Marked Subgraph Isomorphism of Ordered Graphs

Xiaoyi Jiang, Horst Bunke

Department of Computer Science, University of Bern
Neubrückstrasse 10, CH-3012 Bern, Switzerland
{jiang,bunke}@iam.unibe.ch

Abstract. Recently, the concept of ordered graphs has been introduced and it was shown that isomorphism of ordered graphs can be solved in quadratic time. In the present paper we consider a special case of the subgraph isomorphism problem for ordered graphs, called marked subgraph isomorphism. An algorithm of $O(m_1 m_2)$ complexity is developed for finding all marked subgraph isomorphisms from a graph G_1 to another graph G_2, where m_1 and m_2 are the number of edges in G_1 and G_2, respectively. We demonstrate the usefulness of our algorithm by applying it to solving the subcircuit extraction problem. It turns out that our approach is much more efficient than traditional methods based on general subgraph isomorphism techniques.

1 Introduction

Graphs are a very powerful and universal tool for epresenting various aspects of the real world. Depending on the requirements of a concrete application, there are numerous methods for the analysis of graphs, such as finding the shortest path, detecting Hamiltonian cycles, coloring the edges of graphs, and many more. A very important problem is the detection of subgraph isomorphism.

Many problems in graph theory are NP-complete, including the subgraph isomorphism problem. By imposing certain restrictions on the underlying graphs, however, it is often possible to derive algorithms of polynomial-time complexity. Examples of such restricted graphs are graphs with bounded valence [9], planar graphs [3], and rooted ordered trees [1]. Alternatively, we may also search for approximate solutions [8]. Recently, the authors [6] have investigated the class of so-called ordered graphs, in which the vertices incident to a vertex have a unique order. Graph isomorphism of ordered graphs can be solved in quadratic time [7] and has found applications in the detection of object symmetries [4, 5] and computer vision [6].

In the present paper we consider a special case of the subgraph isomorphism problem for ordered graphs, called marked subgraph isomorphism. In this case we know *a priori* that the degree of some vertices is preserved under the subgraph isomorphism mapping. This information is explicitly utilized to develop a polynomial-time algorithm. The usefulness of the marked subgraph isomorphism algorithm is demonstrated by solving the subcircuit extraction problem.

It is shown that this restricted form of subgraph isomorphism is much more efficient than general subgraph isomorphism algorithms.

We begin our discussion with some definitions and notations. In Section 3 we develop the marked subgraph isomorphism algorithm. Its application to the subcircuit extraction problem is described in Section 4. Finally, some discussions conclude the paper.

2 Definitions and notations

In the present paper we consider undirected ordered graphs and their attributed version.

Definition 1. An *ordered graph* is a triple (V, E, L) where (V, E) defines an undirected graph. For each vertex $v \in V$, the edges $vv_1, vv_2, \cdots, vv_k$ incident to v have a unique order which is represented by a cyclic list $L(v)$. L is the set of the lists $L(v)$ for each $v \in V$. The number k is called the degree of v, denoted by $d(v)$.

Definition 2. $G_1 = (V_1, E_1)$ is a *subgraph* of $G_2 = (V_2, E_2)$ if there exists an injective mapping $f : V_1 \to V_2$ such that $v_1v_2 \in E_1 \Rightarrow f(v_1)f(v_2) \in E_2$ holds. That is, v_1v_2 is an edge of G_1 iff $f(v_1)f(v_2)$ is an edge of G_2. The mapping f is called *subgraph isomorphism.*

Subgraph isomorphism in general is a well-known NP-complete problem [2]. In the present paper we restrict our efforts to the special case of marked subgraph isomorphism. In the general subgraph isomorphism problem, we have $d(v) \leq d(f(v))$, $v \in V_1$. In some applications we know *a priori* a subset $V_1^* \subseteq V_1$ for which the vertex degree is preserved under the subgraph isomorphism mapping. That is, for each $v \in V_1^*$, $d(v) = d(f(v))$ must be fulfilled by the mapping.

