GRAPH ISOMORPHISM ALGORITHM BY PERFECT MATCHING

Kazuma Fukuda
Department of Internet Media Sysytem
Information Technology R&D Center
MITSUBISHI ELECTRIC Corporation
Kamakura, Kanagawa, JAPAN
kfukuda@isl.melco.co.jp

Mario Nakamori
Department of Computer Science
Tokyo A&T University
Koganei, Tokyo, JAPAN
nakamori@cc.tuat.ac.jp

Abstract No polynomial time algorithm is known for the graph isomorphism problem. In this paper, we determine graph isomorphism with the help of perfect matching algorithm, to limit the range of search of 1 to 1 correspondences between the two graphs: We reconfigure the graphs into layered graphs, labeling vertices by partitioning the set of vertices by degrees. We prepare a correspondence table by means of whether labels on 2 layered graphs match or not. Using that table, we seek a 1 to 1 correspondence between the two graphs. By limiting the search for 1 to 1 correspondences between the two graphs to information in the table, we are able to determine graph isomorphism more efficiently than by other known algorithms. The algorithm was timed with on experimental data and we obtained a complextity of $O(n^4)$.

Keywords: Graph Isomorphism, Regular Graph

1. Introduction

The graph isomorphism problem is to determine whether two given graphs are isomorphic or not. It is not known whether the problem belongs to the class P or the class NP–complete. It has been shown,

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-0-387-35699-0_19
E. W. Sachs et al. (eds.), *System Modeling and Optimization XX*

however, that the problem can be reduced to a group theory problem (van Leeuwen, 1990).

Most studies of graph isomorphism (Hopcroft and Wong, 1974; Lueker, 1979; Babai et al., 1980; Galil et al., 1987; Hirata and Inagaki, 1988; Akutsu, 1988) restrict graphs by their characteristics. Some studies are undertaken based on group theory. Most studies are concerned on the existence of algorithms (Filotti and Mayer, 1980; Babai et al., 1982; Luks, 1982; Babai and Luks, 1983; Agrawal and Arvind, 1996), and a few papers report the implementation of algorithms (Corneil and Gotlieb, 1970) and experimental results.

At present the best computational complexity by worst case analysis (Babai and Luks, 1983; Kreher and Stinson, 1998) is $O\left(c^{n^{1/2+o(1)}}\right)$. This algorithm makes use of the unique certification of a graph.

In the present paper, we consider the graph isomorphism problem for non-oriented connected regular graphs whose vertices and edges have no weight. We seek graph isomorphism by means of perfect matching to limit the range of 1-to-1 correspondences between the two graphs as follows.

First, we choose one vertex as root for each graph and reconfigure the graphs into layered graphs corresponding to the chosen vertices. Next, we label those vertices by partitioning the set of vertices by the distance from the root vertex. We construct a correspondence table which reflects whether labels on 2 layered graphs are the same or not. Then, referring to that table, we search for a 1-to-1 correspondence between the two graphs.

In other words, we create a bipartite graph between V_1 and V_2 and find a perfect matching in this bipartite graph.

In the worst case, we might enumerate all the combinations of vertices among the two graphs, which would be of exponential order. However, we have been successful in determining the isomorphism of graphs within a reasonable time using experimental data; these results are also reported in the present paper.

We consider only regular graphs. Since the general graph isomorphism problem can be reduced to the regular graph isomorphism problem in polynomial time (Booth, 1978), this restriction does not lose generality.

1.1. Perfect Matching Problem

The matching problem on a bipartite graph is a problem that of finding a set of edges such that any two edges do not share the same vertex (Iri, 1969). If the set covers all the vertices, the set is called *perfect matching.*

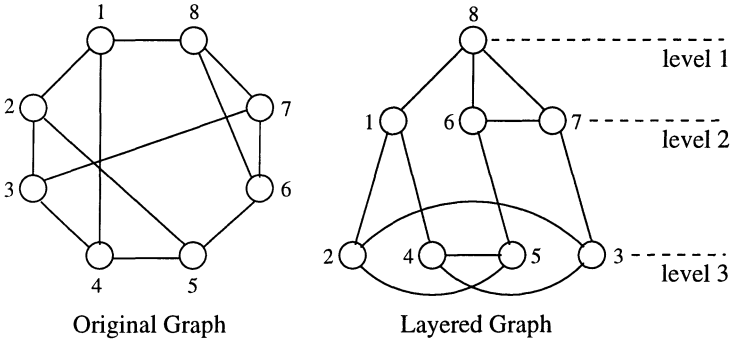

Figure 1. Layered Graph

It is known that there exist polynomial algorithms of finding a perfect matching. (Micali and Vazirani, 1980 etc.)

