distinction between non-brute-force computation and brute-force computation as the most fundamental problem in computer science. Subsequently, we prove, by the diagonalization method, that constructed self-referential CSPs cannot be solved by non-brute-force computation, which is stronger than P $\neq$ NP. This constructive method for proving impossibility results is very different (and missing) from existing approaches in computational complexity theory, but aligns with Gödel's technique for proving logical impossibility. Just as Gödel showed that proving formal unprovability is feasible in mathematics, our results show that proving computational hardness is not hard in mathematics. Specifically, proving lower bounds for many problems, such as 3-SAT, can be challenging because these problems have various effective strategies available to avoid exhaustive search. However, for self-referential examples that are extremely hard, exhaustive search becomes unavoidable, making its necessity easier to prove. Consequently, it renders the separation between non-brute-force computation and brute-force computation much simpler than that between P and NP. Finally, our results are akin to Gödel's incompleteness theorem, as they reveal the limits of reasoning and highlight the intrinsic distinction between syntax and semantics.

Keywords
CSP; SAT; exhaustive search; brute-force computation; lower bounds; self-reference; Model RB

1 Introduction

Model RB is a random constraint satisfaction problem (CSP) model that was proposed by Xu and Li [1] in 2000, which could also be encoded to well-known NP-complete problems like SAT and CLIQUE. The purpose of this model was to address the issue of trivial unsatisfiability that was prevalent in previous random CSP models. One of the key features of Model RB is that its domain size d grows with the number of variables n. Additionally, Model RB has been proved to exhibit exact phase transitions from satisfiability to unsatisfiability, making it a useful tool for analyzing and evaluating the performance of algorithms. Over the last two decades, Model RB has been extensively researched from multiple perspectives, as evidenced by various studies (e.g., [2–21]). Moreover, this model has gained significant popularity and widespread use in renowned international algorithm competitions. A random instance of Model RB with a planted solution named frb100-40, where $n = 100$ and $d = 40$, has remained elusive since it was made available online in 2005 as a 20-year challenge for algorithms[1)]. Despite numerous attempts, no one has been able to solve it thus far. In summary, the results suggest that Model RB possesses nice mathematical properties that can be easily derived. In contrast to its mathematical tractability, the random instances of this model, particularly those generated in the phase transition region, present a significant challenge for various algorithms, proving to be extremely difficult to solve.

As shown in the proof of Gödel's incompleteness theorem [22], the constructive approach plays an indispensable role in revealing the fundamental limitations of finite formal systems. An algorithm is essentially a mechanized finite formal system. In this paper, we will study the limitations of algorithms based on the constructive approach. Specifically, we will explore whether non-exhaustive (non-brute-force) algorithms can always replace exhaustive (brute-force) ones, or if some computable problems inherently lack such alternatives. The Strong Exponential Time Hypothesis [23] conjectures that there is no non-brute-force algorithm for SAT. This problem is very similar to the foundational problem resolved by Gödel's incompleteness theorem, originally proposed by David Hilbert nearly a century ago: whether a finite formal system can always replace a branch of mathematics (e.g., arithmetic) that contains infinitely many true mathematical statements. These two

1) See tinyurl.com/2p53xbd7 website.

problems raise essentially the same deep philosophical question: can the part always replace the whole within the limits of reasoning? This inquiry concerns the limits of human knowledge —a subject extensively explored by many great philosophers (e.g., Laozi, Zeno, Socrates, Descartes, Kant, and Wittgenstein). Therefore, *the most fundamental problem in computer science is non-brute-force computation vs. brute-force computation*, rather than P vs. NP. From a mathematical perspective, the distinction between non-brute-force computation, which takes $O(T^c)$ time (where $c<1$ is a constant), and brute-force computation, which takes T time, seems more natural and intuitive compared to that between P and NP. From a practical standpoint, the framework of non-brute-force computation vs. brute-force computation is also broader and more universally applicable than that of P vs. NP. For instance, the P vs. NP paradigm does not apply to problems where brute-force algorithms run in polynomial time or those that lie outside NP. However, even in such cases, we can still explore the possibility of developing non-brute-force algorithms. Finally, from a historical perspective, non-brute-force computation vs. brute-force computation extends Gödel's framework, which unveils the limits of mathematics. Similarly, P vs. NP builds on Turing's framework, shedding light on the limits of machines. The limits of mathematics are more fundamental than those of machines, as anything mathematics cannot achieve is also beyond the reach of machines.

The advantages of Model RB enable us to choose specific threshold points at which instances with a symmetry requirement are on the edge of being satisfiable and unsatisfiable. In fact, we will show that there exist instances at exactly the same point which are either satisfiable with exactly one solution or unsatisfiable but only fail on one constraint. The satisfiability of such instances can be flipped under a special symmetry mapping. As a result, they form a fixed point set under this mapping, allowing us to create the most indistinguishable examples (self-referential examples) which are a source of computational hardness. Based on the symmetry mapping and driven by the famous method of diagonalization and self-reference, we show that unless exhaustive search is executed, the satisfiability of a certain constraint (thus the whole instance) is possible to be changed, while the subproblems of the whole instance remain unchanged. Therefore, whether the whole instance is satisfiable or unsatisfiable cannot be distinguished without exhaustive search. In summary, if we can construct the most indistinguishable examples with exactly the same method and the same parameter values (a task that is both rare and challenging), then it is not hard to understand and prove why they are extremely hard to solve.

2 Model RB

A random instance I of Model RB consists of the following:

- A set of variables $\mathcal{X}=\{x_1,\dots,x_n\}$: Each variable x_i takes values from its domain D_i, and the domain size is $|D_i|=d$, where $d=n^{\alpha}$ for $i=1,\dots,n$, and $\alpha>0$ is a constant.
- A set of constraints $\mathcal{C}=\{C_1,\dots,C_m\}$ ($m=rn\ln d$, where $r>0$ is a constant): for each $i=1,\dots,m$, constraint $C_i=(X_i,R_i)$. $X_i=(x_{i_1},x_{i_2},\dots,x_{i_k})$ ($k\geqslant 2$ is a constant) is a sequence of k distinct variables chosen uniformly at random without repetition from $\mathcal{X}$. R_i is the permitted set of tuples of values which are selected uniformly without repetition from the subsets of $D_{i_1}\times D_{i_2}\times\cdots\times D_{i_k}$, and $|R_i|=(1-p)d^k$ where $0<p<1$ is a constant.

In this paper, we have a symmetry requirement of the permitted set of each constraint, and the m permitted sets will be generated in the following way. Initially, we generate a symmetry set R which contains $(1-p)d^k$ tuples of values, then generate each permitted set R_i of the constraint C_i ($i=1,2,\dots,m$) by running random permutations of domains of $k-1$ variables in X_i based on R. For example, if $k=2$ and the domains are $D_1=D_2=\{1,2,3,4\}$, then $R=\{(1,1),(1,2),(2,1),(2,2),(3,3),(3,4),(4,3),(4,4)\}$ is a symmetry set. If we run a random permutation of D_1, e.g., $f(1)=3,f(2)=1,f(3)=4,f(4)=2$, then we get a permitted set $\{(3,1),(3,2),(1,1),(1,2),(4,3),(4,4),(2,3),(2,4)\}$. Through this method all $R_i(i=1,\dots,m)$ are isomorphic and every domain value of the variables shares the same properties.

A constraint $C_i=(X_i,R_i)$ is said to be satisfied by an assignment $\sigma\in D_1\times D_2\times\cdots\times D_n$ if the values assigned to X_i are in the set R_i. An assignment σ is called a solution if it satisfies all the constraints. I is called satisfiable if there exists a solution, and called unsatisfiable if there is no solution. It has been proved that Model RB not only avoids the trivial asymptotic behavior but also has exact phase transitions from satisfiability to unsatisfiability. Indeed, denote $\mathbf{Pr}[I \text{ is SAT}]$ the probability that a random instance I of Model RB is satisfiable, then

Theorem 2.1 ([1]). Let $r_{cr}=\frac{1}{-\ln(1-p)}$. If $\alpha>1/k$, $0<p<1$ are two constants and k,p satisfy the inequality $k\geqslant 1/(1-p)$, then

$$\lim_{n\to\infty}\mathbf{Pr}[I\ is\ SAT\ ]=1 \text{ if } r<r_{cr},$$
$$\lim_{n\to\infty}\mathbf{Pr}[I\ is\ SAT\ ]=0 \text{ if } r>r_{cr}.$$

In the following we will present some properties of Model RB which are important to prove our main theorems in the next section. From here on we tacitly take $r=r_{cr}+\frac{\delta}{n\ln d}$, where $\delta=\frac{\ln 2}{-\ln(1-p)}$, and take

$$\alpha>\max\left\{1,\inf\{\alpha:\omega<0\},-2\left(\frac{100}{99}\right)^2\frac{\ln(1-p)}{k},\frac{100\ln(1-p)}{k\ln(1-\frac{p}{3})}\right\},\tag{2.1}$$

where $\omega=1+\alpha(1-r_{cr}pk)$. In fact note that $\omega=1+\alpha\left(1+\frac{pk}{\ln(1-p)}\right)$, and a simple calculation yields that $pk+\ln(1-p)>0$ for all $p\in(0,1)$ under the condition that $k\geqslant\frac{1}{1-p}$. Thus it is possible to take $\alpha>0$ large enough such that $\omega<0$.

