Mathematical
Structures and Modeling
2018. N. 4(48). PP. 53–60

UDC 510.52
DOI: 10.25513/2222-8772.2018.4.53-60

ALL MAXIMALLY COMPLEX PROBLEMS ALLOW SIMPLIFYING DIVIDE-AND-CONQUER APPROACH: INTUITIVE EXPLANATION OF A SOMEWHAT COUNTERINTUITIVE LADNER'S RESULT

Olga Kosheleva
Ph.D. (Phys.-Math.), Associate Professor, e-mail: olgak@utep.edu
Vladik Kreinovich
Ph.D. (Phys.-Math.), Professor, e-mail: vladik@utep.edu

University of Texas at El Paso, El Paso, Texas 79968, USA

Abstract. Ladner's 1975 result says that any NP-complete problem — i.e., in effect, any maximally complex problem — can be reduced to solving two easier problems. This result sounds counterintuitive: if a problem is maximally complex, how can it be reduced to simpler ones? In this paper, we provide an intuitive explanation for this result. Our main argument is that since complexity and easiness-to-divide are not perfectly correlated, it is natural to expect that a maximally complex problem is not maximally difficult to divide. Our related argument is that — as this result shows — NP-completeness is a sufficient but not a necessary condition for a problem to be maximally complex; how to come up with a more adequate notion of complexity is still an open problem.

Keywords: NP-complete, divide-and-conquer approach, Ladner's result, maximally complex problems.

1. Formulation of the Problem

Ladner's result: a brief description and why it is counterintuitive. Ladner's 1975 result (see, e.g., [1,3]) says that any NP-complete problem — i.e., in effect, any maximally complex problem — can be reduced to solving two easier problems.

This result sounds counterintuitive: if a problem is maximally complex, how can it be reduced to simpler ones?

What we do in this paper. In this paper, we provide an intuitive explanation for this result.

To provide this explanation, we first need to recall what is NP-completeness and what exactly is Lander's result. This will be done in the remaining part of this section. The next section will contain our explanation.

Which algorithms are feasible: a brief reminder. The definition of NP-completeness is based on the notion of a feasible algorithm. Thus, in order to explain what is NP-completeness, we first need to explain what is a feasible algorithm.

This notion formalizes the intuitive idea that while some algorithms are practically feasible, other algorithms require so much computation time that they are not practically possible. For example:

- an algorithm that requires computation time n^2, where n is the length of the input, is usually practical, while
- an algorithm that require computation time 2^n is not — since even for reasonable-size inputs $n \approx 500$, the resulting computation time would exceed the lifetime of the Universe.

Usually:

- polynomial-time algorithms — i.e., algorithms A whose running time $t_A(x)$ on each inputs x do not exceed some polynomial $P(n)$ of the length $n = \text{len}(x)$ of the input – are feasible, while
- algorithms for which $t^w_A(n) \overset{\text{def}}{=} \max\limits_{x:\text{len}(x)\leqslant n} t_A(x)$ is not bounded by any polynomial are not feasible.

Because of this, usually, an algorithm is defined to be feasible if it is polynomial-time; see, e.g., [2,4].

Comment. It is well known that this definition is not perfect; e.g.:

- an algorithm that takes time $t^w(n) = 10^{500} \cdot n$ is polynomial-time but not practically feasible, while
- an algorithm that takes time $\exp(10^{-20} \cdot n)$ is practically feasible but not polynomial-time.

However, this is the most adequate definition we have.

What is a "problem". Another important notion needed to explain what is NP-complete is the notion of the class NP. This notion comes from the attempt to formally describe the intuitive idea of a (general) problem.

In all practical situations, when we formulate a problem, we expect that there is a clear and feasible way to check whether a given candidate for a solution is indeed what we want.

For example, in mathematics, the main activity is proving theorems. Once we have a formulation,

- finding a proof is difficult — it may take hundreds of years for the whole mathematical community — but
- once a detailed proof is presented, it is relative easy to check step-by-step that the proof is correct: e.g., computer programs for checking proofs were available already in the 1960s, when computers were much slower.

Similarly, in physics,

- finding a law that explains all the observations may be difficult, but
- once the explaining formula is found, checking that all the observations are consistent with this formula is straightforward.

In engineering,

- it is sometimes difficult to find a design that satisfies the given specifications — e.g., designing a compact antenna for a smart phone requires complex computations — but

- once a design is presented, it is usually straightforward to check that this design satisfies all the desired specifications.

In all these cases, we are given some information x, and we want to find some object y that satisfies a feasible property $C(x, y)$.