1.2. Preliminaries

Let the two given regular graphs be $G_1 = (V_1, E_1)$, $G_2 = (V_2, E_2)$, where $|V_1| = |V_2| = |V| = n$, $|E_1| = |E_2| = |E|$ $(= O(n^2))$. Each vertex is uniquely labeled and is stored in an array of size n. Graph isomorphism is defined as follows.

Definition 1 *Two graphs* $G_1 = (V_1, E_1)$ *and* $G_2 = (V_2, E_2)$ *are isomorphic, if there is a 1-to-1 correspondence* $f : V_1 \to V_2$, *such that* $(v, v') \in E_1$ *iff* $(f(v), f(v')) \in E_2$ *for any* $(v, v') \in E_1$. *This function* f *is called an isomorphism between* G_1 *and* G_2.

Similarly we could define graph isomorphism in the case where one vertex is fixed in each graph.

We consider only regular graphs for which the vertex degree satisfies $3 \leq d \leq \lfloor \frac{n-1}{2} \rfloor$, because of the relation between a graph and its complement.

2. Reconfiguring Graphs to Layered Graphs

In the present paper, we make use of layered graphs to determine isomorphism.

2.1. Layered Graphs

Given a graph G and a vertex $r \in V$, the layered graph $L(G, r)$ with root r consists of

- vertices of G,

- edges of G,
- $level(u)$ for each vertex u,

where $level(u)$ is the shortest distance (or the depth) from r to u (Figure 1). Transforming a graph with n vertices to a layered graph can be done in $O(n^2)$ time.

2.2. Characteristics of Layered Graphs

We divide the set of vertices adjacent to v into 3 subsets, $D_u(v)$, $D_s(v)$, and $D_d(v)$, as follows:

- $D_u(v) = \{v' \mid (v, v') \in E \text{ and } level(v') = level(v) - 1\}$,
- $D_s(v) = \{v' \mid (v, v') \in E \text{ and } level(v') = level(v)\}$,
- $D_d(v) = \{v' \mid (v, v') \in E \text{ and } level(v') = level(v) + 1\}$.

Let the number of vertices of each subset be d_u , d_s, and d_d :

- $d_u(v) = |D_u(v)|$, (upper degree)
- $d_s(v) = |D_s(v)|$, (same level degree)
- $d_d(v) = |D_d(v)|$. (lower degree)

It follows that the degree of v, $d(v)$, is equal to $d_u(v) + d_s(v) + d_d(v)$.

It is trivial to derive at the following:

- $d_u(r) = d_s(r) = 0$, $d_d(r) = d(r)$,
- each vertex v except the root vertex satisfies $d_u(v) \geq 1$,
- all vertices adjacent to the vertices in *level* i have *level* i or $(i \pm 1)$.

Given these assumptions, we propose the following.

Proposition 1 *Two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ are isomorphic if and only if there are vertices $v_1 (\in V_1)$ and $v_2 (\in V_2)$ and the two layered graphs $L(G_1, v_1)$ and $L(G_2, v_2)$ are isomorphic.*

Each vertex $v (\in V)$ has a label[1] ($level(v)$, $d_u(v)$, $d_s(v)$, $d_d(v)$). Let the label be denoted by $M(v)$. We call the set of vertices that have the same labels a "class," which we denote by B_i $(1 \leq i \leq k$, where k is the number of classes). For example, data from Figure 1 are shown in Table 1 sorted by label. We denote by $\mathcal{L}(G, v)$ the vertices of G partitioned into classes.

[1] A label for a general vertex is constructed by graph appending each vertix's degree $d(v)$ to the level.

Table 1. Example of Labeling. Data are from graph shown in Figure 1

	8	1	6	7	2	4	5	3
level	1	2	2	2	3	3	3	3
$d(v)$	3	3	3	3	3	3	3	3
$d_u(v)$	0	1	1	1	1	1	1	1
$d_s(v)$	0	0	1	1	2	2	2	2
$d_d(v)$	3	2	1	1	0	0	0	0
class	1	2	3	3	4	4	4	4

3. Finding a 1-to-1 correspondence between two graphs

In this section, we consider how to make use of perfect matching algorithm in order to determine the isomorphism of graphs.