First, we bound the probability that a random RB instance is satisfiable.

Lemma 2.2 Let I be a random CSP instance of Model RB with n variables and $rn\ln d$ constraints. Then

$$\frac{1}{3} \leqslant \mathbf{Pr}(I \text{ is } SAT) \leqslant \frac{1}{2}.$$

Proof Let X be the number of solutions of I, then

$$\mathbf{Pr}(X > 0) \leqslant \mathbf{E}[X] = d^n(1-p)^{rn\ln d} = \frac{1}{2}. \tag{2.2}$$

As shown in [1],

$$\mathbf{E}[X^2] = \sum_{S=0}^{n} d^n \binom{n}{S}(d-1)^{n-S}\left((1-p)\frac{\binom{S}{k}}{\binom{n}{k}} + (1-p)^2\left(1-\frac{\binom{S}{k}}{\binom{n}{k}}\right)\right)^{rn\ln d}$$
$$= \mathbf{E}[X]^2\left(1+O\left(\frac{1}{n}\right)\right)\sum_{S=0}^{n} F(S), \tag{2.3}$$

where $F(S) = \binom{n}{S}\left(1-\frac{1}{d}\right)^{n-S}\left(\frac{1}{d}\right)^S\left[1+\frac{p}{1-p}s^k\right]^{rn\ln d}$, and $S = ns$ is the number of variables for which an assignment pair take the same values. Note that $\alpha > 1$, using an argument similar to that in [1,7], we obtain that only the terms near $S = 0$ and $S = n$ are not negligible, and

$$\frac{\mathbf{E}[X^2]}{\mathbf{E}[X]^2} \leqslant 1 + \frac{1}{\mathbf{E}[X]} + o(1) = 3 + o(1). \tag{2.4}$$

Indeed, asymptotic calculations show that

$$F(0) = 1 - o(1), F(i) = (1+o(1))n^{i(1-\alpha)}, \ldots,$$
$$F(n-i) = (1+o(1))\exp\{i(\ln n + \ln d - pkr\ln d)\}/\mathbf{E}[X],$$
$$F(n) = 1/\mathbf{E}[X],$$

where $i = 1, 2, \ldots$ is an integer. Note that $\exp\{i(\ln n + \ln d - pkr\ln d)\} = n^{i\omega + o(1)}$ and $\alpha > 1, \omega < 0$ are constants, thus the upper bound of (2.4) comes from $F(0)$ and $F(n)$.

Using the Cauchy inequality, we get $\mathbf{Pr}(X > 0) \geqslant \mathbf{E}[X]^2/\mathbf{E}[X^2] \geqslant \frac{1}{3}$. □

As an immediate consequence of Lemma 2.2 we obtain a lower bound of the probability that I has exactly one solution.

Corollary 2.3 Let I be a random CSP instance of Model RB with n variables and $rn\ln d$ constraints. Then the probability that I has exactly one solution is at least $1/6$.

Proof Let ρ_1 be the probability that I has exactly one solution, and $\rho_{\geqslant 2}$ be the probability that I has at least two solutions. Then from Lemma 2.2, we have

$$\mathbf{E}[X] = \frac{1}{2} \geqslant \rho_1 + 2\rho_{\geqslant 2},$$
$$\mathbf{Pr}(X > 0) = \rho_1 + \rho_{\geqslant 2} \geqslant \frac{1}{3}.$$

Therefore $\rho_1 \geqslant 1/6$. □

Next we show that if a random instance is unsatisfiable, then w.h.p.[2)] it fails at only one constraint. We introduce the following definitions.

Definition 2.1 Let I be a CSP instance. A constraint C is called a self-unsatisfiable constraint if there exists an assignment under which C is the only unsatisfied constraint in I. If variable x is contained in C, then x is called a self-unsatisfiable variable. If I is unsatisfiable and every variable is a self-unsatisfiable variable, then I is called a self-unsatisfiable formula.

Lemma 2.4 Let I be a random CSP instance of Model RB with n variables and $rn\ln d$ constraints. If I is unsatisfiable, then w.h.p. I is a self-unsatisfiable formula.

Proof First we show that for any constraint C of I, with positive probability there exists an assignment which satisfies all constraints except C. In fact let N be the number of such assignments, then

$$\mathbf{E}[N] = d^n(1-p)^{rn\ln d - 1}p. \tag{2.5}$$

Using a similar argument as in [1], we have

$$\mathbf{E}[N^2] = \sum_{S=0}^{n} d^n\binom{n}{S}(d-1)^{n-S}\left((1-p)\frac{\binom{S}{k}}{\binom{n}{k}} + (1-p)^2\left(1-\frac{\binom{S}{k}}{\binom{n}{k}}\right)\right)^{rn\ln d - 1} \cdot$$
$$\left(p\frac{\binom{S}{k}}{\binom{n}{k}} + p^2\left(1-\frac{\binom{S}{k}}{\binom{n}{k}}\right)\right).$$

Hence, (2.3),(2.4) and (2.5) ensure that

$$\frac{\mathbf{E}[N^2]}{\mathbf{E}[N]^2} = \frac{\mathbf{E}[X^2]}{\mathbf{E}[X]^2} \cdot \frac{1+\frac{1-p}{p}s^k}{1+\frac{p}{1-p}s^k} \leqslant \frac{1}{p}\frac{\mathbf{E}[X^2]}{\mathbf{E}[X]^2} \leqslant \frac{3}{p}.$$

Then $\mathbf{Pr}(N > 0) \geqslant \frac{\mathbf{E}[N]^2}{\mathbf{E}[N^2]} \geqslant \frac{p}{3}$. Therefore the probability that C is a self-unsatisfiable constraint is at least $\frac{p}{3}$.

Next, since the number of constraints is $rn\ln d$, the average degree of each variable $x \in \mathcal{X}$ is $rk\ln d$. By the Chernoff Bound,

$$\mathbf{Pr}\left[\mathrm{Deg}(x) \leqslant \frac{1}{100}rk\ln d\right] \leqslant e^{-(99/100)^2 rk\ln d/2} = n^{-(99/100)^2 rk\alpha/2},$$

where $\mathrm{Deg}(x)$ denotes the degree of variable x. From the requirement (2.1) we know that $1 - (99/100)^2 rk\alpha/2 < 0$, thus

$$n\mathbf{Pr}\left[\mathrm{Deg}(x) \leqslant \frac{1}{100}rk\ln d\right] \leqslant o(1). \tag{2.6}$$

Therefore, almost surely all variables have degree at least $\frac{1}{100}rk\ln d$.

Furthermore, note that each variable appears in at least $\frac{1}{100}rk\ln d$ constraints and the probability that each constraint appears to be a self-unsatisfiable constraint is at least $\frac{p}{3}$, thus the probability that x is not a self-unsatisfiable variable is at most $\left(1-\frac{p}{3}\right)^{\frac{1}{100}rk\ln d}$.

Note that (2.1) entails that $1 + \frac{1}{100}rk\alpha\ln(1-p/3) < 0$, therefore the probability that there exists a variable which is not self-unsatisfiable is at most

$$n\left(1-\frac{p}{3}\right)^{\frac{1}{100}rk\ln d} = n^{1+\frac{1}{100}rk\alpha\ln(1-p/3)} = o(1). \tag{2.7}$$

Thus w.h.p. all variables are self-unsatisfiable variables. □

[2)] We say a property holds w.h.p. (with high probability) if this property holds with probability tending to 1 as the number of variables approaches infinity.

