Adding Constraints – A (Seemingly Counterintuitive but) Useful Heuristic in Solving Difficult Problems

Olga Kosheleva, Martine Ceberio, and Vladik Kreinovich

University of Texas at El Paso
El Paso, TX 79968, USA
{olgak,mceberio,vladik}@utep.edu

Abstract. Intuitively, the more constraints we impose on a problem, the more difficult it is to solve it. However, in practice, difficult-to-solve problems sometimes get solved when we impose additional constraints and thus, make the problems seemingly more complex. In this methodological paper, we explain this seemingly counter-intuitive phenomenon, and we show that, dues to this explanation, additional constraints can serve as a useful heuristic in solving difficult problems.

Keywords: constraints, algorithmic problems, heuristics

Commonsense intuition: the more constraints, the more difficult the problem. Intuitively, the more constraints we impose on a problem, the more difficult it is to solve it.

For example, if a university has a vacant position of a lecturer in Computer Science Department, and we want to hire a person with a PhD in Computer Science to teach the corresponding classes, then this hiring is a reasonably easy task. However, once we impose constraints: that the person has several years of teaching experience at similar schools and has good evaluations to show for this experience, that this person's research is in the area close to the classes that he or she needs to teach, etc., then hiring becomes a more and more complicated task.

If a person coming to a conference is looking for a hotel to stay, this is usually an easy problem to solve. But once you start adding constraints on how far this hotel is from the conference site, how expensive it is, how noisy it is, etc., the problems becomes difficult to solve.

Similarly, in numerical computations, unconstrained optimization problems are usually reasonably straightforward to solve, but once we add constraints, the problems often become much more difficult.

Sometimes constraints help: a seemingly counterintuitive phenomenon. In practice, difficult-to-solve problems sometimes get solved when we impose additional constraints and thus, make the problems seemingly more complex.

Sometimes this easiness to solve is easy to explain. For example, when a traveler prefers a certain hotel chain, and make this chain's brand name a constraint, then making reservations in a small town is usually not a difficult problem to solve, because in this town, there is usually only one hotel from this chain.

However, in other cases, the resulting easiness-to-solve is not so easy to explain.

Many such examples come from mathematicians solving practical problems. For example, in application problems, mathematicians often aim for an optimal control or an optimal design. To a practitioner, this desire for the exact optimum may seem like a waste of time. Yes, it is desirable to find an engineering design with the smallest cost under the given constraints – or, vice versa, with the best performance under the given cost constraints – but since we can predict the actual consequences of each design only approximately, wasting time to exactly optimize the approximately optimize the approximately known function does not seem to make sense. If we only know the objective function $f(x)$ with accuracy $\varepsilon > 0$ (e.g., 0.1), then once we are within ε of the maximum, we can as well stop.

In some cases, it is sufficient to simply satisfy some constraint $f(x) \geq f_0$ for some value f_0. However, from the algorithmic viewpoint, often, the best way to solve this problem is to find the maximum of the function $f(x)$ on a given domain – by equating partial derivatives of $f(x)$ to 0. If there is a value x for which $f(x) \geq f_0$, then definitely $\max\limits_y f(y) \geq f_0$, so the place x where the function $f(y)$ attains its maximum satisfies the desired constraint. In other words, by imposing an additional constraint – that not only $f(x) \geq f_0$, but also that $f(x) = \max\limits_y f(y)$ – we make the problem easier to solve.

In theoretical mathematics, a challenging hypothesis often becomes proven when instead of simply looking for its proof, we look for proofs that can be applied to other cases as well – in other words, when we apply an additional constraint of generalizability; see, e.g., [16] and references therein.

Similarly, interesting results about a physical system become proven in the realm of rigorous mathematics, while, due to the approximate character of the model, arguments on the physical level of rigor would be (and often are) sufficient.

