グラフ同形性判定アルゴリズムにおける層別グラフ利用による効率性向上について

福田 和真，　中森 眞理雄
三菱電機株式会社，　東京農工大学

グラフ同形性判定問題は一般的なグラフにおいて多項式時間で判定可能なアルゴリズムが知られていない問題である．この論文では，次のように 2 つのグラフの節点の 1 対 1 対応の範囲を限定することによりグラフ同形性判定を行う方法について述べる．まず，グラフを層別グラフに再表現し，次数により節点の集合を分類することによりラベル付けする．次に，それぞれのグラフのすべての層別グラフの組合せにおいて，節点のラベルが一致するかどうかで対応表を構成する．そして，その表を使用して 2 つのグラフの 1 対 1 対応を調査する．表を利用して 2 つのグラフ間の節点における 1 対 1 対応の探索範囲を限定することにより，既存の同形性判定のアルゴリズムにおいてもより効率的になることが期待できる．対応表を利用した同形性判定のためのプログラムを実現し，いくつかの正則グラフで実行したところ，実際的かつ安定した時間で判定された．

Accelarating graph isomorphism algorithm by finer classification of layered graphs

Kazuma Fukuda,　Mario Nakamori
MITSUBISHI ELECTRIC Corporation,　Tokyo A&T University

No polynomial time algorithm is known for the graph isomorphism problem. In this paper, we determine graph isomorphism by limiting the range of 1 to 1 correspondences between two graphs as follows: 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 this method, 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 $O(n^4)$.

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, however, that the problem can be reduced to a group theory problem ([7]).

Most studies of graph isomorphism ([3], etc.) restrict graphs by their characteristics, are concerned on the existence of algorithms ([4, 5], etc.), and only a few papers report the implementation of algorithms ([2]) and experimental results.

At present the best computational complexity by worst case analysis ([4]) 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 limiting 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 distance from the root vertex. We prepare a correspondence table which reflects whether labels on 2 layered graphs match or not.

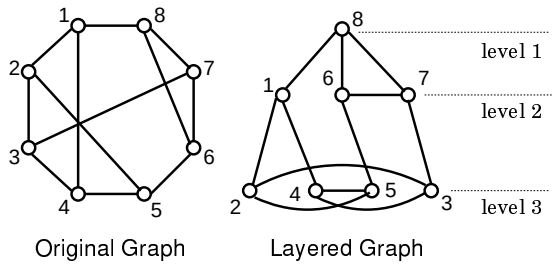

Figure 1: Layered Graph

Then, refering to that table, we search for 1 to 1 correspondences between the two graphs.

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 ([1]), this does not lose generality.

1.1 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.*

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

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

1. vertices on G
2. edges on G
3. $level(u)$ for each vertex u

where $level(u)$ is the shortest distance (or the depth) from r to u (Figure 1). Transforming an n vertex graph 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.

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

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

1. $d_u(v) = |D_u(v)|$ (upper degree)
2. $d_s(v) = |D_s(v)|$ (same level degree)
3. $d_d(v) = |D_d(v)|$ (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.

Proposotion 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

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

Table 1: Example of Labeling. Data's from graph shown in Figure 1

節点	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

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 is shown in Table 1 sorted by label. We denote by $\mathcal{L}(G, v)$ the vertices of G partitioned into classes.

3 Finding a 1 to 1 correspondence between two graphs

In this section, we consider solving graph isomorphism problems by limiting the range of 1 to 1 correspondences between the two graphs making use of layered 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 $v_j \in V_2$, $c_{ij} = 1$ 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.

The entries with a value of 1 are candidates for a 1 to 1 correspondence between vertices the two graphs.

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 correspondences[2] in the correspondence table. Of course, the possible correspondences do not always indicate isomorphism, so we have to enumerate all the 1 to 1 correspondences and test for isomorphism. However, the table limits the range searched for a 1 to 1 correpondence.

[2]In practice, it is not neccesary to enumerate those correspondences to determine isomorphism.