Definition 3. Given a partition $V_1 = V_1^* \cup V_1^+$, $V_1^* \cap V_1^+ = \emptyset$, a *marked subgraph isomorphism* f is a constrained form of subgraph isomorphism such that for each $v \in V_1^*$, $d(v) = d(f(v))$ holds. We call the vertices of V_1^* and V_1^+ *internal* and *external* vertices, respectively.

In dealing with ordered graphs we also require the preservation of order information under the subgraph isomorphism mapping f.

Definition 4. Given two ordered graphs $G_1 = (V_1, E_1, L_1)$ and $G_2 = (V_2, E_2, L_2)$, a marked subgraph isomorphism f from G_1 to G_2 implies that, if for any internal vertex $v \in V_1^*$, we have $L_1(v) = (vv_1, vv_2, \cdots, vv_k)$, then $L_2(f(v)) = (f(v)f(v_1), f(v)f(v_2), \cdots, f(v)f(v_k))$ holds.

Figure 1 shows three ordered graphs where the edges incident to a vertex are assumed to be ordered clockwise. Clearly, there exists a subgraph isomorphism

$$f_{13} = \{(v_{1k}, v_{3k}) \mid k = 1, 2, \cdots, 8\}$$

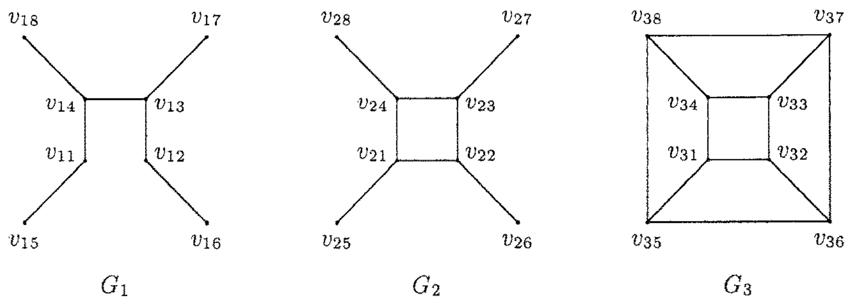

Fig. 1. Three ordered graphs.

from G_1 to G_3. Under the constraint partition $V_1^* = \{v_{11}, v_{12}, v_{13}, v_{14}\}$, however, f_{13} is not a marked subgraph isomorphism because both v_{11} and v_{12} have a degree different from their respective mapping in G_3. On the other hand, the mapping

$$f_{23} = \{(v_{2k}, v_{3k}) \mid k = 1, 2, \cdots, 8\}$$

represents a marked subgraph isomorphism under the constraint partition $V_2^* = \{v_{21}, v_{22}, v_{23}, v_{24}\}$ from G_2 to G_3.

Notice that in Definition 4 we don't ask for the order preservation of external vertices. If this is needed as well, the definition of marked subgraph isomorphism of ordered graphs should be extended as follows. For any external vertex $v \in V_1^+$, if we have $L_1(v) = (vv_1, vv_2, \cdots, vv_k)$, then $(f(v)f(v_1), f(v)f(v_2), \cdots, f(v)f(v_k))$ must be a sublist of $L_2(f(v))$.

Definition 5. An *attributed ordered graph* G is a 5-tuple, $G = (V, E, L, \mu, \nu)$ where (V, E, L) defines an ordered graph. The function $\mu : V \to A_V$ assigns attributes to the vertices. The function $\nu : E \to A_E$ assigns attributes to the edges; the edges v_iv_j and v_jv_i have identical attributes. The sets A_V and A_E of attributes may be numerical values, symbolic labels, or vectors of any type.

The (marked) subgraph isomorphism of attributed ordered graphs may be defined in different ways. We may only require that the basic structures of two ordered graphs without attributes are isomorphic. Alternatively, we may additionally ask for the preservation of attributes, i.e., $\mu_1(v) = \mu_2(f(v))$ for any vertex $v \in V_1$, and $\nu_1(v_iv_j) = \nu_2(f(v_i)f(v_j))$ for any edge $v_iv_j \in E_1$.