Remark 2.1 We claim that Lemmas 2.2 and 2.4 hold for any domain size greater than polynomial. Take exponential domain size $d=\beta^n$ (where $\beta>1$ is a constant) for example, similarly, the dominant contributions of $\mathbf{E}[X^2]/\mathbf{E}[X]^2$ come from $F(0)=1-o(1)$ and $F(n)=1/\mathbf{E}[X]$, and asymptotic calculations show that $F(i)=\Theta\left((\frac{n}{d})^i\right)$ and

$$F(n-i)=(1+o(1))\frac{1}{\mathbf{E}[X]}\exp\left\{i\ln n+i\Big(1+\frac{pk}{\ln(1-p)}\Big)\ln d\right\}$$

are negligible for small integer i, since $1+\frac{pk}{\ln(1-p)}<0$. Moreover, probability analysis holds more easily in the proof of Lemma 2.4 if $d=\beta^n$ ((2.6) and (2.7)).

Next we define a *symmetry mapping* of a constraint which changes its permitted set slightly.

Definition 2.2 Consider a random instance I of Model RB with $k=2$. Assume that $C=(X,R)$ is a constraint of I and $X=(x_1,x_2)$, then a symmetry mapping of C is to change R by choosing $u_1,u_2\in D_1$ $(u_1\neq u_2)$, $v_1,v_2\in D_2$ $(v_1\neq v_2)$, where $(u_1,v_1),(u_2,v_2)\in R$ and $(u_1,v_2),(u_2,v_1)\notin R$, and then exchanging u_1 with u_2 (see Fig. 1).

With the above properties of Model RB, we obtain the following interesting results.

Theorem 2.5 There exists an infinite set of satisfiable and unsatisfiable instances of Model RB such that this set is a fixed point under the symmetry mapping of changing satisfiability.

Proof Let $\mathcal{I}$ be the set of RB instances with n variables and $rn\ln d$ constraints, where each instance either has a unique solution or has no solution.

Assume that $I\in\mathcal{I}$ has exactly one solution σ, which happens with probability at least $1/6$ from Corollary 2.3. For an arbitrary constraint C', assume without loss that $C'=(X',R')$, $X'=(x,x_1),\sigma(x)=u,\sigma(x_1)=v$, and D,D_1 are the domains of x and x_1, respectively. By the symmetry requirement, there exist $u'\in D,v'\in D_1$ such that $(u',v')\in R',(u,v')\notin R',(u',v)\notin R'$, then we will exchange u with u'. It is easy to see that this symmetry mapping will convert (u,v) into an unpermitted tuple and convert $(u,v'),(u',v)$ into permitted tuples, thus σ is no longer a solution. However, it is possible that at most $2(1-p)d$ pairs $(u,*),(u',*)\in R$ can be expanded to new solutions (this is because by the symmetry requirement, the degree of each domain value of each variable is $(1-p)d$). Specifically, the probability that (u,v') can be expanded to a new solution is at most $d^{n-2}(1-p)^{rn\ln d-1}=\frac{1}{2(1-p)d^2}$, thus a simple calculation yields that the probability that I is still satisfiable after the symmetry mapping is at most $O(\frac{1}{d})=o(1)$.

Assume that $I\in\mathcal{I}$ is an unsatisfiable instance, then w.h.p. all variables in I are self-unsatisfiable variables from Lemma 2.4. This implies that there exist a constraint $C''\in C_x$ and an assignment τ such that C'' is the only unsatisfied one under τ. Assume without loss that $C''=(X'',R''),X''=(x,x_2),\tau(x)=u,\tau(x_2)=w$, and D,D_2 are the domains of x and x_2, respectively. It is apparent that $(u,w)\notin R''$. By our symmetry requirement, there exist $u'\in D,w'\in D_2$, where $(u,w'),(u',w)\in R''$ and $(u',w')\notin R''$, such that a symmetry mapping of exchanging u with u' will convert (u,w) into a permitted tuple, thus C'' becomes satisfiable under τ. Moreover, using a similar argument as above, we can see that the probability that the new pairs $(u,*),(u',*)\in R$ could expand to solutions is at most $O(\frac{1}{d})=o(1)$. Thus w.h.p. I has only one solution after this symmetry mapping.

From the above two cases we can see that for any $I\in\mathcal{I}$, the symmetry mapping changes its satisfiability, however, I still belongs to $\mathcal{I}$ after the mapping, thus $\mathcal{I}$ can be considered as a fixed point under the symmetry mapping. □

In this section, it has been shown that we can construct satisfiable and unsatisfiable instances using exactly the same method and the same parameter values. Moreover, these satisfiable and unsatisfiable instances can be transformed into each other by performing the same

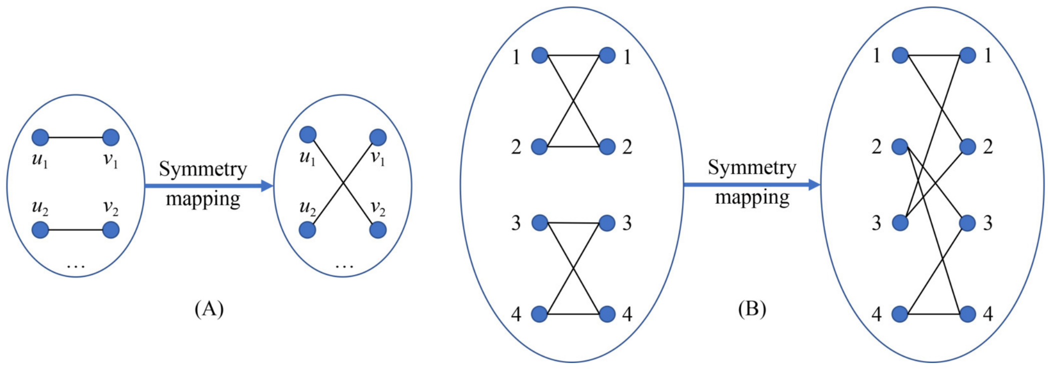

Fig. 1 (A) shows a symmetry mapping of the constraint C by exchanging u_1 with u_2. (B) shows an example of a symmetry mapping by exchanging the domain value 2 with 3 in D_1, where we set $D_1=D_2=\{1,2,3,4\}$ and the original permitted set $R=\{(1,1),(1,2),(2,1),(2,2),(3,3),(3,4),(4,3),(4,4)\}$. R becomes $\{(1,1),(1,2),(2,3),(2,4),(3,1),(3,2),(4,3),(4,4)\}$ after such a symmetry mapping. Thus for an assignment σ, if $\sigma(x_1)=2,\sigma(x_2)=2$, then C changes from satisfiable to unsatisfiable under σ, since $(2,2)$ does not belong to the permitted set R any more; for an assignment τ, if $\tau(x_1)=2,\tau(x_2)=3$, then C changes from unsatisfiable to satisfiable under τ, since $(2,3)$ belongs to the permitted set R after this symmetry mapping

mapping. This property is very similar to that of the self-referential proposition introduced by Gödel [22] in order to prove that such a proposition can be neither proved nor disproved (i.e., whether this proposition is true or false cannot be distinguished in finite formal systems). Gödel's results reveal the fundamental difference between the syntax defined by rules and the semantics defined by models (or assignments). An algorithm is essentially a finite sequence of rules, which can also be viewed as a finite formal system. In contrast, the exhaustive search method determines, using the semantic definition of a property, whether this property can be satisfied through trying every possible model (or assignment) one by one. Inspired by Gödel's idea, we refer to these satisfiable and unsatisfiable instances as *the most indistinguishable examples* or *self-referential examples*. Their self-referential property makes them extremely hard to differentiate syntactically. Simply put, in this context, syntax cannot replace semantics.

3 Main results

Proving complexity lower bounds (algorithmic impossibility results) for a given problem is essentially reasoning and making conclusions about an infinite set of algorithms. In mathematics, any such conclusion should be based on assumptions about the nature of the infinite set. These assumptions must be consistent with the reality and usually appear as axioms. Similar to many combinatorial problems, the general CSP has no global structure that can be exploited to design algorithms. The only exact algorithm currently available for solving CSPs is a divide-and-conquer method that systematically explores the solution space while employing various pruning strategies to enhance efficiency.

