VOLUME 1 (HTTPS://BHAVANA.ORG.IN/CATEGORY/VOLUME-1/) ISSUE 2 (HTTPS://BHAVANA.ORG.IN/CATEGORY/ISSUE-2/) APRIL 2017 (HTTPS://BHAVANA.ORG.IN/CATEGORY/APRIL-2017/)

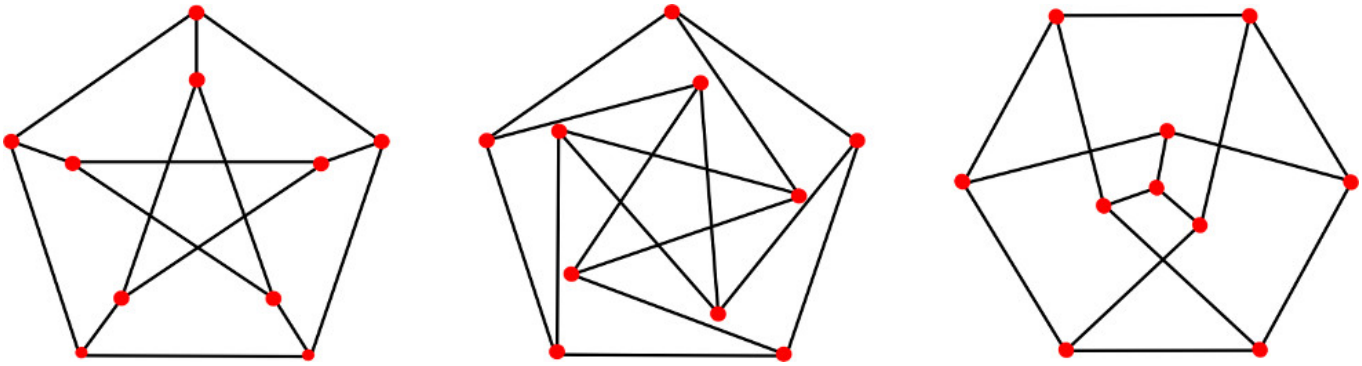
EXPOSITORY (HTTPS://BHAVANA.ORG.IN/CATEGORY/EXPOSITORY/)

Two Steps Forward, One Step Back

The Ongoing Journey of László Babai's Proof of the Graph Isomorphism Problem

by SOURAV CHAKRABORTY

(https://bhavana.org.in/babais-proof-graph-isomorphism/)



In November 2015, news spread across the academic community that Prof. László Babai had designed a “quasi-polynomial time” algorithm for what is known as the graph isomorphism problem. This was no ordinary news, for the graph isomorphism problem is one of the central problems in theoretical computer science, at the intersection of mathematics and computing. If the news was true, it meant Babai had devised an algorithm for this problem which was significantly faster than the previous best algorithm that had been developed more than three decades ago, in part by Babai himself. There has been almost no significant progress since then, so the prospect of a faster algorithm was very exciting to me as a researcher in this field. To add to the excitement, Babai—popularly addressed by his nickname, Laci—was my Ph.D. advisor.

Graph isomorphism is a simple problem to state. In combinatorics, the word “graph” is used to refer to an object that represents binary relations between elements of a set. These elements are usually represented as nodes, and the binary relation is represented by an edge that connects two nodes if the corresponding elements are related. Thus, a graph can be represented as a network of nodes, with edges connecting some

of the nodes. For example, the nodes can represent Facebook users, and an edge connects two nodes if the two Facebook users are friends. Such a graph is called a friendship graph. Graphs are used extensively to model different types of relations and processes arising in almost all fields of research, including physics, biology, engineering and computer science. Good algorithms on graphs are behind the performance of many gadgets and smartphone apps. For example, communication networks, such as the road network in a city, can be represented using graphs. When one uses software to find the shortest/quickest path between two points in a city, one is using an algorithm for “finding the shortest path” on the graph that represents the road network. Thus the study of graphs, called graph theory, and graph algorithms, are central areas of research.

The graph isomorphism question simply asks if two given graphs are really the same graph in disguise—that is, is there a one-to-one mapping (an *isomorphism*) between their nodes, such that an edge connects two nodes in the first graph if and only if the two corresponding nodes in the second graph are also connected by an edge. If such a one-to-one mapping exists, then the two graphs are called isomorphic. Though the problem is quite easy to state, actually checking if two graphs are isomorphic can be quite tricky. For example, the three graphs in the picture on the previous page are all isomorphic though they look very different at first glance.