3 Marked subgraph isomorphism algorithm

The definition of marked subgraph isomorphism lends itself to the following straightforward algorithm. By using an arbitrary subgraph isomorphism algorithm we find out the subgraph isomorphism mappings between two graphs. These mappings are then checked for the fulfillment of order and degree constraints. This approach, however, needs exponential computation time in the worst case. In the following we utilize the order and degree information to derive a polynomial-time algorithm.

step	used	Q	$S(v_{11}v_{15})$	$C(v_{11}v_{15})$
0	$U_0 = \{v_{11}\}$	(v_{11}, v_{15})	$\emptyset$	$\emptyset$
1	$U_1 = U_0 \cup \{v_{15}, v_{14}\}$	(v_{14}, v_{11})	$S_1 = v_{15}v_{14}$	$C_1 = 23$
2	$U_2 = U_1 \cup \{v_{18}, v_{13}\}$	(v_{13}, v_{14})	$S_2 = S_1v_{11}v_{18}v_{13}$	$C_2 = C_1145$
3	$U_3 = U_2 \cup \{v_{17}, v_{12}\}$	(v_{12}, v_{13})	$S_3 = S_2v_{14}v_{17}v_{12}$	$C_3 = C_2367$
4	$U_4 = U_3 \cup \{v_{16}\}$	$\emptyset$	$S_4 = S_3v_{13}v_{16}$	$C_4 = C_358$

Table 1. Generation of the code $C(v_{11}v_{15})$.

3.1 Graph coding

To determine whether G_1 is a subgraph of G_2 under the constraint partition G_1^*, we compute a code of G_1 for some fixed start edge $v_iv_j \in E_1$ as follows. We assume that the start vertex v_i be an internal vertex, i.e., $v_i \in V_1^*$. At first, all vertices of G_1 except v_i are set to unused. In addition, we need a queue Q initialized to (v_i, v_j) and a string $S(v_iv_j)$ being initially empty. We always fetch the first record (v_i^*, v_j^*) from Q. Starting from v_j^*, we then visit all vertices v_k incident to v_i^* according to the order $L_1(v_i^*)$ and append them in this order to $S(v_iv_j)$. Each time an unused internal vertex $v_k \in V_1^*$ is encountered, we put a record (v_k, v_i^*) to the queue Q and set v_k to used. This process is repeated until the queue Q becomes empty. Now we add a numbering scheme to the string $S(v_iv_j)$ so that a code is obtained. Each vertex is labeled by a natural number, beginning with 1 in the order the vertices are set to used. Thus, the label for v_i is 1, the label for v_j is 2, the other $d(v_i) - 1$ neighbors of v_i are labeled by $3, 4, \cdots, d(v_i) + 1$, respectively. Essentially, this is a relabeling of the vertices of G_1. The numbering of the string $S(v_iv_j)$ results in a string of natural numbers which we call the code for v_iv_j, denoted by $C(v_iv_j)$.

To illustrate the coding procedure, we consider the ordered graph G_1 shown in Figure 1 under the constraint partition $V_1^* = \{v_{11}, v_{12}, v_{13}, v_{14}\}$. Using $v_{11}v_{15}$ as the starting edge, Table 1 lists the used vertices, the contents of the queue Q, and the string S at each step. Therefore, we have:

$$S(v_{11}v_{15}) = v_{15}v_{14}v_{11}v_{18}v_{13}v_{14}v_{17}v_{12}v_{13}v_{16}.$$