In engineering and science, often, problems get solved when someone starts looking not just for a solution but for a solution that satisfies additional constraints of symmetry, beauty, etc. – or when a physicist looks for a physical theory that fits his philosophical view of the world; a large number of examples how the search for a beautiful solution helped many famous mathematicians and physicists – including Bolzmann and Einstein – are described in [8].

In software design, at first glance, additional constraints imposed by software engineering – like the need to have comments, the need to have simple modules, etc. – seem to make a problem more complicated, but in reality, complex designs often become possible only after all these constraints are imposed.

This phenomenon extends to informal problems as well. For example, in art, many great objects have been designed within strict requirements on shape, form, etc. – under the constraints of a specific reasonable regulated style of music, ballet, poetry, painting, while free-form art while seemingly simpler and less restrictive, does not always lead to more impressive art objects. Some people find personal happiness when accepting well-regulated life rules – e.g., within a traditional religious community – while they could not find personal happiness in their earlier freer life.

How can we explain this seemingly counter-intuitive phenomenon?

Analysis of the problem. By definition, when we impose an additional constraint, this means that some alternatives which were originally solutions to the problem, stop being such solutions – since we impose extra constraints, constraints that are not always satisfied by all original solutions.

Thus, the effect of adding a constraint is that the number of solution decreases. At the extreme, when we have added the largest possible number of constraints, we get a unique solution.

It turns out that this indeed explains why adding constraints can make the problems easier.

Related known results: the fewer solutions, the easier to solve the problem. Many numerical problems are, in general, algorithmically undecidable: for example, no algorithm can always find a solution to an algorithmically defined system of equation or find a location of the maximum of an algorithmically defined function; see, e.g., [1, 2, 4–6, 17, 18, 22].

The proofs of most algorithmic non-computability results essentially use functions which have several maxima and/or equations which have several solutions. It turned out that this is not an accident: uniqueness actually implies algorithmic computability. Such a result was first proven in [19], where an algorithm was designed that inputs a constructive function of one or several real variables on a bounded set that attains its maximum on this set at exactly one point – and computes this global maximum point. In [20], this result was to constructive functions on general constructive compact spaces.

In [12, 14], this result was applied to design many algorithms: from optimal approximation of functions to designing a convex body from its metric to constructive a shortest path in a curved space to designing a Riemannian space most tightly enclosing unit spheres in a given Finsler space [7]. Several efficient algorithms based on uniqueness have been described in [9–11].

On the other hand, it was proven that a general algorithm is not possible for functions that have exactly two global maxima or systems that have exactly two solutions; see, e.g., [12–15, 17].

Moreover, there are results showing that for every m, problems with exactly m solutions are, in general, more computationally difficult than problems with $m-1$ solutions; see, e.g., [21].

Resulting recommendation. The above discussion leads to the following seemingly counter-intuitive recommendation: If a problem turns out to be too complex to solve, maybe a good heuristic is to add constraints and make it more complex.

For example, if the problem that we have difficulty solving is an applied mathematical problem, based on an approximate description of reality, maybe a good idea is not to simplify this problem but rather to make it more realistic.

This recommendation may sound counter-intuitive, but applied mathematicians know that often, learning more about the physical or engineering problem helps to solve it.

This can also be applied to education. If students have a hard time solving a class of problems, maybe a good idea is not to make these problems easier, but to make them more complex. Again, at first glance, this recommendation may sound counter-intuitive, but in pedagogy, it is a known fact: if a school is failing, the solution is usually not to make classes easier – this will lead to a further decline in knowledge. Anecdotal evidence shows that a turnaround happens when a new teacher starts giving students more complex more challenging problems – and this boosts their knowledge.

This recommendation is in line with a general American idea – that to be satisfying, the job, among other things, must be a challenge.

Caution. Of course, it is important not to introduce so many constraints that the problem simply stops having solutions at all. Since it is difficult to guess which level of constraints will lead to inconsistency, it may be a good idea to simultaneously several different versions of the original problem, with different number of constraints added – this way, we will hopefully be able to successfully solve one of them.