In this paper, we view an algorithm as a finite formal system which is defined by a finite set of symbols and rules. It is easy to see that finite formal systems possess greater expressive power than algorithms (Turing machines). This is because finite formal systems are essentially finite sets of rules while algorithms are essentially finite sequences of rules. Here we use finite formal systems to solve CSPs (i.e., to determine if a CSP instance is satisfiable). We only need to assume that this task is finished by dividing the original problem into subproblems. Under this assumption, we have the following lemma.

Lemma 3.1 If a CSP problem with n variables and domain size d can be solved in $T(n) = O(d^{cn})$ time ($0 < c < 1$ is a constant), then at most $O(d^c)$ subproblems with $n-1$ variables are needed to solve the original problem.

Proof It is easy to see that d subproblems with $n-1$ variables are sufficient to solve the original problem. By condition, a CSP problem with $n-1$ variables can be solved in $T(n-1)$ time. Note that

$$T(n) = O(d^{cn}) = O(d^c d^{c(n-1)}) = O(d^c)T(n-1),$$

thus at most $O(d^c)$ subproblems with $n-1$ variables are needed to solve the original problem. □

The above lemma is a key to proving the main results of this paper. For a better understanding, we take the classical sorting problem as an example. Assume that our goal is to prove that sorting n elements cannot be done in $T(n) = O(n)$ time. By condition, we have $T(n) = O(n-1) + O(1) = T(n-1) + O(1)$. This means that we need a subproblem of $n-1$ elements with additional $O(1)$ operations to solve the original problem. We can show by contradiction that this is impossible and so finish the proof.

Theorem 3.2 Model RB cannot be solved in $O(d^{cn})$ time for any constant $0 < c < 1$.

Proof For the sake of simplicity, we will prove that Theorem 3.2 holds for Model RB with $k = 2$ by contradiction. Let I be a RB instance with n variables and $rn\ln d$ constraints. Suppose there exists some constant $0 < c < 1$ such that I can be solved in $O(d^{cn})$ time, then Lemma 3.1 implies that I can also be solved by assigning at most $O(d^c)$ values to an arbitrary variable, say x, and then solving the resulting subproblems (with $n-1$ variables) which require $O(d^{c(n-1)})$ time. We will show that there exist instances where the $O(d^c)$ subproblems produced by assigning $O(d^c)$ values to x are impossible to determine its satisfiability. For convenience, let D be the domain of x, $\widetilde{D}$ be the set of $O(d^c)$ values which have been assigned to x ($|\widetilde{D}| = O(d^c)$), and C_x be the set of constraints containing x.

Follow the strategy of the proof of Theorem 2.5, if I is an instance having exactly one solution σ, then the constraint C' will be arbitrarily selected from C_x. Note that $O(d^c)$ is $o(1)$ compared with the domain size d, thus the probability that u belongs to $\widetilde{D}$ is $o(1)$. Therefore the symmetry mapping in Theorem 2.5 will be performed by exchanging u with u', where u' is chosen from $D \backslash \widetilde{D}$, then I becomes unsatisfiable. Similarly, if I is an unsatisfiable instance, then the symmetry mapping of exchanging u with u', where u' is chosen from $D \backslash \widetilde{D}$, will convert (u, w) into a permitted tuple, thus I becomes satisfiable.

Note that in either of the above cases, the $O(d^c)$ subproblems with $n-1$ variables remain unchanged, while the satisfiability of the original problem with n variables has been changed after performing the symmetry mapping. We can conclude that the $O(d^c)$ subproblems are insufficient thus impossible to determine if I is satisfiable or unsatisfiable. This completes the proof of Theorem 3.2. □

As mentioned before, finite formal systems possess greater expressive power than algorithms (Turing machines). The above theorem indicates that no finite formal system for solving Model RB can significantly outperform or replace the exhaustive search method, which evaluates d^n possibilities one by one. That is to say, non-brute-force computation cannot replace brute-force computation for Model RB. More interestingly, the above proof follows the same method (i.e., diagonalization and self-reference) used by Kurt Gödel [22] and Alan Turing [24], respectively, in their epoch-making papers of proving logical and computational impossibility results. This classical method, pioneered by Cantor, stands out as not only simple, elegant, and powerful, but also as the only approach that has successfully provided lower bounds on complexity classes.

Moreover, the distinctive mathematical properties and the inherent extreme hardness of Model RB make it both feasible and effective in establishing algorithmic impossibility results. Importantly, applying this method to constructed instances, rather than directly to complexity classes, is crucial for achieving a successful proof. This study highlights the significance of creatively utilizing classical methods, particularly when addressing long-standing open problems. We believe that the underlying idea of this paper (i.e., Constructing the Most Indistinguishable Examples) will unlock the door to proving complexity lower bounds, which has always been a challenging task, even for many polynomial-time solvable problems.

It is worth mentioning that Xu and Li [3] established a direct link between proved phase transitions and the abundance with hard examples by proving that Model RB has both of these two properties. They also made a detailed comparison between Model RB and some other well-studied models with phase transitions, and stated that "such mathematical tractability should be another advantage of RB/RD, making it possible to obtain some interesting results which do not hold or cannot be easily obtained for random 3-SAT". This paper confirms that remarkable results can indeed be achieved through a simple and elegant approach, as expected. More than two thousand years ago, Laozi, a great Chinese thinker and philosopher, once said: "*The greatest truths are the simplest*". We believe that the truth of computational hardness should also adhere to this fundamental and universal principle advocated by Laozi. In this sense, both the problem and the results of this paper are simple yet intuitively accessible. Additionally, if a problem is so difficult that there are no nontrivial results, then the most likely scenario is that the problem under study is not inherently foundational. In such a case, what should be done is to redefine the problem in alignment with Laozi's principle. Foundational problems must be grounded in fundamental concepts. In this paper, we argue that non-brute-force computation vs. brute-force computation is more foundational than P vs. NP. This is because the concept of brute-force computation is inherently more fundamental than that of polynomial-time computation.

CSP can be encoded into SAT by use of the log-encoding [25] which introduces a new Boolean variable for each bit in the domain value of each CSP variable and thus uses a logarithmic number of Boolean variables to encode domains. It must be emphasized that each clause of these encoded SAT instances could be very long with $\Theta(\log_2 d)$ Boolean variables if the domain size d grows with the number of CSP variables n. This is in sharp contrast to 3-SAT that is very short in clause length and has received the most attention in the past half-century. For encoded SAT instances of Model RB using the log-encoding, we have totally $N = n\log_2 d$ Boolean variables, $O(Nd^k)$ clauses and $O(Nd^k \log_2 d)$ literals. Note that $d^{cn} = 2^{cn\log_2 d} = 2^{cN}$, so it is easy to derive the following corollary from Theorem 3.2.

Corollary 3.3 SAT with N Boolean variables cannot be solved in $O(2^{cN})$ time for any constant $0 < c < 1$.

The above corollary holds for SAT with no restriction on the clause length, and so is not directly applicable to k-SAT with constant $k \geqslant 3$ whose lower bounds can be obtained by reduction from encoded SAT instances of Model RB. Other CSP instances of domain size l can also be encoded from Model RB using $N = n\log_l d$ variables, thus cannot be solved in $O(l^{cN})$ time for any constant $0 < c < 1$.

It is well-known that SAT is NP-complete [26], and so it follows that P ≠ NP.

■ Competing interests

Ke XU is Deputy Editors-in-Chief of the journal and a co-author of this article. To minimize bias, he was excluded from all editorial decision-making related to the acceptance of this article for publication. The remaining authors declare no conflict of interest.