In all these cases, an additional requirement is that the length of y should be feasible, i.e., similarly to time, that the length of y not exceed some polynomial of the length of x: $\text{len}(y) \leqslant P_\ell(\text{len}(x))$. Indeed:

- In mathematics, if a proof is too long, it is not possible to check it.
- In physics, if a formula is too complex, it is worthless: we could as well use, e.g., piece-wise linear interpolation of the experimental data.
- In engineering, if the design is too complicated, it is not feasible to complement, etc.

In all these cases, we have a feasible algorithm $C(x, y)$ and we have a polynomial $P_\ell(n)$. The problem is: given x, find y such that $C(x, y)$ and

$$\text{len}(y) \leqslant P_\ell(\text{len}(x)).$$

Such a solution y is not always possible. So, before we solve the problem of actually finding y, we need to check whether such a y exists — or, in set-theoretic terms, whether x belongs to the set S of such x's for which the corresponding y exists.

The class of all such checking problems is known as NP, for Non-deterministic Polynomial. This name came from the fact that in all such problem,

- once we guessed a solution y,
- we can check, feasibly (i.e., in polynomial time) whether this guess is indeed a solution.

In other words, we can solve this problem in polynomial time if, in addition to computations, we allow guesses — such general "computations" are known as *non-deterministic*.

Comment. Not all the problems belong to the class NP. For example, if we are looking for an optimal solution, then there is no easy general way to check that a given candidate is indeed an optimal solution: that would require comparing it with all other possible solutions, and there are usually exponentially many of them.

However, in practice, what we really want is, e.g., a solution and a proof that this solution is optimal — it is always feasibly checkable, otherwise this problem is not practically useful.

Is NP equal to P? Some problems from the class NP can be solved by feasible (polynomial-time) algorithms. The class of all such problems is usually denoted by P.

Most computer scientists believe that there are problems that cannot be solved by feasible algorithms, i.e., that NP $\neq$ P. However, no one has been able to prove or disprove this, it is still an open problem. In this paper, we will operate under the assumption that NP $\neq$ P.

The notion of reduction. The last auxiliary notion that we need to explain what is NP-completeness is the notion of reduction.

Often, one general problem can be reduced to another one, in the sense that for each particular instance of the first problem we can feasible compute one (or several) instances of the second problem so that, based on the solution(s) to the second problem, we can feasibly compute the solution to the original problem.

For example, equations $a\cdot x+\frac{b}{x}=c$ can be reduced — by multiplying by x — to quadratic equations. Similarly, the general problem of quadratic equations can be reduced to solving cubic equations: to perform this reduction, it is sufficient to add the term $0\cdot x^3$ to the left-hand side of the quadratic equation $a\cdot x^2+b\cdot x+c=0$.

The notion of NP-completeness. If a problem A can be reduced to a problem B, this means, intuitively, that the problem B is:

- either more complex than the problem A (as in the case of quadratic vs. cubic equations)
- or of the same complexity (as in the first example of reduction).

Thus, if we have a problem from the class NP to which every other problem from NP can be reduced, then such a problem is clearly maximally complex. Such problems are called *NP-complete.*

Ladner's result. In 1975, Richard E. Ladner proved that, if NP $\neq$ P, then every NP-complete set S can be represented as a union $S=S_1\cup S_2$ of two disjoint set sets S_1 and S_2 none of which is NP-complete; see, e.g., [1,3].

Why this result is somewhat counterintuitive. What Ladner's result shows is that we can reduce the original NP-complete problem of checking whether a given input belongs to the set S to two easier problems of checking whether $x\in S_1$ or $x\in S_2$. Once we have a positive answer to one of the two new problems, this means that $x\in S$ — and vice versa, if $x\in S$, this means that either $x\in S_1$ or $x\in S_2$.

But if the original problem S is NP-complete — i.e., maximally complex — how can it be reduced to simpler ones? That would make this problem easier.

What we do in this paper. In this paper, we provide an intuitive explanation of Ladner's result — an explanation that, hopefully, makes it less counterintuitive.

2. Our Main Argument

Let us reformulate out situation in general terms. To describe our explanation, let us reformulate the situation in general terms. We have two functions:

- a function that describes the complexity of a problem A; we will denote this function by $f(A)$, and
- a function that describe to what extent the given problem is difficult to divide into two easier-to-handle ones; we will denote the second function by $g(A)$.

Ladner's result is that problems that maximize $f(A)$ do not maximize $g(A)$ — although these two functions are clearly strongly correlated. How can we explain this?