Graph isomorphism algorithms are used in many fields of research. In chemical informatics, graph isomorphism algorithms are used to identify a chemical compound within a chemical database. Graph isomorphism is also the backbone of the “Layout Versus Schematic” circuit design step in the subject of electronic design automation. For these applications, one needs an algorithm that can quickly check if two graphs are isomorphic. In theoretical computer science, the speed of an algorithm, that is, the number of steps needed by the algorithm, is measured as a function of the input size. For the graph isomorphism problem, the input size is the number of nodes of the graph. Thus, if the number of nodes is n, then the speed or running time of the algorithm is the number of steps, as a function of n, needed by the algorithm to check if the graphs are isomorphic.

There are two ways to measure the speed of an algorithm that checks if two graphs are isomorphic. The worst-case running time of such an algorithm is the maximum number of steps it needs, where the maximum is over all possible pairs of input graphs on n vertices. That is, if the worst-case running time of the algorithm is, say, n^2 it means that for any pair of graphs on n vertices the algorithm makes at most n^2 steps for checking if the two graphs are isomorphic. This is different from the algorithm having an “average-case running time” of n^2, which would mean that it takes less than n^2 steps for most pairs of graphs on n vertices but for some rare cases it may make way more number of steps. One can also come up with some algorithm which would answer correctly for most of the inputs but fails for some rare cases. The graph

isomorphism problem is to design a *foolproof* algorithm ("foolproof" means that for any pair of graphs, the algorithm must output the correct answer) for checking if two given graphs are isomorphism while reducing the worst-case running time as much as possible.

Here is a simple algorithm to check if two graphs G and H with n nodes each are isomorphic: Go over all the possible $n!$ one-to-one correspondences between the nodes of the two graphs, and check if the graphs are isomorphic for any of the correspondences. That is:

1. Let the vertices in graph G be labeled $u_1, \ldots, u_n$ and the vertices in graph H be labeled $v_1, \ldots, v_n$.
2. Now let π be a permutation of $1, \ldots, n$, that is, π sends i to $\pi(i)$.
3. Then let us consider the correspondence that takes u_i to $u_{\pi(i)}$. Now to check if this correspondence gives an isomorphism, check if the following holds: for all vertices u_i and u_j in G there is an edge between u_i and u_j if and only if there is an edge between $v_{\pi(i)}$ and $v_{\pi(j)}$.

The algorithm goes over all possible $n!$ permutations π on $\{1, \ldots, n\}$ and checks if the underlying correspondence between the vertices gives an isomorphism or not. This algorithm, called an exhaustive search algorithm, is a correct algorithm, but since it has to go over all the possible permutations its speed is $O(2^n)$. In the language of computer science, this is an exponential time algorithm because the number of steps the algorithm needs to run is an exponential in n. If the number of steps needed by an algorithm is a polynomial in n (say n^8), then it is called a polynomial time algorithm. The $O(.)$ in the $O(2^n)$ is called the Big-oh notation. For any function $f(n)$ the notation $O(f(n))$ stands for a function that is upper-bounded by $cf(n)$, with c being some constant factor. So for an algorithm running in time $O(2^n)$, if the number of nodes in the input graph increases by 1 the number of steps taken by the algorithm doubles. This makes the algorithm really slow and unusable in practice. The main challenge of the graph isomorphism problem is to design algorithms that are significantly faster than the exhaustive search algorithm.

Consider another simple algorithm:

1. Measure the degree of each node—that is, the number of edges attached to it. Let $g_0, \ldots, g_{n-1}$ and $h_0, \ldots, h_{n-1}$ be the *degree sequences* of G and H respectively, where g_i and h_i are the number of vertices with degree i in G and H respectively.
2. Note that if G and H have different degree sequences, then they cannot be isomorphic. This can be a simple way to check if the two graphs are isomorphic.

One can argue that "most" pairs of non-isomorphic graphs will have different degree sequences, and hence will be caught by the algorithm. Although the number of steps needed by the algorithm is only $O(n^2)$, the algorithm is not a foolproof algorithm as there are many non-isomorphic graphs that have the same degree sequences. In contrast, the exhaustive search algorithm is a foolproof algorithm.

One can come up with more sophisticated algorithms for the graph isomorphism problem which would give the right answer "most" of the time, but in some extreme cases the algorithm would fail. These kind of algorithms are "good in practice" and in fact, we have extremely fast algorithms that are of this kind. But the main challenge from a theoretical computer scientist's point of view is to come up with a *fast* algorithm that gives the correct answer for *all* inputs. Sometimes unfortunately the algorithm that correctly outputs on all inputs is not as fast or lacks other properties that another algorithm which is "good in practice" might have—but that's the trade-off between theory and practice.

