Isomorphic Subgraphs

Sabine Bachl

University of Passau, 94030 Passau, Germany
Sabine.Bachl@informatik.uni-passau.de

Abstract. We are interested in finding symmetries in graphs and then use these symmetries for graph drawing algorithms.
There are two general approaches to this problem, the first one is known as GEOMETRIC SYMMETRIES on the basis of drawings, the other rests upon the graph-theoretical notion of graphs. For a given graph G the ISOMORPHIC SUBGRAPHS problem makes use of the second approach and tries to find the two largest disjoint isomorphic subgraphs in G. Hence, G consists of two identical copies and a remainder.
There are many NP-complete or open problems related to our problem, like GRAPH ISOMORPHISM, GRAPH AUTOMORPHISM or LARGEST COMMON SUBGRAPH.
We show that the ISOMORPHIC SUBGRAPHS problem is NP-hard for connected outerplanar graphs, and 2-connected planar graphs and is solvable in linear time when restricted to trees.
Additionally we will shortly discuss the applicability of ISOMORPHIC SUBGRAPHS in graph drawing algorithms.

1 Introduction

As graphs are used in various application areas, symmetric drawings are desirable for a better and faster understandability ([17],[16]). Therefore, symmetry is one of the major aesthetics in graph drawing apart from edge crossing, edge length and routing. There are two different approaches to symmetry. The first one is based on the drawing itself, the second rests upon the graph-theoretical notion of graphs.

In the first approach, symmetry in drawings is defined as a geometric property, using rotations or reflections. This has been investigated in depth by Manning [13] and by Manning and Atallah ([14],[15]) for trees, outerplanar graphs and embedded graphs. They proved that several variants of GEOMETRIC SYMMETRIES are NP-complete. Lipton et al. [11] described a model for measuring the symmetry of straight-line drawings.

An investigation on symmetry is also possible on the basis of the graph-theoretical definition of a graph $G = (V, E)$ as a set of nodes and a set of edges. This view leads to the notion of isomorphism, automorphism or homomorphism. Throughout the last decades many people have contributed to this area.
First of all there is the GRAPH ISOMORPHISM problem, which is one of the challenging open problems in complexity theory ([8],[20]). GRAPH ISOMORPHISM

is neither known to be NP-complete nor exists a polynomial algorithm to solve this problem. A detailed summary of various heuristics can be found in [18]. They are all based on so-called graph invariants which remain unchanged under isomorphism, like node degree etc.
A well-known NP-complete problem in this field is SUBGRAPH ISOMORPHISM (GT48 in [6]), containing CLIQUE, COMPLETE BIPARTITE SUBGRAPH or HAMILTONIAN as a special case. SUBGRAPH ISOMORPHISM is also NP-complete for outerplanar graphs ([21]), sets of chains and forests ([6]). There exist polynomial algorithms if both graphs are trees ([19]) or 2-connected outerplanar ([10]).
Another related NP-complete problem is LARGEST COMMON SUBGRAPH (GT49 in [6]). It is only polynomial solvable if the graphs are trees. Levi and Luccio ([9]) proposed a heuristic which is also based on graph invariants.
Further related NP-complete problems are ISOMORPHISM WITH RESTRICTIONS ([12]), MAXIMUM SUBGRAPH MATCHING (GT50 in [6]), GRAPH-K-ISOMORPHISM ([24]), GRAPH AUTOMORPHISM and many variants ([12]).

The ISOMORPHIC SUBGRAPHS problem can be described as searching for the two largest isomorphic disjoint subgraphs in a given graph. It is defined formally in Section 2. There exist two variants of the problem: IES and INS where the subgraphs can be either edge-induced or node-induced (see Section 2).

Most of the known NP-complete problems in this area except GRAPH AUTOMORPHISM and GEOMETRIC SYMMETRIES rest upon two given graphs. In the ISOMORPHIC SUBGRAPHS problem only one graph is given and therefore the cut-line must be found in order to reduce our problem to the known NP-complete problem. One might think, that the ISOMORPHIC SUBGRAPHS problem is a variation of the well-known SUBGRAPH ISOMORPHISM problem with graphs G and H now taking the disjoint union $G \cup H$. However, this not true in general. $G \cup H$ may contain isomorphic subgraphs which are each larger than G. Easy reductions to the other similar NP-complete problems fail, too.
For the purpose of displaying symmetries in graphs, GRAPH AUTOMORPHISMS are too restrictive. A few nodes or edges may destroy a nontrivial automorphism. The ISOMORPHIC SUBGRAPHS problem is a less restrictive approach where it is possible to have a remainder in the given graph consisting of unused nodes and edges.
At first sight, GEOMETRIC SYMMETRIES are very similar to the ISOMORPHIC SUBGRAPHS problem. But they have only edges in the remainder and additionally these edges must fulfil a certain symmetry along the axis of symmetry. Our remainder consists of nodes and edges and is completely disregarded.

Finding the isomorphic subgraphs is not the only interesting point of view. Its integration in a layout algorithm will be our focus in the future.