Table 2: Table of Layered Graphs

		G_1						
		1	2	$\cdots$	i	$\cdots$	$n-1$	n
	1	0	1	$\cdots$	0	$\cdots$	1	0
	2	0	1	$\cdots$	0	$\cdots$	1	0
	$\vdots$				$\vdots$			
G_2	j	1	0	$\cdots$	1	$\cdots$	0	1
	$\vdots$				$\vdots$			
	$n-1$	1	0	$\cdots$	1	$\cdots$	0	1
	n	0	0	$\cdots$	0	$\cdots$	1	0

If there is no 1 to 1 correspondence 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 apply it experimentally to determine isomorphism. We test for 1 to 1 correspondence between vertices in two graphs as follows.

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

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

4 Experiments

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

4.1 Environment and Graph Data

Experiments were 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."

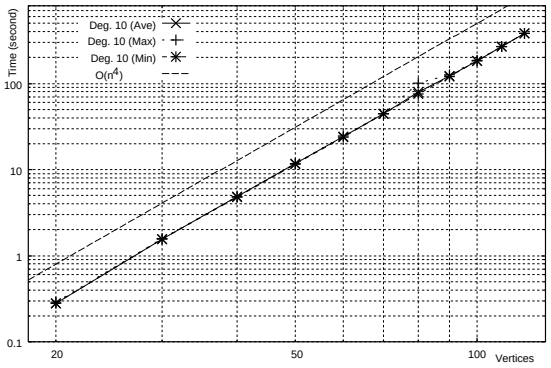

Figure 2: Isomorphic case

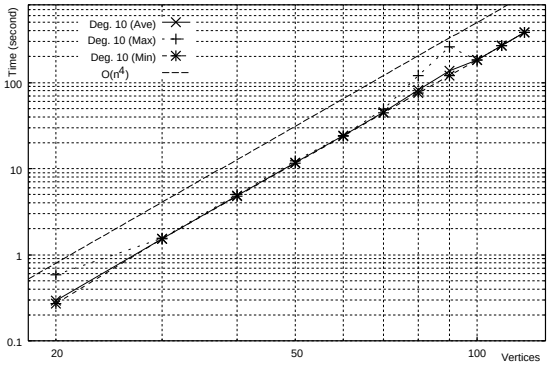

Figure 3: Non-isomorphic case

We have constructed various regular graphs for input using a program that was implemented according to [6]. Those graphs have numbers of vertices from 20 to 120 with vertex degree of 10.

4.2 Results and Estimation

Computational results are shown Figures 2 and 3. 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].

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 between the two graphs.

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 did not 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).

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

5 Conclusions

In the present paper, targetting regular graphs, we considered graph isomorphism by limiting the range of 1 to 1 correspondence between 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 [4] whose worst complexity are known to have exponential time.

References

[1] K. S. Booth, "Isomorphism Testing for Graphs, Semigroups, and Finite Automata are Polynomially Equivalent Problems," SIAM J. Comput., 7, 273–279, 1978

[2] D. G. Corneil, C. C. Gotlieb, "An Efficient Algorithm for Graph Isomorphism," J. ACM, 17, 51–64, 1970

[3] J.E. Hopcroft, J.K. Wong, "Linear Time Algorithms for Isomorphism of Planar Graphs," Proc. 6th Annual ACM Symp. Theory of Computing, 172–184, 1974

[4] L. Babai, E. M. Luks, "Canonical Labeling of Graphs," Proc. 14th Annual ACM Symp. on Theory of Computing, Boston, 171–183, 1983

[5] M. Agrawal, V. Arvind, "A Note on Decision versus Search for Graph Automorphism," Information and Computation, 131, 179–189, 1996

[6] Y. Matsuda, H. Enohara, H. Nakano, S. Horiuchi, "An Algorithm For Generating Regular Graphs", IPSJ 92–AL–25–3, 1992

[7] J. van Leeuwen, "Handbook of Theoretical Computer Science, Vol. A: Algorithm and Complexity," Elseveir, 1990