The two functions are different. First, let us notice that the functions $f(A)$ and $g(A)$ are different — in the sense that they lead to different order between problems.

A classical example of this difference comes from *linear programming (LP)* — i.e., checking whether a given finite set of linear inequalities $\sum\limits_{j=1}^{n} a_{ij} \cdot x_j \leqslant b_i$, with known a_{ij} and b_i and unknown x_j, is consistent. This problem is known to be maximally difficult to parallelize – P-hard — thus, maximally difficult to divide into two easier-to-handle cases. However, this is *not* a maximally complex problem at all: it is actually in the class P; see, e.g., [4].

Thus, we have a clear case when the function $g(A)$ attains its maximum, while the value of the first function $f(A)$ is much much smaller than its maximum value. Thus, the functions $f(A)$ and $g(A)$ are indeed different.

When do maxima of two functions always coincide? In general, if we have two functions on a set, when do these two function attains their maxima on exactly the same elements?

In general, we are talking about conditional maxima, i.e., maxima on a subset $\mathcal{A}$ of the set U of all the elements. One can easily check that the two functions leads to the maximal elements on each subset $\mathcal{A}$ of the set U of all elements if and only if they generate the same order:

Proposition 1. *Let $f, g : U \to L$ be two functions from a set U to a partially ordered set L. Then, the following two conditions are equivalent to each other:*

- *for every subset $\mathcal{A} \subset U$, the sets of all f-largest and g-largest elements of $\mathcal{A}$ coincide:*

$$\{A \in \mathcal{A} : \forall B \in \mathcal{A}(f(B) \leqslant f(A))\} = \{A \in \mathcal{A} : \forall B \in \mathcal{A}(g(B) \leqslant g(A))\};$$

- *the functions f and g lead to the same order, i.e., for all $A, B \in U$, we have $f(A) \leqslant f(B)$ if and only if $g(A) \leqslant g(B)$.*

Proof. Clearly, if f and g lead to the same order, then the set of f-largest and g-largest elements coincide.

Vice versa, if the sets of f-largest and g-largest elements always coincide, then, for all A and B, we can consider the set $\mathcal{A} = \{A, B\}$. If $f(A) \leqslant f(B)$, this means that B is in the set of f-largest elements. Thus, B is also in the set of g-largest elements, and therefore, $g(A) \leqslant g(B)$.

Same argument shows that if $g(A) \leqslant g(B)$ then $f(A) \leqslant f(B)$. The proposition is proven.

This is related to Kendall's tau. If the two functions $f(A)$ and $g(A)$ were perfectly aligned, we would always have $f(A) \leqslant f(B)$ if and only if $g(A) \leqslant g(B)$. We know that our two functions are strongly related but not perfectly aligned. Thus, for pairs (A, B), the f- and g-orders sometimes differ.

In statistics, such a situation is well-known, it is described by *Kendall's tau* (see, e.g., [5]), which is defined as $\tau = 2r - 1$, where r is the proportion of all the pairs (A, B) for which f- and g-orders coincide.

The fact that for some pairs, f- and g-orders differ means that in our case, we have $r < 1$.

What is the probability that an f-largest element is also g-largest: an intuitive estimate and the resulting explanation of Ladner's result. An element A is f-largest if $f(B) \leqslant f(A)$ for all B. What is the probability that this element is also g-largest, i.e., that we have $g(B) \leqslant g(A)$ for all B?

For each B, the probability that $g(B) \leqslant g(A)$ is equal to r — by definition of the quantity r. We have no reason to believe that these is a positive or negative correlation between inequalities corresponding to different elements B. Thus, it is reasonable to assume that these inequalities are independent.

Due to the independence assumption, the probability that we have $g(B) \leqslant g(A)$ for all B can be estimated as the product of the corresponding probabilities, i.e., as r^N, where N denotes the overall number of elements in the set U.

In our case, N is large, so r^N is practically 0. Thus, the probability that an f-largest (i.e., NP-complete) problem is also g-largest (i.e., maximally difficult to reduce to two easier-to-solve problems) is close to 0. This is perfectly in line with Ladner's result.

Comment. Of course, our intuitive explanation does not explain the whole result: we explained, in effect, why it is reasonable to believe that "almost all" NP-complete problems are not maximally difficult to divide, but this does not explain that this is true for *all* NP-complete problems. This additional explanation comes from the fact that, by definition, all NP-complete problems are (kind of) equivalent to each other — in the sense that every two problems can be reduced to each other. Thus, it is reasonable to expect that what is true for one NP-complete problem is also true for all of them.