Accordingly, the code $C(v_{11}v_{15})$ is:

$$C(v_{11}v_{15}) = 2314536758.$$

Let $V_1^* = \{v_{i1} = v_i, v_{i2}, \cdots, v_{ik}\}$ and the labels of $v_{i1}, v_{i2}, \cdots, v_{ik}$ in $C(v_iv_j)$ be of ascending order. The string $S(v_iv_j)$ has the following structure. The first $d(v_{i1})$ vertices of $S(v_iv_j)$ are the vertices incident to v_{i1} in an order defined by $L_1(v_{i1})$. Th next $d(v_{i2})$ vertices are the vertices incident to v_{i2}, and so on. Finally, $S(v_iv_j)$ ends with the $d(v_{ik})$ vertices incident to v_{ik}. Therefore, the length of $S(v_iv_j)$ and $C(v_iv_j)$ is given by:

$$|S(v_iv_j)| = |C(v_iv_j)| = \sum_{v_k \in V_1^*} d(v_k).$$

Notice that dependent on the constraint partition of G_1, the coding procedure described above may not reach all vertices of G_1. Reachable are vertices v_k for which there exists at least one chain of connected edges from v_i to v_k such that the vertices from v_i to the vertex before v_k on the chain are all internal vertices. In the present paper we only allow those partitions of G_1 that enable the reachability of all vertices of G_1.

3.2 Algorithm

Given the code $C(v_iv_j)$ of G_1, our algorithm considers all directed edges $v_i'v_j'$ of G_2. For each $v_i'v_j'$, we generate a code $C(v_i'v_j')$ and this code is then used for subgraph isomorphism testing. The generation of $C(v_i'v_j')$ in G_2 is similar to that of $C(v_iv_j)$ in G_1. At first, all vertices of G_2 except v_i' are set to unused and v_i' is labeled by 1. The queue Q is initialized to (v_i', v_j') and the string $S(v_i'v_j')$ is set to empty. We always fetch the first record (v_i^*, v_j^*) from Q. Starting from v_j^*, we then visit all vertices v_k incident to v_i^* according to the order $L_2(v_i^*)$ and append them in this order to $S(v_i'v_j')$. In addition we append the corresponding labels to $C(v_i'v_j')$. If unused vertices are involved here, then new labels are generated and they are set to used afterwards. Each time an unused vertex v_k is encountered, we also look at the vertex in G_1 that has the same label as v_k. If that vertex is an internal vertex, then we put a record (v_k, v_i^*) to the queue Q. This process is repeated until either $C(v_i'v_j')$ becomes of the same length as $C(v_iv_j)$ or the queue Q becomes empty.

The following two theorems are fundamental to our subgraph isomorphism algorithm.

Theorem 6. If there exists a marked subgraph isomorphism f from G_1 to G_2 such that f maps v_i and v_j to v_i' and v_j', respectively, then we have $C(v_iv_j) = C(v_i'v_j')$.

Proof For notational simplicity we relabel the vertices v_i' of G_2 such that $f(v_i) = v_i'$ is satisfied. We consider an ordered subgraph $\hat{G}_2 = (\hat{V}_2, \hat{E}_2, \hat{L}_2)$:

$$\hat{V}_2 = \{v_i' \mid v_i \in V_1\}$$
$$\hat{E}_2 = \{v_i'v_j' \mid v_iv_j \in E_1\}$$

The order relationship $\hat{L}_2(v_i')$ for each $v_i' \in \hat{V}_2$ is inherited from the corresponding vertex v_i of G_1. The graph $\hat{G}_2$ actually represents a mirrored version of G_1 in G_2. By introducing the constraint partition:

$$\hat{V}_2^* = \{v_i' \mid v_i \in V_1^*\}$$

on $\hat{G}_2$, the coding procedure described in Section 3.1 yields the code $\hat{C}(v_i'v_j')$ on $\hat{G}_2$ for the start edge $v_i'v_j'$. Clearly, $\hat{C}(v_i'v_j') = C(v_iv_j)$ holds. On the other hand, the generation of $\hat{C}(v_i'v_j')$ on $\hat{G}_2$ is exactly mimed by the coding procedure for $C(v_i'v_j')$ on G_2. Therefore, the relationship $C(v_iv_j) = C(v_i'v_j')$ is proved. □

Theorem 7. If $C(v_iv_j) = C(v_i'v_j')$ holds, then under the two conditions

1. For each internal vertex $v_k \in V_1^*$, the corresponding vertex v_k' (v_k' and v_k have the same label) has the same degree as v_k;
2. For each edge $v_kv_l \in E_1$, $v_k \in V_1^+$, $v_l \in V_1^+$, there is an edge $v_k'v_l' \in E_2$, where $v_k'(v_l')$ has the same label as $v_k(v_l)$;

there exists an ordered subgraph isomorphism f from G_1 to G_2 such that f maps v_i and v_j to v_i' and v_j', respectively.

