Self-reference as key to proving extreme hardness
——— Reply to the comment by Allender and Williams

Ke Xu[1], Guangyan Zhou[2]

[1]State Key Laboratory of Complex and Critical Software Environment, Beihang University, Beijing, 100083, China

kexu@buaa.edu.cn

[2]Department of Mathematics and Statistics, Beijing Technology and Business University, Beijing, 100048, China

zhouguangyan@btbu.edu.cn

Abstract

In this reply, we present two key arguments. First, the assumption used to establish the extreme hardness of self-referential CSP instances, which serves as a conceptual bridge to the diagonalization method, is both necessary and well-founded. Second, the extreme hardness of these instances stems not from the aforementioned assumption or any concrete computational model, but from the inherent self-referential property of the instances themselves.

Keywords: extreme hardness, brute-force computation, self-reference

The most important contribution of our paper [6] lies in redefining the foundational problem of computer science. More precisely, the most fundamental problem in computer science is non-brute-force computation vs. brute-force computation, rather than P vs. NP. Our paper and current computational complexity theory represent two distinct paths branching off from the same starting point in computer science. Specifically, non-brute-force computation vs. brute-force computation extends Gödel's framework, while P vs. NP extends Turing's framework. Moreover, self-reference serves as the key to proving extreme hardness—i.e., the necessity of brute-force computation. Although P vs. NP is a renowned Millennium Prize Problem, we aim not to emphasize it in our paper. The core argument of our paper [6] is that the P vs. NP is not the most fundamental problem. Instead, it stands as the greatest obstacle: it hinders the application of self-reference and diagonalization, which represents the most powerful technique for proving impossibility results. We believe that any discussion is feasible, but first we must identify the right problem—for the problem itself matters more than the result.

It is widely believed that algorithms have various forms; for instance, there are multiple distinct algorithms for linear programming problems. However, it should be noted that problems with such a variety of algorithms generally fall into the category of algebraic problems. Barak [2] has discussed possible algorithms for solving CSPs, noting a key distinction: unlike algebraic problems, which possess inherent structure, CSPs are fundamentally unstructured. Furthermore, he argues that this lack of structure makes the space of possible algorithms for solving CSPs much more limited, and that this

characteristic may help towards the goal of proving unconditional lower bounds.

The main results of our paper [6] hold under the assumption about exact algorithms for CSPs. This assumption is consistent with the reality of exact algorithms for CSPs. Allender and Williams [1] claim that there might be another sort of algorithm violating this assumption. But they also say that "We do not claim that an algorithm of this sort is likely to succeed". So our assumption has not been violated until now. For the algorithm proposed by Williams [4], it seems that this algorithm requires exponential space. Researchers do not include this type of algorithm in their comparative analysis of algorithms for CSPs. This is because even the exhaustive algorithm for CSPs only requires polynomial space. In algorithm design, space efficiency often takes precedence over time efficiency, especially in scenarios where storage resources are constrained or space bottlenecks directly limit system feasibility. Consequently, when developing alternative algorithms to replace the exhaustive algorithm for CSPs, satisfying the polynomial space requirement is not just a preferred optimization, but a fundamental constraint that must be met. Currently, no known algorithm can solve CSPs in $O(d^{cn})$ time for some constant $c < 1$ when the domain size d is very large. This is very similar to the case of k-SAT with very large k. For details, see Ref. [3] and the relevant references therein. To the best of our knowledge, current research on algorithms for CSPs generally omits explicit discussion of space complexity, while implicitly assuming polynomial space as a standard condition. When comparing the time complexity of different CSP algorithms, those requiring more than polynomial space are usually excluded to ensure a fair and meaningful comparison. Our paper [6] adheres to this established convention.

In [6], we constructed self-referential instances that inherently require exhaustive search, as in these cases no proper part can serve as a substitute for the whole. This approach is closely related to that of [5], which pursued complexity lower bounds through the construction of so-called "bad inputs". However, our constructed instances go beyond merely "bad inputs"; they constitute the worst-case inputs for any algorithm. The proof of our main results [6] is based on the assumption regarding exact algorithms for CSPs. However, as can be seen from the proof process, the reason why the diagonalization method can be used for proof by contradiction mainly relies on the nature of the constructed instances themselves. Specifically, the worst-case hardness exhibited by these instances does not originate from the aforementioned assumption or any concrete computational model; instead, this hardness is directly rooted in the self-referential property intrinsically embedded in the instances. Such a property allows for the straightforward construction of a counter-instance from the original, using a simple approach. The role of the assumption is purely instrumental: it acts as a bridge that links these self-referential instances to the diagonalization method. We believe that constructing self-referential instances is an effective (and perhaps the only) method for demonstrating the necessity of brute-force computation, and that this approach can also be applied to other problems that require brute-force computation in the worst-case scenario.

The goal of our paper [6] is not to claim that we have solved a Millennium Prize Problem, then sit back and await fortune and fame. Rather, our purpose is to illuminate what constitutes the foundational problem to pursue. We hope that people, especially

bright young minds, will not continue to waste their time on a non-foundational problem. Many people have extensive experience in handling numerous cases involving flawed proofs of the P vs. NP problem. The case of our paper [6], however, is entirely distinct from previous ones: the core issue lies not in whether a proof of P vs. NP is correct, but rather in whether P vs. NP itself constitutes the right question to pose. We have noticed that there have already been some online discussions [1] regarding the P vs. NP problem itself. In light of this, we call for broader participation and advocate for full openness and transparency in all aspects.

Finally, we believe that any discussion conducted around academic issues themselves is bound to have a positive impact on the development of academia, even though the process may be quite tortuous. Prof. Gregory Chaitin once told us: "*Of course, you are having problems convincing the community, which is only natural. It will take time, but you should persist!*" Everything that is happening now proves how correct his foresight was.

References

[1] E. Allender and R. Williams. Comment on "SAT Requires Exhaustive Search". Frontiers of Computer Science, 2026, 20(1): 2001405.

[2] B. Barak. Structure vs combinatorics in computational complexity. Bulletin of the European Association for Theoretical Computer Science, 112 (2014).

[3] T. Hertli, I. Hurbain, S. Millius, R. A. Moser, D. Scheder, M. Szedlák. The PPSZ algorithm for constraint satisfaction problems on more than two colors. In International Conference on Principles and Practice of Constraint Programming. Cham: Springer International Publishing, 2016: 421-437.

[4] R. Williams. A new algorithm for optimal 2-constraint satisfaction and its implications. Theoretical Computer Science, 348(2-3):357–365, 2005.

[5] R. Williams. The power of constructing bad inputs. Bulletin of the European Association for Theoretical Computer Science, 139 (2023).

[6] K. Xu and G. Zhou. SAT requires exhaustive search. Frontiers of Computer Science, 2025, 19(12): 1912405

[1] See https://rjlipton.com/2009/07/03/is-pnp-an-ill-posed-problem/