3. Our Related Argument

Ladner's result is somewhat counterintuitive: a brief reminder. In theoretical computer science, for problems from the class NP, NP-completeness is usually identified with being maximally complex. So, if a problem is not NP-complete, this means that it is not as complex as the NP-complete problems.

From this viewpoint, the possibility to reduce an NP-complete problem to two non-NP-complete ones is counterintuitive: maximally complex problem is reduced to two less complex ones. Intuitively, if maximally complex problem is reduced to two problems, at least one of them should be of the same complexity as the original problem.

Ladner's result becomes even more counterintuitive if we consider its consequence proven in [1] about a similar notion of function complexity: there is a case when the composition of two function is of maximal possible complexity, while both composed functions are easier to compute.

But is the usual identification correct? Ladner's result becomes counterintuitive if we identify maximally complex problems with NP-complete ones. But let us recall where this identification comes from. It comes from the fact that *if* the problem is NP-complete — i.e., if every problem from the class NP can be reduced to this problem — then this problem is clearly of the largest possible complexity. This is clear and intuitive.

However, there are no intuitive arguments for saying that if the problem is not NP-complete, then it must be easier. In fact, intuitively, what Ladner's result shows is that at least one of the sets S_1 or S_2, while not NP-complete, is, from the intuitive viewpoint, almost as complex as NP-complete problems.

Resulting explanation and the resulting open problem. So, it is not Ladner's result itself that is counterintuitive, what makes this result counterintuitive is the identification of NP-completeness and maximal complexity. What this result shows is that this not-very-justified association is counterintuitive — while every NP-complete problem is maximally complex, this result clearly shows that there are problems which are intuitively maximally complex but not NP-complete.

It is thus desirable to come up with a more intuitive definition of maximally complex problems. In the corresponding definition, if a maximally complex set S is a union $S = S_1 \cup \ldots \cup S_m$ of finitely many sets from the class NP, at least one of the sets S_i should be maximally complex in the same sense. How to come up with such a definition is an open problem.

Acknowledgments

This work was supported in part by the US National Science Foundation grant HRD-1242122 (Cyber-ShARE Center of Excellence).

References

1. Hemaspaandra L.A., Spakowski H. Team diagonalization // ACM SIGACT News. 2018. Vol. 48, No. 3. P. 51–61.
2. Kreinovich V., Lakeyev A., Rohn J., Kahl P. Computational complexity and Feasibility of Data Processing and Interval Computations. Kluwer, Dordrecht, 1997.
3. Ladner R. On the structure of polynomial time reducibility // Journal of the ACM. 1975. Vol. 22, No. 1. P. 155–171.
4. Papadimitriou C.H. Computational Complexity. Pearson, Boston, Massachusetts, 1993.
5. Sheskin D.J. Handbook of Parametric and Nonparametric Statistical Procedures. Chapman and Hall / CRC, Boca Raton, Florida, 2011.

ВСЕ МАКСИМАЛЬНО СЛОЖНЫЕ ПРОБЛЕМЫ ДОПУСКАЮТ УПРОЩЕНИЕ ПРИНЦИПОМ «РАЗДЕЛЯЙ И ВЛАСТВУЙ»: ИНТУИТИВНОЕ ОБЪЯСНЕНИЕ НЕСКОЛЬКО ПРОТИВОРЕЧИВОГО РЕЗУЛЬТАТА ЛАДНЕРА

О. Кошелева
к.ф.-м.н., доцент, e-mail: olgak@utep.edu
В. Крейнович
к.ф.-м.н., профессор, e-mail: vladik@utep.edu

Техасский университет в Эль Пасо, США

Аннотация. Результат Ладнера 1975 года говорит о том, что любая NP-полная проблема, то есть, по сути, любая максимально сложная проблема, может быть сведена к решению двух более простых задач. Этот результат звучит парадоксально: если проблема максимально сложна, как её можно свести к более простым? В этой статье мы даём интуитивное объяснение этому результату. Наш главный аргумент состоит в том, что, поскольку сложность проблемы и лёгкость разделения проблемы не вполне коррелированы, естественно ожидать, что максимально сложную задачу не обязательно настолько же сложно разделить. Наш связанный с этим аргумент состоит в том, что, как показывает этот результат, NP-полнота является достаточным, но не необходимым условием, чтобы задача была максимально сложной; как придумать более адекватное понятие сложности — по-прежнему остаётся открытой проблемой.

Ключевые слова: NP-полный, принцип «разделяй и властвуй», результат Ладнера, максимально сложные проблемы.

Дата поступления в редакцию: 17.10.2018