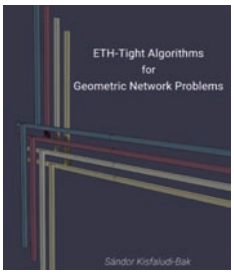
In de verdediging

| In defence



ETH-Tight Algorithms for Geometric Network Problems
Sándor Kisfaludi-Bak

In June 2019 Sándor Kisfaludi-Bak from the Eindhoven University of Technology successfully defended his PhD thesis with the title *ETH-Tight Algorithms for Geometric Network Problems*. Sándor carried out his research under the supervision of prof. dr. Mark de Berg and prof. dr. Hans Bodlaender (Utrecht University). He received his degree cum laude, a very rare distinction that is only given for outstanding results that have been obtained with a very high level of independence. Moreover, his thesis also received the distinguished dissertation award from the European Association for Theoretical Computer Science (EATCS).

During his PhD Sándor worked on algorithms for geometric network problems. In his thesis he presented a general framework to develop sub-exponential algorithms for certain types of geometric intersection graphs. This framework applies to many different problems and leads to algorithms that are both faster and more general than what was known. At the moment Sándor is a postdoc at the Algorithms and Complexity Group of the Max Planck Institute for Informatics, in Saarbrücken, Germany.

Geometric networks, algorithms and complexity

In geometric networks vertices correspond to points in some geometric space and edges indicate whether there is some kind of connection between these points.

A lot of problems from graph theory have geometric variants when considering geometric networks. Think of the Traveling Salesman Problem for example, or TSP for short. In the TSP a complete undirected graph G with positive weights on the edges is given. The goal is to compute a cycle visiting every vertex exactly once and having minimum weight. In the geometric variant, called the Euclidean TSP, the input is a set of n points in $\mathbb{R}^d$ and the goal is to find a cycle of minimum Euclidean length visiting all the points.

In order to solve combinatorial problems, algorithms are used. When looking for the most efficient algorithm for a given problem, it is useful to have some method to give lower bounds: to prove that improving the algorithm's running time beyond some point is not possible. Unfortunately, such strong claims can rarely be made: for most combinatorial problems, there is no good methodology to prove unconditional lower bounds. For this reason, most lower bounds are conditional, i.e., only hold on the basis that some complexity theoretic assumption holds.

Conditional lower bounds are usually done via *reductions:* proving that a problem is at least as hard as some older problem that researchers have failed to improve upon for decades. Under the condition that the older problem is indeed not solvable within a certain amount of time, this then proves that the problem at hand

Pas gepromoveerden brengen hun werk onder de aandacht. Heeft u tips voor deze rubriek of bent u zelf pas gepromoveerd? Laat het weten aan onze redacteur.

Redacteur: Nicolaos Starreveld
FNWI, Universiteit van Amsterdam
Postbus 94214
1090 GE Amsterdam
verdediging@nieuwarchief.nl

is not solvable in that time either. In other words, in order to prove a lower bound for problem B, you try to find a polynomial algorithm that maps yes- and no-instances of some hard problem A to yes- and no-instances of problem B respectively.

The classic complexity theoretic assumption is that NP-hard problems are not solvable in polynomial time or, in other words, that $P \neq NP$. If a proper reduction is found, people say that the problem has no polynomial algorithm under the $P \neq NP$ assumption, i.e., there is no algorithm that solves the problem in n^c time for any constant $c > 0$ (where n is the size of our input).

ETH-tight complexity

While $P \neq NP$ is still the most popular hypothesis, it does not allow one to distinguish algorithms with single exponential running times, e.g. $2^{\Theta(n)}$, from those with $2^{o(n)}$ running times; the latter is usually called *sub-exponential time*. There is a huge difference between $2^{\Theta(n)}$ running time and, say, $2^{o(\sqrt{n})}$ running time. It is therefore interesting to be able to distinguish between problems that admit the latter type of running time, and problems that (most likely) do not. For this reason, stronger hypotheses than $P \neq NP$ were developed. One of the most well-known ones is the Exponential Time Hypothesis which is formulated in terms of the 3-Satisfiability problem, see the box below for more details. We say that an algorithm is ETH-tight if the algorithm runs in $2^{f(n)}$ time, and having an algorithm with running time $2^{o(f(n))}$ would contradict ETH.