■ Open Access

■ References

[1] Xu K, Li W. Exact phase transitions in random constraint satisfaction problems. Journal of Artificial Intelligence Research, 2000, 12(1): 93−103

[2] Xu K, Boussemart F, Hemery F, Lecoutre C. A simple model to generate hard satisfiable instances. In: Proceedings of the 19th International Joint Conference on Artificial Intelligence. 2005, 337−342

[3] Xu K, Li W. Many hard examples in exact phase transitions. Theoretical Computer Science, 2006, 355(3): 291−302

[4] Xu K, Boussemart F, Hemery F, Lecoutre C. Random constraint satisfaction: easy generation of hard (satisfiable) instances. Artificial Intelligence, 2007, 171(8−9): 514−534

[5] Liu T, Lin X, Wang C, Su K, Xu K. Large hinge width on sparse random hypergraphs. In: Proceedings of the 22nd International Joint Conference on Artificial Intelligence. 2011, 611−616

[6] Cai S, Su K, Sattar A. Local search with edge weighting and configuration checking heuristics for minimum vertex cover. Artificial Intelligence, 2011, 175(9−10): 1672−1696

[7] Zhao C, Zheng Z. Threshold behaviors of a random constraint satisfaction problem with exact phase transitions. Information Processing Letters, 2011, 111(20): 985−988

[8] Saitta L, Giordana A, Cornuejols A. Phase Transitions in Machine

Learning. Cambridge: Cambridge University Press, 2011
[9] Zhao C, Zhang P, Zheng Z, Xu K. Analytical and belief-propagation studies of random constraint satisfaction problems with growing domains. Physical Review E, 2012, 85(1): 016106
[10] Fan Y, Shen J, Xu K. A general model and thresholds for random constraint satisfaction problems. Artificial Intelligence, 2012, 193: 1−17
[11] Lecoutre C. Constraint Networks: Targeting Simplicity for Techniques and Algorithms. John Wiley & Sons, 2013
[12] Huang P, Yin M. An upper (lower) bound for Max (Min) CSP. Science China Information Sciences, 2014, 57(7): 1−9
[13] Xu W, Zhang P, Liu T, Gong F. The solution space structure of random constraint satisfaction problems with growing domains. Journal of Statistical Mechanics: Theory and Experiment, 2015, 2015: P12006
[14] Liu T, Wang C, Xu K. Large hypertree width for sparse random hypergraphs. Journal of Combinatorial Optimization, 2015, 29(3): 531−540
[15] Knuth D E. The Art of Computer Programming. Addison-Wesley Professional, 2015
[16] Xu W, Gong F. Performances of pure random walk algorithms on constraint satisfaction problems with growing domains. Journal of Combinatorial Optimization, 2016, 32(1): 51−66
[17] Fang Z, Li C M, Xu K. An exact algorithm based on MaxSAT reasoning for the maximum weight clique problem. Journal of Artificial Intelligence Research, 2016, 55(1): 799−833
[18] Wang X F, Xu D Y. Convergence of the belief propagation algorithm for RB model instances. Journal of Software, 2016, 27(11): 2712−2724
[19] Li C M, Fang Z, Jiang H, Xu K. Incremental upper bound for the maximum clique problem. INFORMS Journal on Computing, 2018, 30(1): 137−153
[20] Karalias N, Loukas A. Erdős goes neural: an unsupervised learning framework for combinatorial optimization on graphs. In: Proceedings of the 34th International Conference on Neural Information Processing Systems. 2020, 6659−6672
[21] Zhou G, Xu W. Super solutions of the model RB. Frontiers of Computer Science, 2022, 16(6): 166406
[22] Gödel K. Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme I. Monatshefte für Mathematik und Physik, 1931, 38(1): 173−198
[23] Calabro C, Impagliazzo R, Paturi R. The complexity of satisfiability of small depth circuits. In: Proceedings of the 4th International Workshop on Parameterized and Exact Computation. 2009, 75−85
[24] Turing A M. On computable numbers, with an application to the entscheidungsproblem. Proceedings of the London Mathematical Society, 1937, S2−42(1): 230−265
[25] Walsh T. SAT *v* CSP. In: Proceedings of the 6th International Conference on Principles and Practice of Constraint Programming. 2000, 441−456
[26] Cook S A. The complexity of theorem-proving procedures. In: Proceedings of the 3rd Annual ACM Symposium on Theory of Computing. 1971, 151−158

Ke XU received the BE, ME, and PhD degrees from Beihang University, China in 1993, 1996, and 2000, respectively. He is currently a professor of computer science at Beihang University, China. His research interests include phase transitions in computational complexity, algorithm design, logic, graph neural networks, and crowd intelligence.

Guangyan ZHOU received the PhD degree from Beihang University, China in 2015. She is currently an associate professor of mathematics at Beijing Technology and Business University, China. Her research interests include phase transitions in computational complexity, combinatorial optimization, and probabilistic analysis.

■ Appendixes

The initial draft of the paper entitled “SAT Requires Exhaustive Search” was completed and released via arXiv in February 2023. During the subsequent two-year period, the authors have engaged in in-depth discussions with a number of esteemed experts in related fields. In the past year, they have actively shared their work with hundreds of experts around the world. As of March 2025, more than 30 experts have provided highly positive feedback, including remarks like “*truly revolutionary*” from Prof. Gregory Chaitin, the father of algorithmic information theory, who proved Chaitin’s incompleteness theorem similar to Gödel’s incompleteness theorem. In response to the comments and suggestions from experts, the authors have redefined the foundational problem of computer science. Additionally, they have added an extended abstract to more comprehensively elaborate on the motivation, conceptual framework, and techniques of this paper. After extensive and thorough deliberations, Frontiers of Computer Science has decided to publish this paper. After obtaining consent from the authors and the reviewers, comments from 7 experts are being published simultaneously, offering thought-provoking perspectives on the core ideas of this paper from various angles.

Appendix A

Construction is all you need ——extended abstract of “SAT requires exhaustive search”

Our paper introduces a new framework for studying computational hardness, which can be seen as an extension of Gödel’s framework for proving the incompleteness theorem. Gödel’s framework demonstrates that a constructed self-referential proposition is unprovable, which means that the reasoning based on syntax cannot determine the semantics of this proposition. Within this new framework, all aspects of computational hardness, including the P vs. NP problem, should be reconsidered. This famous problem obscures the surprising fact that we do not know any nontrivial result of complexity lower bounds (e.g., a super-linear lower bound), but we know barriers to obtaining such results. In many cases, barriers arise simply because we are not on the right track, and sometimes the problem itself is the biggest barrier. As the old saying goes, a question well asked is half answered. Therefore, we must look back at history and return to the origin to redefine the foundational problem of computer science — —namely, identifying the right question to ask when a given problem is already computable.

Computation is fundamentally a mechanical process of reasoning. In other words, it can be viewed as mechanized mathematics. There are two main lines of research on the limits of computation. Gödel’s results address the limits of reasoning within finite formal systems, while Turing’s results explore the limits of mechanization (i.e., whether mechanical processes can be realized in the physical universe). Computational time fundamentally relies on the reasoning of concrete examples rather than the computability of abstract problems. The P vs. NP problem is a natural extension of Turing’s concept of computability. However, even if a problem is uncomputable, each instance might still be solved quickly, though no general (mechanical) method exists to handle all instances. In other words, computability is unrelated to computational time. Therefore, Turing’s framework of computability is not suitable for studying computational hardness. This also explains why there are so many barriers in current complexity theory studies: we might have taken the wrong path from the very beginning. It is also worth noting that Turing himself clearly recognized that reasoning, rather than computability, is related to time. He once posed the profound question: “Does time enter into ‘This proposition is difficult to prove’?” (L. Wittgenstein, Lectures on the Foundations of Mathematics, Cambridge, 1939: From the Notes of R.G. Bosanquet, N. Malcolm, R. Rhees, and Y. Smythies. Harvester Press, 1976).

In fact, the Turing machine itself is an assumption about machines (i.e., mechanized finite formal systems) that can be realized in the physical universe. In essence, the Turing machine represents a fundamental physical assumption, and Turing’s findings on uncomputability signify the physical limits of mechanically solving all instances of a problem. The computational hardness of concrete examples arises from the limits of reasoning. It is unnecessary to discuss the mechanical solving of these examples based on the concept of Turing machines. This is because if the reasoning of concrete examples requires a long time, then solving these examples mechanically will also take a long time. We can also say that computational hardness results are essentially mathematical impossibility results of reasoning. Any mathematical impossibility result cannot be proved without restrictions on the form of mathematics. To derive such results, it is essential to make a mathematical (rather than physical) assumption about reasoning.

The essence of computational hardness lies not in the distinction between deterministic polynomial-time computation and non-deterministic polynomial-time computation (i.e., P vs. NP), but in the distinction between non-brute-force computation and brute-force computation. This distinction essentially questions whether syntax can replace semantics (i.e., whether the part can replace the whole through reasoning). For CSPs, this question reduces to whether a certain number of subproblems can replace the whole problem, based on a mathematical assumption about the reasoning of CSPs. We then construct self-referential CSPs and ultimately prove by contradiction that a certain number of subproblems cannot replace the whole problem for these CSPs. This is very similar to Gödel’s study on whether a finite set of true mathematical statements (i.e., axioms) can replace an infinite set of all true mathematical statements through reasoning. Therefore, our result can be considered an incompleteness theorem for CSPs, primarily due to the gap between syntax and semantics.