Proof We relabel the vertices of G_1 by their corresponding number in $C(v_iv_j)$. Similarly, a vertex of G_2 covered in $S(v_i'v_j')$ is relabeled by k' if that vertex has a number k in $C(v_i'v_j')$; vertices of G_2 not covered in $S(v_i'v_j')$ are irrelevant in this context and are labeled so that no confusion arises. After the relabeling the initial items $S(v_iv_j)$, $C(v_iv_j)$, $S(v_i'v_j')$ and $C(v_i'v_j')$ become $S(12)$, $C(12)$, $S(1'2')$, and $C(1'2')$, respectively. In addition $S(12) = C(12)$ and $S(1'2') = C(1'2')$ hold. In the following we prove that the mapping $f(b) = b'$ is a marked subgraph isomorphism from G_1 to G_2.

Assume $V_1^* = \{n_1 = 1, n_2, \cdots, n_k\}$, i.e., there are k internal vertices $1 = n_1 < n_2 < \cdots < n_k$ in G_1. Then, the first $d(n_1)$ numbers of $C(v_{12})$ must be the vertices incident to vertex n_1, and their order in $C(12)$ corresponds to the ordered list $L_1(n_1)$ in G_1. Similarly, the next $d(n_2)$ numbers in $C(12)$ describe the local ordered structure of vertex n_2, and so on. On the other hand, the coding procedure for $C(1'2')$ guarantees that the first $d(n_1')$ numbers of $C(1'2')$ represent the local ordered structure of vertex n_1', and the next $d(n_2')$ the local ordered structure of n_2', etc. From the facts $C(12) = C(1'2')$ and $d(n_i) = d(n_i')$, $n_i \in V_1^*$, (condition 1) it follows that under the mapping $f(b) = b'$, each edge $n_iv \in E_1$, $n_i \in V_1^*$, in G_1 has a corresponding edge $n_i'v'$ in G_2, and the order relationships for all internal vertices of G_1 are retained. It remains to show that all edges $v_kv_l \in E_1$, $v_k \in V_1^+$, $v_l \in V_1^+$, consisting of only external vertices in G_1 have corresponding edges in G_2 as well. This is given by condition 2. In summary, the mapping $f(b) = b'$ defines a marked subgraph isomorphism from G_1 to G_2. □

Note that the two conditions in Theorem 7 don't put any further constraints on the graphs that can be handled by the proposed algorithm. Instead, they are complementary conditions on the mapping f besides the code equivalence so that f represents a true subgraph isomorphism. Condition 2, for instance, is necessary because information about edges connecting two external vertices is *not* included in the code.

Based on the two theorems above we propose the following algorithm for finding all marked subgraph isomorphisms from G_1 and G_2:

step	used	Q	$S(v_{31}v_{35})$	$C(v_{31}v_{35})$
0	$U_0 = \{v_{31}\}$	(v_{31}, v_{35})	$\emptyset$	$\emptyset$
1	$U_1 = U_0 \cup \{v_{35}, v_{34}, v_{32}\}$	$(v_{34}, v_{31}), (v_{32}, v_{31})$	$S_1 = v_{35}v_{34}v_{32}$	$C_1 = 234$

Table 2. Generation of the code $C(v_{31}v_{35})$.

1. Choose arbitrarily a directed edge $v_iv_j \in E_1$, $v_i \in V_1^*$, of G_1 and compute $S(v_iv_j)$ and $C(v_iv_j)$.
2. For each directed edge $v_i'v_j'$ of G_2 do steps 2.1 and 2.2.
2.1. Compute $S(v_i'v_j')$ and $C(v_i'v_j')$. If $C(v_iv_j) \neq C(v_i'v_j')$, go to step 2 for the next edge of G_2.
2.2. Test the two conditions in Theorem 7. If both are fulfilled, then there exists a marked subgraph isomorphism from G_1 to G_2 that maps each vertex in $S(v_iv_j)$ to the corresponding vertex in $S(v_i'v_j')$.