In general, one could design other types of algorithms for a problem, such as a randomized algorithm, where the answer calculated is correct with high probability, or approximation algorithms, where the answer calculated may be an approximation to the actual answer. However, these approaches work mainly for optimization problems, where the goal of the algorithm is to optimize a quantity, under certain constraints. The graph isomorphism problem, on the other hand, is not an optimization problem but a decision problem, where the answer is either Yes or No. Randomized and approximate algorithms have been well-studied, both in terms of designing better algorithms, and in terms of proving the limitations of these algorithms. (In 2014, Subhash Khot was awarded the Nevanlinna Prize, a prestigious prize in theoretical computer science, for his work on "Hardness of Approximation".)

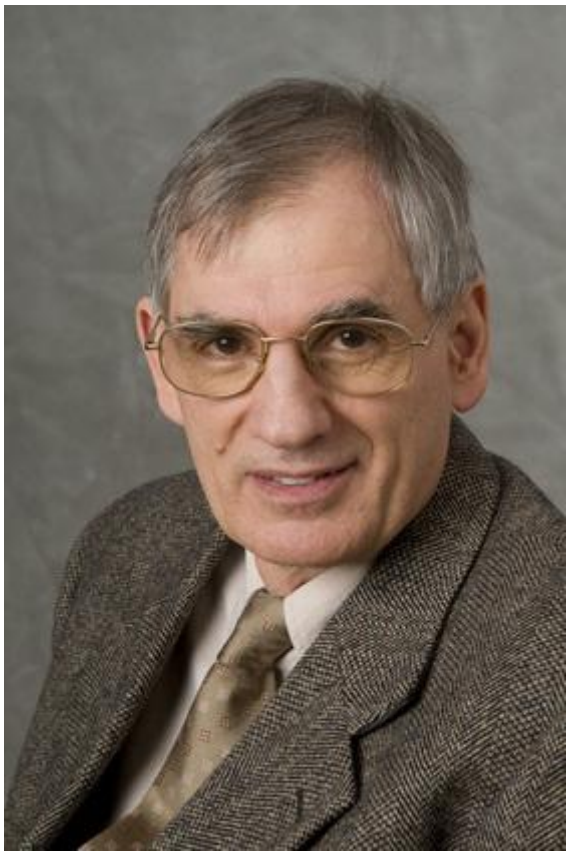
For many problems, smarter randomized or approximation algorithms have been found. For example, for checking whether a natural number p is a prime number we had a randomized algorithm by Gary L. Miller and Michael O. Rabin which ran in polynomial in $\log p$ number of steps much before Manindra Agrawal, Neeraj Kayal and Nitin Saxena's 2002 seminal paper where they designed a deterministic algorithm (that is, without using randomization) that ran in polynomial in $\log p$ number of steps. Still the randomized algorithm of Miller and Rabin is faster. Similarly, for the problem of finding the smallest *vertex cover* in a graph (that is, the smallest subset of the vertex set such that all other vertices in the graph have an edge to at least one of the vertices in the subset) there is a simple polynomial time algorithm that approximates the size of the smallest vertex cover upto a multiplicative factor of 2. That is, if the size of the smallest vertex cover is k the algorithm will output a vertex cover of size between k and $2k$. But no polynomial time algorithm is known for calculating the size of the minimum vertex cover and possibly does not exist. Unfortunately, for the graph isomorphism and related problems, randomization and approximation approaches are not known to help in designing faster algorithms.

The graph isomorphism problem is one of a few super-star problems in theoretical computer science. Almost every year, somebody or the other claims to have solved, or made significant progress in solving, one of these super-star problems. Almost every such attempt either turns out to have a serious flaw or is, at best, a small step towards a solution. Sometimes, the people who claim to solve these problems are not experts in the field, and hence make mistakes like

misunderstanding definitions. But sometimes, false claims are made by experienced researchers. In a complicated mathematical proof, an error that has crept in can be really hard to catch. It may take years before the flaw is realized. Rarely do we see truly ground-breaking work that makes significant progress on one of these star problems. Thus, when news spreads that someone has solved or made significant progress on one of these star problems, researchers in the community tend to be sceptical, or even ignore such news.

But that was not the case when the news spread that László Babai had made significant progress on the graph isomorphism problem. Because of the lack of progress over the last three decades, not many people were expecting this result to appear anytime soon. But nobody could ignore the news, as Babai is a stalwart in the field of combinatorics and theoretical computer science. He is a professor of Mathematics and Computer Science at the University of Chicago, and has been conferred with various prestigious awards including the Hungarian State Prize in 1990, the Gödel Prize in 1993 and the Knuth Prize in 2015. His main area of research is in algorithms, combinatorics and complexity. In fact, in the context of the graph isomorphism problem, the last significant result was also proved by him in a joint paper with Eugene M. Luks [1], way back in 1983, where they gave an algorithm that took $n^{O(\sqrt{n})}$ number of steps in the worst case. Thus, if ever there was an expert in this area, it would be Laci.