Acknowledgments. This work was supported in part by the National Science Foundation grants HRD-0734825 and DUE-0926721 and by Grant 1 T36 GM078000-01 from the National Institutes of Health.

References

1. Aberth, O.: Precise Numerical Analysis Using C++, Academic Press, New York (1998)
2. Beeson, M.J.: Foundations of Constructive Mathematics. Springer-Verlag, New York (1985)
3. Bishop, E.: Foundations of Constructive Analysis, McGraw-Hill, New York (1967)
4. Bishop E., Bridges, D.S.: Constructive Analysis, Springer, New York (1985)
5. Bridges, D.S.: Constructive Functional Analysis, Pitman, London (1979)
6. Bridges, D.S., Via, S.L.: Techniques of Constructive Analysis, Springer-Verlag, New York (2006)
7. Busemann, H.: The Geometry of Geodesics, Dover Publ., New York (2005)
8. Chandrasekhar, S.: Beauty and the quest for beauty in science. Physics Today 32(7), 25–30 (1979); reprinted in 62(12), 57–62 (2010)
9. Kohlenbach, U.: Theorie der majorisierbaren und stetigen Funktionale und ihre Anwendung bei der Extraktion von Schranken aus inkonstruktiven Beweisen: Effektive Eindeutigkeitsmodule bei besten Approximationen aus ineffektiven Eindeutigkeitsbeweisen, Ph.D. Dissertation, Frankfurt am Main (1990), in German
10. Kohlenbach, U.: Effective moduli from ineffective uniqueness proofs. An unwinding of de La Vallée Poussin's proof for Chebycheff approximation. Annals for Pure and Applied Logic 64(1), 27–94 (1993)

11. Kohlenbach, U.: Applied Proof Theory: Proof Interpretations and their Use in Mathematics. Springer Verlag, Berlin-Heidelberg (2008)
12. Kreinovich, V.: Uniqueness implies algorithmic computability, Proceedings of the 4th Student Mathematical Conference, Leningrad University, Leningrad, 19–21 (1975), in Russian
13. Kreinovich, V.: Reviewer's remarks in a review of Bridges, D.S.: Constrictive functional analysis, Pitman, London (1979); Zentralblatt für Mathematik 401, 22–24 (1979)
14. Kreinovich, V.: Categories of space-time models, Ph.D. dissertation, Novosibirsk, Soviet Academy of Sciences, Siberian Branch, Institute of Mathematics (1979), in Russian
15. Kreinovich, V.: Physics-motivated ideas for extracting efficient bounds (and algorithms) from classical proofs: beyond local compactness, beyond uniqueness.In: Abstracts of the Conference on the Methods of Proof Theory in Mathematics, Max-Planck Institut für Mathematik, Bonn, Germany, June 3–10 (2007) p. 8.
16. Kreinovich, V.: Any (true) statement can be generalized so that it becomes trivial: a simple formalization of D. K. Faddeev's belief. Applied Mathematical Sciences 47, 2343–2347 (2009)
17. Kreinovich, V., Lakeyev, A., Rohn, J., Kahl, P.: Computational complexity and feasibility of data processing and interval computations, Kluwer, Dordrecht (1998)
18. Kushner, B.A.: Lectures on Constructive Mathematical Analysis, Amer. Math. Soc., Providence, Rhode Island (1984).
19. Lacombe, D.: Les ensembles récursivement ouvert ou fermés, et leurs applications à l'analyse récurslve. Compt Rend. 245(13), 1040–1043 (1957)
20. Lifschitz, V.A.: Investigation of constructive functions by the method of fillings, J. Soviet Math. 1, 41–47 (1973)
21. Longpré, L., Kreinovich, V., Gasarch, W., Walster, G.W.: m Solutions Good, $m-1$ Solutions Better. Applied Math. Sciences 2(5), pp. 223–239 (2008)
22. Pour-El, M., Richards, J.: Computability in Analysis and Physics. Springer-Verlag, New York, (1989)