A question that arises is the following, how does one prove that a problem cannot be solved in $2^{o(f(n))}$ time, assuming ETH? This is done, as said above, via reductions. The difference between classic reductions for proving NP-hardness and those needed for ETH-tight lower bounds is that the created instance of problem B should be sufficiently small. This is why traditional techniques developed for proving NP-hardness cannot be used in this context, new techniques for proving ETH-tight lower bounds are needed.

In his thesis Sándor developed a general framework to develop sub-exponential algorithms for certain types of geometric intersection graphs. The algorithmic framework uses the paradigm of divide and conquer. In this paradigm, one tries to find a so-called separator in a graph. The separator is a relatively small subset of vertices, which when taken away, divides the graph into two smaller components that have no edges going between them, essentially creating two smaller independent instances of the original problem. One can then consider all options of how the solution interacts with the separator (since the separator is small, this can be done efficiently), and for each such choice solve the problems on the two sides of the separator recursively. While the divide and conquer paradigm has been around for a long time, the thesis proves stronger separator theorems, which is the key ingredient required for the faster algorithms. In addition, the framework can be applied to more graph problems when compared to the existing techniques.

3-Satisfiability problem

The goal is to decide if there is a satisfying assignment to a 3-CNF formula on n variables. Such a formula is in conjunctive normal form and has clauses of size three. In other words, its basic building blocks are n variables and their negations, and it is a disjunction of size three conjunctions. The following expression is an example of a 3-CNF-formula:

$$(x_1 \vee \neg x_2 \vee x_3) \wedge (x_2 \vee \neg x_4 \vee x_5) \wedge (\neg x_1 \vee \neg x_3 \vee x_4).$$

Currently, the best worst-case algorithm that solves 3-Satisfiability runs in $1.3071^n = 2^{\Theta(n)}$ time. Note that simply trying all the 2^n ways in which variables can be set to true and false already yields an algorithm with running time $2^{O(n)}$. Not succeeding in finding a faster algorithm for this problem, i.e., no $2^{o(n)}$ algorithm, for so many years led R. Impagliazzo and R. Paturi in 2001 to state the Exponential Time Hypothesis (ETH). It states that there is a constant $c > 0$ such that there is no 2^{cn} algorithm for 3-SAT.

Euclidean Travelling Salesman Problem

In 1972 it was shown that the TSP problem is NP-hard. A brute-force algorithm for TSP runs in $O(n!)$, but the celebrated Held–Karp dynamic-programming algorithm, discovered independently by Held and Karp and Bellman, runs in $O(2^n n^2)$ time. Despite extensive efforts and progress on special cases, it is still open if an exact algorithm for TSP exists with running time $O((2-\epsilon)^n \mathrm{poly}(n))$.

The geometric counterpart of this problem has been studied extensively and it can be considered one of the most important geometric optimization problems. In the early nineties, algorithms with $n^{\sqrt{n}}$ running time were presented for Euclidean TSP in the planar case, and some years later an algorithm with $n^{O(n^{1-1/d})}$ running time was presented for any $d \geq 2$. Despite significant interest in sub-exponential exact algorithms over the past decade, there has been no progress on Euclidean TSP, except for a lower bound stating that the problem admits no $2^{O(n^{1-1/d-\epsilon})}$ algorithm unless ETH fails.

Sándor succeeded in settling the complexity of Euclidean TSP, up to constant factors in the exponent. He constructed an algorithm for Euclidean TSP in $\mathbb{R}^d$, where $d > 2$ is a fixed constant, with running time $2^{O(1-1/d)}$, and he showed that no $2^{o(1-1/d)}$ algorithm exists unless ETH fails.

The more personal aspect

Behind all dissertations there is always a person, with flesh and bones, who has endured the long path of a PhD trajectory and has produced the work at hand.

Were you also involved in some other activities and events during your PhD?

"My PhD was done in the Networks program, which organized several events throughout the years. The training weeks were an especially good experience for me and I imagine for many others as well. It was a loose community where hard work was encouraged, but at the same time the atmosphere was not too competitive, and it was easy to socialize with like-minded peers. Those weeks were also a good opportunity to break out of my research bubble a little bit."

What are your plans for the future?

"I plan to stay in academia. I really enjoy research, and pushing the limits of our understanding. Teaching is also rewarding; I would know next to nothing without having had the great teachers and colleagues I had the pleasure to learn from. So the short-term goal is to get a position more permanent than that of a postdoc, preferably in Europe. And to perform the job hunt while maintaining the current momentum in several solo and collaborative research projects."