For more efficiency this algorithm can be modified in the following way. If some vertex number during the generation of $C(v_i'v_j')$ is not identical to the corresponding vertex number in $C(v_iv_j)$, we stop the coding process immediately and turn to the next edge of G_2.

To illustrate the algorithm, we verify that the mapping f_{13} given in Section 2 is not a marked subgraph isomorphism from G_1 to G_3 shown in Figure 1. For G_1 we compute $S(v_{11}v_{15})$ and $C(v_{11}v_{15})$, see Section 3.1. Step 2 of the algorithm computes $S(v_{31}v_{35})$ and $C(v_{31}v_{35})$, and is tabulated in Table 2. Already after a single step a difference between $C(v_{11}v_{15})$ and $C(v_{31}v_{35})$ occurs. Therefore, we conclude that there is no marked subgraph isomorphism from G_1 and G_3 that maps $v_{11}v_{15}$ to $v_{31}v_{35}$ and immediately turn to the next edge of G_2. On the other hand, it is easy to verify that f_{23} is a marked subgraph isomorphism from G_2 to G_3.

So far we have considered unattributed graphs. The extension to attributed ordered graphs is straightforward. When we have reached a vertex v during the generation of $C(v_i'v_j')$, we ask for the equivalence of the attribute of v with that of the corresponding vertex in G_1. In addition, we have to make sure that the attribute of the edge on which v has just been reached agrees with that of the corresponding edge in G_1. Finally, the test of condition 2 in step 2.2 of the algorithm must include an attribute equivalence test as well.

Now we analyze the complexity of the algorithm. It is assumed that ordered graphs are represented by a suitable data structure so that the next edge of a vertex relative to another edge is retrieved in constant time. For example, this can be achieved by storing the edges incident to a vertex in an array. Let $G_1(G_2)$ have $n_1(n_2)$ vertices and $m_1(m_2)$ edges. The generation of the code $C(v_iv_j)$ for a particular directed edge v_iv_j in G_1 requires $O(m_1)$ operations. For each of the $2m_2$ directed edges in G_2, step 2.1 requires $O(m_1)$ time while step 2.2 is done in $O(n_1 + m_1)$ time. Therefore, the algorithm has an $O((n_1 + m_1)m_2)$ time complexity totally. Since $n_1 - 1 \leq m_1$, the time complexity is finally quantified by $O(m_1 m_2)$. The space requirement includes $O(m_1 + m_2)$ for the data structure of the ordered graphs and $O(n_1 + n_2)$ for the queue Q. Totally, we need therefore $O(m_1 + m_2)$ space.

Finally, it is worth mentioning that if Definition 4 is extended to require the order preservation of external vertices, then step 2.2 of the marked subgraph isomorphism algorithm should include an order preservation test for all external vertices of G_1 accordingly.

4 Application: Subcircuit extraction

The problem of finding subcircuits in a larger circuit arises in many contexts in computer-aided design. Most of the proposed algorithms for this task rely on specific characteristics of the circuit technology. Recent efforts to technology-independent subcircuit extraction are reported in [10, 11]. The SubGemini system described in [10] is based on a general subgraph isomorphism algorithm.

We propose to use the marked subgraph isomorphism algorithm developed in the last section for the subcircuit extraction problem. Frequently, circuit design is done by using library based cells. Then, the library cells can be used as the subgraphs to be identified in the main circuit. In this case the order information of the library cells is preserved in the main circuit and this justifies the use of ordered graphs for modeling circuits.