In November 2015, Laci submitted his paper to a very prestigious conference [2] and in December 2015, the paper was made available for the public [3]. The paper was around 80 pages long, and Laci gave a series of talks at different places explaining the nitty-gritty of his work. The news had drawn so much attention in the media that almost all his talks were house-full, with more following the talks online. I was fortunate enough to attend a series of talks by Laci when he explained his work to a group of experts at a workshop in Germany in December 2015. One thing that everybody agreed on was that the paper was seriously involved. Personally, I could only get a high-level picture of the complete proof, and was able to appreciate some of the beautiful subparts/lemmas in the paper. But I could not follow a major chunk of the proof—it was too complicated for me. I was not alone, however. Nobody could follow the whole of the paper. The situation was aptly described by Luks—one of the stalwarts in group algorithms (algorithms in group theory which is an integral part of the technique used in Laci's algorithm for the graph isomorphism problem) and Laci's collaborator for the previous best algorithm—during a chat over coffee, where he expressed doubts as to how many people in the world could properly review this paper. He did not even include himself in that list.



László Babai

COURTESY University of Chicago

The paper was finally accepted in the conference, and won the best paper award. In his proof, Laci has heavily relied upon three areas of mathematics—group theory, combinatorics and algorithms. Recall the algorithm using the degree sequence that we discussed earlier. Although there are many pairs of graphs for which the algorithm fails, the same trick can reveal a lot about the plausible one-to-one mapping between the nodes of the two graphs in question. A simple observation is that if two graphs G and H are indeed isomorphic, then the one-to-one mapping between the vertex sets of the two graphs must ensure that the vertices with degree d in G are mapped to vertices with degree d in H, for any degree d. Thus, by looking at the degrees of the vertices, one can reduce the number of one-to-one mappings that one may have to check. This new plausible set of one-to-one mappings between the vertex sets forms a subgroup of the permutation group S_n. The main idea of Laci's algorithm is to find *canonical labelings* (like the degree of vertices) to narrow the size of the subgroup of plausible one-to-one mappings, and then check if one of the remaining mappings gives an isomorphism between G and H. However, one would not like to use an exhaustive search algorithm over the remaining plausible set of mappings. Instead, one would like to use the structure of the group to narrow down this plausible set of mappings even further. The heart of Laci's paper is using the structure of permutation groups to break the subgroup of plausible mappings into even smaller bite-sized pieces. Laci's algorithm is a recursive algorithm that heavily uses various results from group theory and combinatorics. Many of these results had to be developed in the paper itself.

Laci's new algorithm runs in time $2^{O(\log n)^c}$ for some constant $c > 1$. This is a significant improvement of the previous fastest algorithm which took $n^{O(\sqrt{n})}$ number of steps. While $n^{O(\sqrt{n})}$ is exponential in n for large n, $2^{O(\log n)^c}$ is much smaller than any exponential function of n, but bigger than any polynomial in n. In the Big-oh notation, any polynomial can be written as $2^{O(\log n)}$, and so $2^{O(\log n)^c}$ for some constant $c > 1$ is called *quasi-polynomial*. This means that it is slightly bigger than a polynomial. Thus Laci's algorithm is a quasi-polynomial time algorithm. The exact c has not been calculated. Also there are a few different ways of calculating the number of steps taken by the algorithm, depending on whether one uses the classification of finite simple groups (CFSG). The CFSG is one of the great results in the field of finite group theory. Its proof is spread over decades of research and hundreds of papers, with a total of over 15,000 pages when put together. So naturally there are some mathematicians who refuse to accept the correctness of CFSG.

Laci said that he has been working on this problem for the past 30 years. In fact, many parts of the paper were solved in the early 1990s. While he had been making small progresses since, the final push came when he encountered an idea in another paper by one of his students in early 2015. He then spent the whole summer of 2015 working towards this result, locked up in his own world, diving as deep as possible in the sea of mathematics. By the end of the summer he had his result.

I returned from the workshop deeply in love with the paper. From whatever I could follow, I was mesmerized. The interplay between the three sub-areas was truly exceptional. And the subparts, the lemmas and the theorems were gems. Although I did not follow all of it, I planned that one day in near future, I will work towards fully understanding the paper.