Gödel’s proof and our proof address two different cases (i.e., first-order logic and propositional logic, respectively). The essence of both proofs (i.e., the use of self-reference) is the same, but there are differences in constructing the self-referential object. Specifically, this involves constructing an object such that negating it results in an object equivalent to itself. In Gödel’s work, the self-referential object is a logical formula. In our work, the self-referential object is an infinite set of satisfiable and unsatisfiable examples. These self-referential examples necessitate brute-force computation because the part cannot replace the whole for these examples. In other words, reasoning based on syntax is ineffective, and only brute-force computation based on semantics can solve these examples. It should also be emphasized that our hardness results apply to finite formal systems under a mathematical assumption. Turing machines, on the other hand, are finite formal systems under a physical assumption. Thus, while both are subsets of a larger set, they cannot be directly compared. In other words, our work and current complexity theory are two paths that extend different frameworks from the same starting point in computer science. Specifically, non-brute-force computation vs. brute-force computation is an extension of Gödel’s framework, while P vs. NP is an extension of Turing’s framework.

Finally, we address the two main barriers in complexity theory. The relativization barrier indicates that the diagonalization method cannot separate P from NP. In our work, this method is applied to constructed satisfiable and unsatisfiable examples, avoiding the relativization barrier. Regarding the natural proof barrier, the diagonalization method can circumvent it. Thus, using constructed examples, we bypass both barriers. The fundamental reason is that Gödel’s framework (instead of Turing’s) is used to study computational hardness. Many believe that complexity theory is difficult because it deals with computational hardness. This is a misunderstanding. Our work shows that extreme hardness (brute-force computation) is easy to understand through the construction of self-referential examples. In short, construction is all you need.

Appendix B

Comments on paper entitled: SAT requires exhaustive search

Bridging theory and practice in computational complexity and algorithm design

Shaowei CAI

Institute of Software, Chinese Academy of Sciences, Beijing 100190, China

This paper provides a new way to understand the hardness of computational problems. I find the idea of understanding the hardness of a problem by constructing extremely hard instances is very interesting, and to some extent can bridge the gap between theory and practice in computational complexity and algorithm design.

It has been widely acknowledged that there is a large discrepancy between the theory and practice of algorithm design. For example, while SAT is proven to be NP-complete, many SAT instances from industries are solved efficiently by modern SAT solvers. This discrepancy arises because complexity theory focuses on the worst-case complexity of problems, while real-world SAT instances often possess structural properties that allow for efficient resolution. For example, industrial SAT problems derived from hardware verification tasks can be large (with millions of variables and clauses) but are highly structured, containing hierarchical and modular components. These structures are exploitable by SAT solvers through techniques like conflict-driven clause learning (CDCL), enabling them to find solutions or prove unsatisfiability without exhaustively exploring the entire solution space.

The fundamental reason for this gap is that computational complexity theory addresses the hardness of problems in general, not specific instances. Previous theoretical works usually provide negative complexity results without giving any concrete hard instance. Complexity classes like NP-complete capture the worst-case scenario. However, many real-world instances do not lie in the difficult regions, and sometimes it is even hard to find a difficult instance, as claimed in the hardness results in theory. In order to solve problems, algorithms are implemented into programs and eventually executed on computers. In this sense, constructive approach, instead of existence theorems, could shed additional useful insights on understanding the hardness of problems, and is helpful for bridging the gap between theory and practice.

The RB model, as described in the paper, generates instances with properties that make them difficult for algorithms to solve without exhaustive search. These instances are designed to have a phase transition where the probability of satisfiability changes abruptly. At this critical point, instances are neither trivially satisfiable nor easily proven to be unsatisfiable. They often have a balance between constraint density and variable freedom that challenges both systematic and stochastic search algorithms. Additionally, the symmetry requirement in RB model instances means that variables and their domains are structured in a way that prevents algorithms from exploiting structures to guide the search. The frb100-40 instance remains an unsolved challenge even after 20 years. Despite numerous attempts, no one has been able to solve it.

Empirically, many CSP instances generated from the RB model have remained unsolved for years despite advances in algorithmic techniques. These instances are constructed to avoid the structural biases that algorithms typically exploit. The hardness of these instances is not an anomaly, but a reflection of the problem's inherent complexity when certain parameters are chosen. To establish complexity lower bounds, the authors rely on a single assumption regarding the exact algorithms for solving CSPs — "We only need to assume that this task is finished by dividing the original problem into subproblems". This should be true in algorithm design. As we know, at least until now, all exact algorithms for CSPs are based on divide-and-conquer style, such as backtracking methods.

This paper provides a constructive approach for understanding the complexity of problems. This constructive approach bridges the gap between theory and practice, as it provides concrete challenges for algorithm designers to overcome. Moreover, insights gained from analyzing these hard instances can inform the development of more robust and generalizable algorithms. This interplay between theory and practice is essential for advancing the field of computational complexity and algorithm design. In essence, theory helps identify barriers and opportunities, while practice refines and expands the range of solvable problems through algorithmic innovation.

On the structure and complexity of computational problems

Yun FAN

School of Mathematice and Statistics, Central China Normal University, Wuhan 430079, China

The paper discusses a special type of CSP, named Model RB, which has been proved to have an exact phase transition from satisfiability to unsatisfiability (K. Xu and W. Li. Exact phase transitions in random constraint satisfaction problems, Journal of Artificial Intelligence Research, 12(1): 93−103, 2000). Xu and Zhou explored what happened in the phase transition point. Inspired by the Gödel's self-referential proposition which is unprovable, they introduced the so-called "self-unsatisfiable constraint", "self-unsatisfiable variable" and "self-unsatisfiable instance" (called "self-unsatisfiable formula" in their paper). They find a way, called symmetry mapping, to exchange self-unsatisfiable instances to the instances with unique solution, and vise versa. Similarly to Gödel's diagonal argument, they proved by contradiction that any algorithm solving RB-problem has at least exponential complexity, i.e., Theorem 3.2 and its Corollary 3.3, which are really surprising results.

Computational problems take many various forms, for example, algebraic, combinatorial, number-theoretic, etc., which vary greatly from one to the other. Generally speaking, the richer the structure of the problem is, the simpler the algorithm of the problem is. Because: the structure would help us to reduce the solving of instances of the problem to a small number of instances. For example, to solve systems of linear equations of n variables, Gaussian elimination reduce systems of linear equations to the systems of linear equations which coefficient matrices have diagonal submatrices.

There are three situations.

The best situation is that by the reductions we can get an algorithm of polynomial time to solve the problem (e.g., solving systems of linear equations described as above).

The worst situation is that the problem is not reducible completely so that we can solve it only by exhaustive search. Xu and Zhou say this case is "extremely hard". For example, the general decoding problem (i.e., essentially speaking, given a subset and a point in a finite metric space, find the point in the subset which is closest to the given point) is extremely hard.

The situation in between is that, though the problem is reducible, we do not find an algorithm of polynomial time to solve it. For example, for the decoding problem of linear codes (i.e., in the decoding problem mentioned above, further assume that the finite metric space is a vector space and the given subset is a linear subspace), though we can use the linear structure to reduce the question to a smaller question about the quotient space, we still have no algorithm of polynomial time to solve it.

As Xu and Zhou said, the situation in between appears often difficult to deal with, however, the extremely hard situation is relatively not so difficult to deal with. Their work means that, though the problem RB can be reducible (e.g., reduce the RB-problem with n variables to the RB-problem with $n-1$ variables), but it does contain many hard instances which present some wonderful natures, so that the authors of the paper can spot the contradiction in their proof.

A big contribution of their work is that they find a new way to study the computation hardness, i.e., construct a subproblem containing the peculiar hard instances, and use it to estimate the complexity of the problem. Their work implies seemingly that such instances are usually appeared in the transitional areas. Thus their work suggested an insightful research point of view: if a problem has an exact phase transition similar to RB-problem, then it maybe interesting to explore what happened in the zone of the phase transition.

A very original innovation of the paper of Xu and Zhou is that they regard the computational problem they are studying as a finite formal system, just like the finite formal systems Gödel worked, and use the Gödel's diagonal argument, and some thing like that, to consider the hardness of the computational problem. This ideology is different from the common ideas in current complexity theory. To be more explicit, finite formal systems may be not algorithms, but algorithms are finite formal systems (mechanized finite formal systems). As Xu and Zhou said: "our work and current complexity theory are two paths that extend different frameworks from the same starting point in computer science." Within the new framework, modifying the Gödel's diagonal argument, they are successful in proving by contradiction that Model RB cannot be solved in $O(d^{cn})$ time for $0<c<1$.