3.1. Correspondence between 2 Layered Graphs

For two given graphs, we consider all layered graphs for which a vertex of the graph is the root.

For $v_i \in V_1$ and $v_j \in V_2$, we set $c_{ij} = 1$ if $\mathcal{L}(G_1, v_i)$ and $\mathcal{L}(G_2, v_j)$ have the same labels and partitions, otherwise $c_{ij} = 0$. Thus, we have a correspondence table as shown in Table 2.

It is easy to see that each table entry's value is unique and does not depend on expressions of the two graphs.

Table 2. Table of Layered Graphs

		G_1						
		1	2	⋯	i	⋯	$n-1$	n
	1	0	1	⋯	0	⋯	1	0
	2	0	1	⋯	0	⋯	1	0
	⋮				⋮			
G_2	j	1	0	⋯	1	⋯	0	1
	⋮				⋮			
	$n-1$	1	0	⋯	1	⋯	0	1
	n	0	0	⋯	0	⋯	1	0

The entries with a value of 1 are candidates for a 1-to-1 correspondence between vertices the two graphs. As a result, we could take that correspondence, by finding perfect matchings according to the table. (Figure 2)

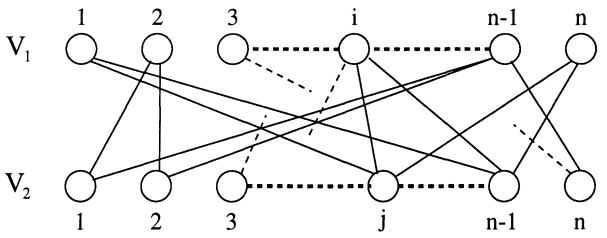

Figure 2. 1-to-1 correspondences as perfect matchings

In the graph isomorphism problem, we have to determine whether there exists a 1-to-1 correspondence between vertices in two graphs checking all possible perfect matchings[2] (in the correspondence table). Of course, the possible perfect matchings do not always indicate isomorphism, so we have to enumerate all perfect matchings and to test for isomorphism. However, the table limits the range searched for a 1-to-1 correpondence.

If there is no perfect matching between two graphs based on this table, they are not isomorphic.

3.2. Solutions and Issues

We have implemented the above algorithm and in Section 4 applied it experimentally to determine isomorphism. We test for 1-to-1 correspondence between vertices in two graphs as follows.

- Construct a 1-to-1 correspondence table as preprocessing.
- Test for 1-to-1 correspondence between vertices in the two graphs.

Next, we enumerate 1-to-1 correspondences one by one until we find a perfect matching between vertices in graphs.

The program based on our algorithm and described in the next section has not adopted stronger methods to bound recursion, because we want to make it easier to understand effectiveness by using a table.

However, if all entries in the table are 1's, we have to enumerate all perfect matchings. This results in many combinations of 1-to-1 correspondence to test. This might be the worst situation for our algorithm. In such situation, however, we could consider 2 cases whether 2 graphs are isomorphic or not.

[2]In practice, it is not necessary to enumerate those perfect matchings to determine isomorphism.

In the former case, since both graphs would have much symmetry, we could find a 1-to-1 correspondence earlier. In the latter case, we do not need to enumerate all perfect matchings as follows :

- Consider the two layered graph which both root vertices are corresponding in the table.
- Within **each corresponding class** between the two layered graphs, test 1-to-1 correspondences.
  - Examine the number of same vertices adjacent to 2 vertices in each corresponding class.
  - If there is no 1-to-1 correspondence for at least one class, they are not isomorphic.

Thus, we could reduce complexity of enumeration.

Also we need to consider what features of graphs indicate the worst complexity.

Among other known algorithms the best complexity in the worst case analysis is of time $O\left(c^{n^{1/2+o(1)}}\right)$ (Babai and Luks, 1983; Kreher and Stinson, 1998). That algorithm determines isomorphism by certifying graphs uniquely. Though it certifies by partitioning the set of vertices recursively, the basic idea in partitioning is as follows : "which partitioned set contains vertices adjacent to a certain vertex?" To prevent unnecessary recursions, it takes advantage of certifications results. The complexity of certification is of exponential order.