We augment the circuit graph representation suggested in [10] by the order information. A circuit graph consists of device vertices and net (wire) vertices. The device vertices may represent transistors, gates, etc, while the net vertices are the terminals or metallic contacts to which the devices are connected. A connection between two devices in the circuit is modeled by connecting each device vertex to a net vertex. There is a natural partition of a subcircuit. All device vertices are internal. A net vertex is considered as external if it is connected to at least one device outside the subcircuit. For all internal vertices the vertex degree remains the same in the main circuit. Now the subcircuit extraction problem can be solved by finding all marked subgraph isomorphisms from the subcircuit graph to the main circuit graph.

Several experiments have been done to test our approach of subcircuit extraction. Table 3 lists the results for some large circuits. The size of circuits is given in terms of the number of transistors. In all experiments, the occurrences of the subcircuit have been completely found. The run time for our algorithm (MSI) and, for comparison purpose, the SubGemini system is also reported in Table 3, both measured on a Sun SparcStation 5. Our approach using a restricted form of subgraph isomorphism is clearly much faster than SubGemini which is based on an general subgraph isomorphism technique.

main circuit G	subcircuit S	size G	size S	#solutions found	time (ms) MSI	time (ms) SubGemini
RAM array	RAM cell	3000	6	500	28	1220
		6000	6	1000	57	2690
		12000	6	2000	110	6640
muxes	echain	1020	2	340	17	1670
	nchain	1020	2	340	17	1790
	gnd-echain	1020	2	0	11	240

Table 3. Experimental results of subcircuit extraction.

5 Conclusion

For many applications the full power of matching techniques for general graphs is not really necessary. Frequently, constrained graphs allow efficient, sometimes even low-polynomial time, algorithms. In the present paper we have developed an $O(m_1 m_2)$ time algorithm for the marked subgraph isomorphism of ordered graphs. It turns out to be useful for solving the subcircuit extraction problem. Our approach has been demonstrated to be much more efficient than general subgraph isomorphism techniques.

Acknowledgments

We want to thank N. Vijaykrishnan of Univ. of South Florida, Tampa, for valuable discussions on the subcircuit extraction problem, and C. Ebeling of Univ. of Washington for providing us the SubGemini program and the test data.

References

1. P. Dublish, Some comments on the subtree isomorphism problem for ordered trees, Information Processing Letters, 36: 273–275, 1990.
2. M.R. Garey and D.S. Johnson, *Computers and intractability: A guide to the theory of NP-completeness*, Freeman and Co., 1979.
3. J.E. Hopcroft and J.K. Wong, Linear time algorithm for isomorphism of planar graphs, *Proc. of 6th Annual ACM Symposium on Theory of Computing*, 172–184, 1974.
4. X. Jiang and H. Bunke, A simple and efficient algorithm for determining the symmetries of polyhedra, *Graphical Models and Image Processing*, 54(1):91–95, 1992.
5. X. Jiang, K. Yu, and H. Bunke, Detection of rotational and involutational symmetries and congruity of polyhedra, *The Visual Computer*, 12(4): 193–201, 1996.
6. X. Jiang and H. Bunke, Including geometry in graph representations: A quadratic-time graph isomorphism algorithm and its applications, In *Advances in Structural and Syntactical Pattern Recognition* (P. Perner, P. Wang, A. Rosenfeld, Eds.), Lectures Notes in Computer Science 1121, Springer-Verlag, 110–119, 1996.

7. X. Jiang and H. Bunke, On the coding of ordered graphs, *Computing*, 1998. (to appear)
8. S. Kasif, L. Kitchen, and A. Rosenfeld, A Hough transform technique for subgraph isomorphism, Pattern Recognition Letters, 2: 83–88, 1983.
9. E.M. Luks, Isomorphism of graphs of bounded valence can be tested in polynomial time, *Journal of Computer and System Science*, 25: 42–65, 1982.
10. M. Ohlrich *et al.*, SubGemini: Identifying subcircuits using a fast subgraph isomorphism algorithm, *Proc. of 30th ACM/IEEE Design Automation Conference*, 31–37, 1993.
11. N. Vijaykrishnan and N. Ranganathan, SUBGEN: A genetic approach for subcircuit extraction, *Proc. of Int. Conf. on VLSI Design*, Bangalore, 1996.