The power of symmetry in proving lower bounds

Zhiguo FU

School of Information Science and Technology, Northeast Normal University, Changchun 130117, China

This paper offers a new perspective on computational complexity by comparing Gödel's and Turing's results on the limits of computation. Specifically, the authors propose a novel approach, viewing computational complexity as the limits of reasoning within bounded time, a perspective that cleverly connects with Gödel's work on proving the incompleteness theorem.

Innovatively, this paper adopts a constructive approach to studying the hardness of problems. The core of computational complexity lies in understanding and proving hardness. However, it is difficult to determine the complexity lower bounds, and we know very little how to characterize hardness effectively. And we are lack of effective methods to generate genuinely hard instances. The concept of NP-completeness only characterizes relative hardness by indicating that some problems may be harder than others, without explaining the underlying reasons. The famous physicist Richard Feynman's quote, "What I cannot create, I do not understand," aptly captures this dilemma. This research addresses the challenge by adopting a constructive method. The authors utilize the phase transition phenomenon to construct random Constraint Satisfaction Problem (CSP) instances based on a well-known CSP model, namely Model

RB. Their proof shows that the extreme hardness of the constructed instances comes from symmetry.

In detail, I think an inherent difficulty in proving hardness lies in the fact that our goal is to demonstrate that an object possesses a certain property related to hardness. However, we lack an efficient method for analyzing this property (i.e., determining whether a given object has this property). If such a method existed, it would contradict the hardness of the object we aim to prove. In this paper, the authors cleverly utilize symmetry to overcome this difficulty, which, as far as I can see, is the most significant technical innovation of this work. Specifically, the authors first use symmetry to construct random CSP instances. Subsequently, they demonstrate that both satisfiable and unsatisfiable instances can be generated at the phase transition point. Finally, by introducing the symmetry mapping, they prove that these satisfiable and unsatisfiable instances possess a certain symmetry, making them difficult to distinguish. I believe that this approach not only proves the hardness, but also helps to understand it intuitively. In addition, I find the assumption about CSPs made in this paper to be reasonable and acceptable.

In summary, this paper not only offers a new perspective on computational complexity but also demonstrates the power of construction and symmetry in proving complexity lower bounds. I believe that it contributes to a fresh understanding of computational hardness and paves the way for future research in this area.

A new direction for computational complexity

Angsheng LI

School of Computer Science and Engineering, Beihang University, Beijing 100191, China

This paper proposed a completely new approach to understanding the concept of “hardness” in designing algorithms.

A computational problem requires us to find a solution by reasoning based on the relationships of the data points of a problem. The solutions of a computational problem are embedded in the space of the data points of the computational problem. Finding a solution of a computational problem is to eliminate the uncertainty that blocks the way to finding the solution.

Apparently, there are uncertainties in finding a solution of a computational problem. Once the uncertainty that is essential for finding a solution of a computational problem is eliminated, a solution of the computational problem shows up.

When Juris Hartmanis defined the metric of computational complexity, he initially tried to propose an information measure of the hardness of a problem. That is, the hardness of a computational problem is the amount of uncertainty that must be eliminated for finding a solution of the problem. Hartmanis examined the metric of Shannon entropy. Due to the fact that Shannon’s entropy measures only the amount of uncertainty that is embedded in a random variable, which is invalid for measuring the amount of uncertainty that is embedded in finding a solution of a computational problem. For this reason, Harmanis defined the time complexity and space complexity simply based on the execution of a Turing machine.

When I visited Juris Hartmanis in 2008, Juris suggested me to give a new definition of information, that is, the “information” in “information processing” in computer science, instead of the “information” in end-to-end communication. I spent 8 years working on this project, eventually published my theory paper on structural information theory in 2016. Since 2016, the principles of structural information theory have experienced the first wave application investigations in a wide range of areas such as identification of chromatin topologically associating domains in cells, classification of cancer cells, network security, ranking algorithms, image identification, game design, natural language understanding, neural network learning etc. Experiments verify that structural information theory provides the principles for learning and intelligent strategies with remarkable advances in both precision and efficiency. From 2016 to 2024, I established the information science principles of artificial intelligence, providing the first mathematical principle of AI. The research so far explores that “information” is a scientific concept with deep theory and wide applications, instead of merely a communication concept as that in Shannon information theory, and that “information” is the key to open the information world.

The paper entitled: SAT Requires Exhaustive Search, by Ke Xu and Guangyan Zhou, persists the same idea in understanding the “hardness” of a computational problem. It shows that:

1. algorithm design is based on reasoning;
2. reasoning is based on relationships;
3. if there is no relationship among the data points, then there is no the basis for reasoning, and hence, there is no cues for designing algorithms;
4. if there is no cue for designing algorithms, meaning that there is no cue to reduce uncertainty, then the only way is to do exhaustive search.

The paper proposed a construction of hard instances based on a random model of constraint satisfaction problem, namely model RB, proposed more than 20 years ago.

The authors proved the hardness of the constructed instances by developing an approach similar to the proof of the incompleteness theorem of Gödel in 1931.

The paper shows:

1. extremely hard instances can be constructed;
2. extremely hard instances can only be constructed, while “natural” instances in everyday life may not be too hard;
3. extremely hard instances are easy to prove that they are hard.

The paper pioneered a white box approach to understanding the hardness of computational problems, leading to a new direction for computational complexity.

On the gap between syntax and semantics

Wanwei LIU

College of Computer Science and Technology, National University of Defense Technology, Changsha 410073, China

The theory of computability and computational complexity is probably the most "elegant" part of theoretical computer science. Many difficult but fundamental problems (including $\mathbf{P} \overset{?}{=} \mathbf{NP}$) just belong to this subject. So far, all efforts to prove "$\mathbf{P} = \mathbf{NP}$" and/or $\mathbf{P} \subsetneq \mathbf{NP}$ have failed. Although we cannot fully understand the most essential cause that makes this problem difficult, some thought-provoking results have still been obtained. In 1975, Baker, Gill and Solovay showed that one can construct both two languages A and B making $\mathbf{P}^A = \mathbf{NP}^A$ and $\mathbf{P}^B \subsetneq \mathbf{NP}^B$. Consequently, $\mathbf{P}$ and $\mathbf{NP}$ cannot be separated by *diagonalization*, meanwhile $\mathbf{P} = \mathbf{NP}$ cannot be proven merely by *analogy* (or, *simulation*).

In this paper, the authors present a new approach to understanding the essence of "hardness" of computing — they present a hard case construction approach, based on a special random model of constraint satisfaction problem (CSP, for short), called RB, which is proposed by Xu and Li in 2000 or so. For CSPs, non-brute-force computation vs. brute-force computation (instead of $\mathbf{P}$ vs. $\mathbf{NP}$) is reduced to "whether the whole problem can be replaced by a certain number of subproblems".

By constructing self-referential examples of CSP, they show that "*Model **RB** (with n variables and domain size d) cannot be solved in O($d^{c\cdot n}$) time for any constant* $0<c<1$",
and as the corollary, they have

"SAT *cannot be solved in time* $O(2^{c\cdot n})$ *for any* $0<c<1$".

This is definitely a landmark in computer science and mathematics, if the conclusion is correct.

From the perspective of logic, I was deeply attracted by the following insight pointed out in Xu and Zhou's paper:

"*... However, even if a problem is uncomputable, each instance might still be solved quickly, though no general (mechanical) method exists to handle all instances ...*"

—it reminds me the relation between *syntax* and *semantics*! As shown in Gödel's theorem of incompleteness: any first order system containing arithmetic axioms cannot be (completely) axiomatized by preserving soundness. Note that this great theorem never denies the *existence* of the system (or simply, formula set) that precisely containing FOL formulas which are arithmetically true. One may just "collect" them into a set as axioms, yet such system becomes "meaningless", because one have no way to decide whether a given formula belongs to it or not. Another example is, suppose, we have some function $f : \mathbb{N} \to \{0,1\}$, namely, f is the characteristic function of some subset of natural numbers, then can we always describe the mapping rule of f using some given "standard operations"?. Definitely not! In addition, we even know that the measure of such "describable functions" is 0, when taking all such functions into account. The reason is, each such function corresponds to a sequence of description with finite length. This might be the "gap" between syntax and semantics.