4. Experiments

We have implemented the program described above and experimented on various regular graphs.

4.1. Environment and Graph Data

Our experiment was carried out with a Celeron 450MHz, 128 MB memory (and 128 MB swaps) and C (gcc-2.91.66) on Linux (2.2.14). We measured running time using a UNIX like OS command "time."

We have constructed various regular graphs for input using a program that was implemented according to Matsuda et al., 1992. Those graphs have numbers of vertices from 20 to 120 with vertex degree of 10. We constructed not only various isomorphic graphs that have the same number of vertices and degree but also non-isomorphic ones.

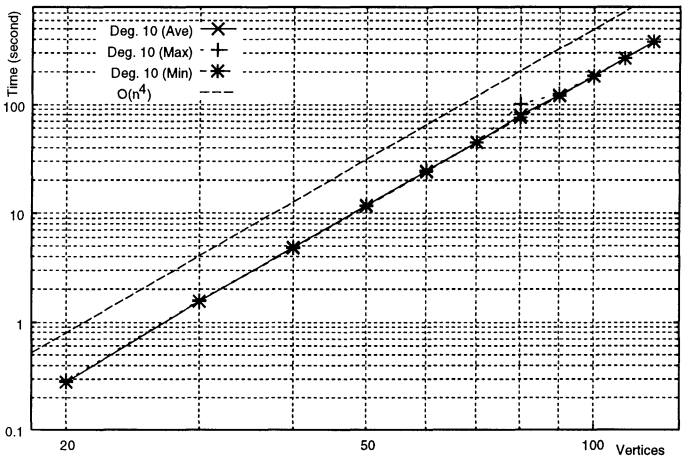

Figure 3. Isomorphic case

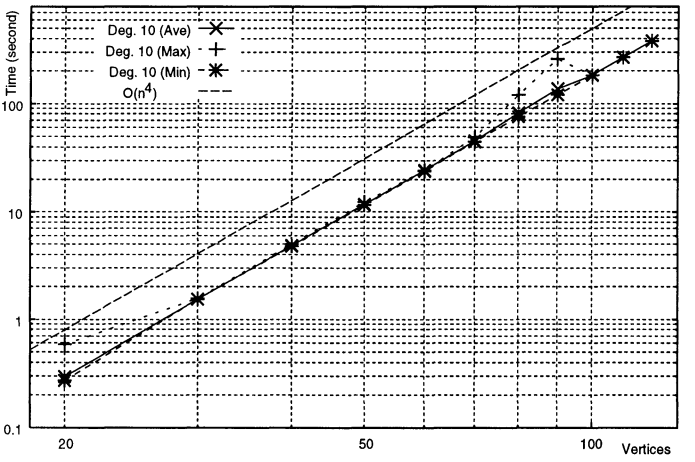

Figure 4. Non-isomorphic case

4.2. Results and Estimation

Computational results are shown in Figures 3 and 4. In these figures, we show the average and the maximum time versus the number of vertices in a graph, and depict the resulting curve[3].

[3]That was multiplied by adequate constants to be easily able to compare.

As a result, we conclude that the experimental time complexity is proportional to $O(n^4)$ regardless of whether the graphs are isomorphic or not. These results tend to be closer to the complexity of making a correspondence table than of examining 1 to 1 correspondences (perfect matchings) between the two graphs. Besides, we have seen almost the same results in both cases isomorphic and non isomorphic.

We anticipated that complexity might be larger as all the perfect matchings might be enumerated in the non-isomorphic case, but the result of our experiment showed to be much more efficient.

Differences between average time, maximum time and minimum time in the number of vertices and degree are very small, so the program is quite stable. Standard deviations in the results are also very small (though not shown here) and didn't have any result over 1 second. Furthermore, in the non-isomorphic case, we could determine lack of isomorphism by testing only the table (in the graphs used at least).

5. Conclusions

In the present paper, targeting nonweighted, undirected and connected regular graphs, we considered graph isomorphism by means of perfect matching to limit the range of 1 to 1 correspondence between two graphs as follows. First, we reconfigured the given graph as a layered graph, labeled vertices by partitioning the set of vertices by distance from a root vertex, and prepared a correspondence table by means of whether labels on 2 layered graphs matched or not. Using that table, we find 1 to 1 correspondences between the two graphs. In our experiments, we could determine isomorphism within a practical and stable time.