But then came a shock. Over one year after his paper came out, Laci made a public statement, on 4 January 2017, that there was an error in the proof. Prof. Harald Helfgott of University of Gottingen had pointed out an error because of which the algorithm runs not in "quasi-polynomial time", but in *sub-exponential time*. That is, in this revised analysis, the number of steps made by the algorithm is much more than any quasi-polynomial in n, but smaller than any exponential function in n. The new algorithm was still significantly faster than the previous best algorithm, but it is also significantly slower than what was initially claimed. Soon after, on 9 January 2017, Laci modified his proof, and said that the modified proof could achieve the same result as was first claimed. In mid-January, the fix for the flaw was written up and posted online [4] . And that's where it stands now while experts scrutinize the paper.

The story of this paper of Laci beautifully summarizes the story of research. Decades of hard work trying to solve a single problem. A flash of light from here and there, and finally, with enormous amount of hard work, concentration and ingenuity, a great result is produced. Beautiful, yet so complicated, that even the best of experts cannot fully understand easily. Even after all this hard work, and in spite of thorough scrutiny, a small flaw in the paper escapes

everyone's eye for over a year. Finally, a fix was found and hopefully we have a gem of a result without any more hiccups. Such results are almost never the end of a journey. They usually start a new sub-area of research. And yet one may have to wait for another 30 years or more for the next significant progress towards solving the graph isomorphism problem.

The real question one would like to answer is: Does there exist an algorithm for the graph isomorphism problem with speed polynomial in n? In the language of complexity theory, the question then is: Is the graph isomorphism problem in P? P stands for the class of problems for which there is an algorithm that has speed polynomial in the input size. Answering this question in the affirmative would indeed be a spectacular result. Some people believe that after the latest result of Laci it is just a matter of time before we put graph isomorphism in P, while many others, including Laci himself, believe that there are significant stumbling blocks on the way towards putting the graph isomorphism problem in P. But an even more spectacular result would be if one can prove that the graph isomorphism problem in not in P, because that would settle the famous $P \text{ vs } NP$ conjecture. This is another super-star problem, and is one of the seven problems in mathematics that is a Millennium Prize Problem—solving it would earn one a million dollars from the Clay Mathematics Institute. Vaguely speaking, the $P \text{ vs } NP$ conjecture asks if solving a problem is as hard as checking if a proposed solution to it is correct (NP is the class of problems for which a solution can be checked in polynomial time). If $P = NP$ then then the world as we know it could face troubles—for example, all our digital security starting from encryption for our credit card transactions to the secret communications used by the military could be breached. However, answering the $P \text{ vs } NP$ conjecture in the positive or the negative is very unlikely anytime in the near future. The graph isomorphism problem is in NP, as one can easily verify if two graphs are isomorphic once the one-to-one mapping between their vertex sets is given. It should be noted that even if someone proves that the graph isomorphism problem is in P, it does not imply that $P = NP$. So, if it is indeed the case that the graph isomorphism problem in not in P, that is, if the speed of any foolproof algorithm for the graph isomorphism problem, its speed cannot be upper bounded by a polynomial in the input size, then chances that we prove it in the near future seem extremely slim.

References

[1] L. Babai and Eugene M. Luks. Canonical Labeling of Graphs. *Proceedings STOC '83*. 1983. pp. 171–183, DOI: 10.1145/800061.808746 (https://doi.org/10.1145/800061.808746)

[2] L. Babai. Graph Isomorphism in Quasipolynomial Time. *STOC '16 Proceedings of the Forty-Eighth Annual ACM Symposium on Theory of Computing*. 2016. pp. 684–697. DOI: 10.1145/2897518.2897542 (https://doi.org/10.1145/2897518.2897542)

[3] L. Babai. Graph Isomorphism in Quasipolynomial Time. 2016. Online; accessed 28 March 2017. (https://arxiv.org/abs/1512.03547)

[4] L. Babai. Fixing the UPCC case of Split-or-Johnson. 2017. Online; accessed 28 March 2017. (http://people.cs.uchicago.edu/~laci/upcc-fix.pdf)

Sourav Chakraborty is an Associate Professor at the Chennai Mathematical Institute (CMI), Chennai. He did his Bachelor's in mathematics and computer science at CMI and went on to obtain his Master's and Ph.D. from the University of Chicago under the supervision of László Babai. His research is in theoretical computer science, particularly complexity theory and algorithms.

JANUARY 2017 COVER (HTTPS://BHAVANA.ORG.IN/JANUARY-2017-COVER/)
GLIMPSES OF P.L. BHATNAGAR (HTTPS://BHAVANA.ORG.IN/GLIMPSES-OF-PL-BHATNAGAR/)