Then, turn to computation complexity, the authors point that

"*... The essence of computational hardness lies not in the distinction between deterministic polynomial computation and non-deterministic polynomial computation (i.e.,* **P** *vs.* **NP***), but in the contradiction between non-brute-force computation and brute-force computation. This distinction essentially questions whether syntax can replace semantics ...*"

In my opinion, the instances created in Xu and Zhou's paper can be seen as twin instances: although they share the same syntax, they differ in their semantics, and semantical feature cannot be captured by their (even, have no) syntactical counterpart.

Then, from Gödel's theorem (of incompleteness) to Tarski's (undefinability) theorem, then to Turing's "uncomputability", maybe the gap between syntax and semantics is the "manipulator" hidden behind them. Xu and Zhou's paper makes me notice this important issue that has once been ignored.

Proving computational hardness: uncertainty can be determined

Bin WANG

Academy of Mathematics and Systems Science, Chinese Academy of Sciences, Beijing 100190, China

Computational complexity theory is a fundamental area of computer science, extensively studied by researchers in mathematics, computer science, and statistical physics. The hardness of a problem stems from the uncertainty of its solvability, thus proving hardness equates to proving its uncertainty.

It is well known that the hardest instances of NP-complete problems are concentrated in the sharp transition region, where instances are either satisfiable or unsatisfiable, exhibiting the highest uncertainty. Although random 3-SAT can generate instances with uncertain satisfiability, this uncertainty has not been mathematically proven, lacking the foundation for a formal proof of hardness. Meanwhile, easy problems like 2-SAT and XORSAT exhibit sharp threshold results but are not suitable for hardness proofs. The fundamental reason behind this is that the correlation within the solution space (i.e., the set of all possible assignments) is a crucial factor influencing the complexity of the problem. Put simply, the more independent (less correlated) the solution space is, the harder the problem becomes. The assignments of 3-SAT are highly correlated, often agreeing on about half of the variables. This correlation causes the second moment to grow exponentially compared to the square of its expectation, leading to the failure of the second-moment method.

In contrast, the uncertainty of whether the RB instances are satisfiable or unsatisfiable can be rigorously proved. From Lemma 2.2 and Corollary 2.3 in Xu and Zhou's paper, both the probability that a random instance I is unsatisfiable and the probability that I has exactly one solution can be bounded below by a positive constant. That is to say, the status of I is determined to be uncertain. This property makes RB instances a viable foundation for establishing hardness results. From a mathematical perspective, proving the uncertainty of the RB model is significantly easier than that of 3-SAT. I think the key reason is that in the RB model, the domain size grows with the number of variables, changing the structure of the solution space. Specifically, this significantly weakens the correlation between assignments, making the solution space appear nearly independent. This independence enables the success of the second-moment method. Note that for any two assignments σ and τ, it holds that $\frac{\mathbf{Pr}(\sigma,\tau \text{ satisfy } I)}{\mathbf{Pr}(\sigma \text{ satisfies } I)\cdot\mathbf{Pr}(\tau \text{ satisfies } I)} \geqslant 1$. Furthermore, σ, τ are independent if and only if the equality holds for the above inequality. As shown by the authors, the RB model has $\frac{\mathbf{E}[X^2]}{\mathbf{E}[X]^2} \approx 1$ in the satisfiable region. Since $\frac{\mathbf{E}[X^2]}{\sum_{\sigma,\tau}\mathbf{Pr}(\sigma,\tau \text{ satisfy } I)} = 1$, and $\frac{\mathbf{E}[X]^2}{\sum_{\sigma,\tau}\mathbf{Pr}(\sigma \text{ satisfies } I)\cdot\mathbf{Pr}(\tau \text{ satisfies } I)} = 1$, then $\frac{\sum_{\sigma,\tau}\mathbf{Pr}(\sigma,\tau \text{ satisfy } I)}{\sum_{\sigma,\tau}\mathbf{Pr}(\sigma \text{ satisfies } I)\cdot\mathbf{Pr}(\tau \text{ satisfies } I)} \approx 1$. Thus, there is strong evidence that solution space of the RB model exhibits near-independence. Furthermore, the dominant assignment pairs contributing to $\mathbf{E}[X^2]$ are those with the largest distance (i.e., agreeing on no variables), strengthening this independence. This property also ensures that the slight change made by the symmetry mapping in Xu and Zhou's paper will not affect the remaining solution space. Precisely due to this property, Xu and Zhou can use the classical diagonalization method to prove, by contradiction, that exhaustive search is indispensable for eliminating uncertainty.

From a computational perspective, solving a combinatorial problem fundamentally involves compressing the solution space, i.e., efficiently filtering feasible solutions from a vast set of candidates. In statistical terms, this is analogous to dimensionality reduction. Interestingly, this paper demonstrates that for CSP and SAT problems, it is possible to construct extremely hard instances that are incompressible. The independence of the solution space prevents any method from significantly reducing the search space, ultimately leading to the incompressibility of the instances.

I find the mathematical derivations in Xu and Zhou's paper to be correct. Additionally, this paper introduces a mathematical assumption to its reasoning, which I find both necessary and well-justified. The challenge of proving impossibility results has a long history in mathematics. For example, it took over two thousand years to prove the impossibility of trisecting an arbitrary angle using only a straightedge and compass. Such impossibility results are inherently conditional —they hold under specific mathematical frameworks rather than in an unrestricted mathematical setting. The authors argue that the Turing machine represents a fundamental physical assumption. This perspective seems reasonable to me, as the Turing machine, in my view, does not impose any restrictions on mathematics. Incompressibility results are a type of mathematical impossibility result, and proving them requires introducing an assumption that restricts the mathematical framework. This necessity aligns with the historical approach to proving impossibility results.

Different phase transitions of random CSPs

Pan ZHANG

Institute of Theoretical Physics, Chinese Academy of Sciences, Beijing 100190, China

The phase transitions in random constraint satisfaction problems (CSPs), such as the K-SAT problem and the coloring problem, have been extensively studied in statistical physics as models of spin glass systems, which exhibit disorder and frustration. Research in statistical physics has focused on the structure of the solution space, linking it to average-case computational complexity and the phase transition between the "almost satisfiable phase" and the "almost unsatisfiable phase." The hardest instances typically lie near this phase transition.

Typical problems in spin-glass theory involve a fixed-length domain size. For example, in K-SAT, each variable has only two possible states. However, Model RB, studied by Xu and Zhou, differs significantly from standard random CSPs like k-SAT. In Model RB, the domain size $d = n^{\alpha}$ grows polynomially with the number of variables n. This growth leads to distinct properties compared to traditional CSPs studied in statistical physics.

1 Solution space correlations and domain size

Model RB's large domain size means that two assignments can be very different, with a large Hamming distance, and most pairs of assignments are nearly independent. This results in a solution space dominated by isolated configurations. In contrast, k-SAT problems, with their fixed-length Boolean variables, exhibit strong local correlations due to variable overlap in clauses. Near the phase transition, solutions form clusters with small Hamming distances, separated by energy or entropy barriers.

2 Critical behavior and phase transition sharpness

Model RB has an exact and mathematically tractable phase transition with a sharp threshold r_{cr}, determined by methods like the second-moment method. Instances at this threshold are either satisfiable or unsatisfiable due to violated constraints. In k-SAT, while a sharp satisfiability threshold exists, the critical region in the hard-to-solve phase is characterized by a proliferation of frozen clusters, where a finite fraction of variables share a partial configuration. This aspect remains challenging for rigorous mathematical methods.

3 Algorithmic hardness and energy landscapes

The exponential number of independent variables in Model RB may create a flat ground-state-energy landscape with exponentially many local minima, forcing algorithms to perform exhaustive searches rather than relying on local heuristics. This is analogous to the frozen phase in k-SAT, where solution clusters are hard to find without directly accessing the frozen variables. Additionally, Model RB avoids trivial unsatisfiability by balancing constraint density and domain growth, unlike k-SAT, which relies solely on clause density tuning.

In conclusion, Model RB's polynomial domain size growth makes its phase transition distinct from traditional random CSPs like k-SAT. These properties make Model RB a unique benchmark for rigorously studying computational limits. The interplay between Model RB and other CSPs enriches both computational complexity theory and statistical physics, highlighting its value as a bridge between these disciplines.