For further research, we have to examine other types of graphs, and analyse complexity of the program for them. Also, we wish to compare our results with practical running results of the best algorithm described in Babai and Luks, 1983 and Kreher and Stinson, 1998 whose worst complexity are known to have exponential time.

References

Agrawal, M. and Arvind, V. (1996). A note on decision versus search for graph automorphism. *Information and Computation*, 131:179–189.

Akutsu, T. (1988). A polynomial time algorithm for subgraph isomorphism of tree-like graphs. *IPSJ 90–AL–17–2.*

Babai, L., Erdös, P., and Selkow, S. M. (1980). Random graph isomorphism. *SIAM J. Comput.*, 9:628–635.

Babai, L., Grigoryev, D. Y., and Mount, D. M. (1982). Isomorphism of graphs with bounded eigenvalue multiplicity. *Proc. 14th Annual ACM Symp. Theory of Computing*, pages 310–324.

Babai, L. and Luks, E. M. (1983). Canonical labeling of graphs. *Proc. 14th Annual ACM Symp. on Theory of Computing, Boston*, pages 171–183.

Babel, L., Baumann, S., Ludecke, M., and Tinhofer, G. (1997). Stabcol: Graph isomorphism testing based on the weisfeiler-leman algorithm. Technical Report Preprint TUM-M9702, Munich.

Barrett, J. W. and Morton, K. W. (1984). Approximate symmetrization and Petrov-Galerkin methods for diffusion-convection problems. *Comput. Methods Appl. Mech. Engrg.*, 45:97–122.

Booth, K. S. (1978). Isomorphism testing for graphs, semigroups, and finite automata are polynomially equivalent problems. *SIAM J. Comput.*, 7:273–279.

Corneil, D. G. and Gotlieb, C. C. (1970). An efficient algorithm for graph isomorphism. *J. ACM*, 17:51–64.

Cull, P. and Pandy, R. (1994). Isomorphism and the n-queens problem. *ACM SIGCSE Bulletin*, 26:29–36.

Filotti, I. S. and Mayer, J. N. (1980). A polynomial-time algorithm for determining the isomorphism of graphs of fixed genus. *Proc. 12th Annual ACM Symp. Theory of Computing*, pages 236–243.

Galil, Z., Hoffmann, C. M., Luks, E. M., Schnorr, C. P., and Weber, A. (1987). An $o(n^3 \log n)$ deterministic and an $o(n^3)$ las vegas isomorphism test for trivalent graphs. *J. ACM*, 34:513–531.

Hirata, T. and Inagaki, Y. (1988). Tree pattern matching algorithm. *IPSJ 88-AL-4-1.*

Hopcroft, J. and Wong, J. (1974). Linear time algorithms for isomorphism of planar graphs. *Proc. 6th Annual ACM Symp. Theory of Computing*, pages 172–184.

Iri, M. (1969). *Network Flow, Transportation and Scheduling.* Academic Press.

Köbler, J., Schöning, U., and Torán, J. (1992). Graph isomorphism is low for pp. *J. of Computer Complexity*, 2:301–330.

Kreher, D. L. and Stinson, D. R. (1998). *Combinational Algorighms: Generation, Enumeration and Search.* CRC.

Lueker, G. S. (1979). A linear time algorithm for deciding interval graph isomorphism. *J. ACM*, 26:183–195.

Luks, E. M. (1982). Isomorphism of graphs of bounded valence can be tested in polynomial time. *J. Computer and System Sciences*, 25:42–65.

Matsuda, Y., Enohara, H., Nakano, H., and Horiuchi, S. (1992). An algorithm for generating regular graphs. *IPSJ 92-AL-25-3.*

Micali, S. and Vazirani, V. V. (1980). An $o(\sqrt{V} \cdot e)$ algorithm for finding maximum matching in general graphs. *Proc. 21st Ann. IEEE Symp. Foundations of Computer Science*, pages 17–27.

Torán, J. (2000). On the hardness of graph isomorphism. *Proc. 41st Annual Symposium on Foundations of Computer Science, California*, pages 180–186.

van Leeuwen, J. (1990). *Handbook of Theoretical Computer Science, Vol. A: Algorithm and Complexity.* Elseveir.