In Section 2 we introduce basic notations and definitions. In Section 3 we prove the NP-completeness of the ISOMORPHIC SUBGRAPHS problem for outerplanar connected and planar 2-connected. This improves and complements our proof for general graphs ([3]) which uses dense graphs with $\Omega(n^2)$ edges. The focus of Section 4 is a linear time algorithm for trees. We conclude with open problems and an outlook.

2 Basic notions

Let $\mathcal{A}$ be an alphabet and let $\$ \notin \mathcal{A}$ be a new symbol.
For a *string* $w = a_1..a_n$ over $\mathcal{A}$ let $|w| = n$ denote its *length.* The *substring* $a_i..a_j$ of w is denoted by $w[i, j]$. If $j = n$, $w[i, n]$ is called *suffix* of w beginning with the i-th symbol.
The *suffix tree* of a string $w\$$ is a rooted tree with $|w| + 1$ leaves, such that every internal node except the root has at least two sons. Every edge is labeled with a nonempty substring of w, such that the labels of edges leaving a node begin with different symbols, and every suffix of $w\$$ is obtained by a concatenation of the labels of the edges on a path from the root to a leaf. The leaf representing the suffix $w[i, n]$ is labeled by i.
Thus, a suffix tree is a compact data structure for the set of suffixes. They are commonly used for an efficient computation of common substrings. We use them for the computation of the largest isomorphic subtrees.
There are several linear time algorithms for the computation of a suffix tree, see e.g. [22], [23] or [7].
Let us now consider an example. The suffix tree of the Fibonacci word $w = abaababa\$$ is shown in figure 1.

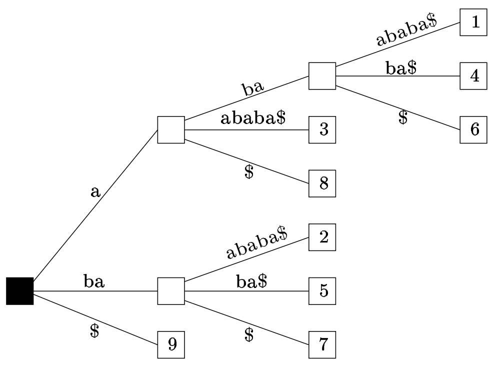

Fig. 1: Suffix tree for the Fibonacci word $w = abaababa\$$.

Next we recall some graph theoretic notions. A graph $G = (V, E)$ consists of a set of nodes V and undirected edges $E = \{\{v_i, v_j\} | v_i, v_j \in V\}$. For convenience, we exclude self-loops and multiple edges. Let the numbers of nodes and edges be denoted by $|V|$ and $|E|$. Two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ are isomorphic, if there is a bijection $f : V_1 \to V_2$ such that $\{u, v\} \in E_1$ if and only if $\{f(u), f(v)\} \in E_2$.

Let $G = (V, E)$ be a graph and $V' \subseteq V$. The subgraph $H = G|_{V'}$ consisting of all nodes V' and the edges $E' = \{\{v_i, v_j\} \in E | v_i, v_j \in V'\}$ between the nodes of V' in G is called *Node-Induced Subgraph.*

Let $G = (V, E)$ be a graph and $E' \subseteq E$. The subgraph $H = G|_{E'}$ consisting of all edges E' and their adjacent nodes is called *Edge-Induced Subgraph.*

For our NP-completeness results we reduce 3-Partition ([6]):

Instance: A finite set $A = a_1, \dots, a_{3m}$ of $3m$ elements ($m \in \mathbb{N}$), a bound $B \in \mathbb{N}$, a 'size' $s(a) \in \mathbb{N}$ for each $a \in A$, such that each $s(a)$ satisfies $\frac{B}{4} < s(a) < \frac{B}{2}$ and $\sum_{a \in A} s(a) = m \cdot B$.
Question: Can A be partitioned into m disjoint sets $A_1, \dots, A_m$ such that, for $1 \le i \le m$, $\sum_{a \in A_i} s(a) = B$. (Notice that the above constraints on the item sizes imply that every such A_i must contain exactly three elements.)

We now come to the central notion of this paper: the Isomorphic Subgraphs problems IES and INS.

Isomorphic Edge-Induced Subgraphs, IES
Instance: A graph $G = (V, E)$ and a positive integer K.
Question: Does G contain two disjoint isomorphic edge-induced subgraphs with at least K edges, i.e. are there two sets of edges E_1 and E_2 with $E_1 \cap E_2 = \emptyset$ such that the subgraphs $H_1 = G|_{E_1} = (V_1, E_1)$ and $H_2 = G|_{E_2} = (V_2, E_2)$ are isomorphic, $V_1 \cap V_2 = \emptyset$ and $|E_1| = |E_2| \ge K$.

Isomorphic Node-Induced Subgraphs, INS
Instance: A graph $G = (V, E)$ and a positive integer K.
Question: Does G contain two disjoint isomorphic node-induced subgraphs with at least K edges, i.e. are there two sets of nodes V_1 and V_2 with $V_1 \cap V_2 = \emptyset$ such that the induced subgraphs $H_1 = G|_{V_1} = (V_1, E_1)$ and $H_2 = G|_{V_2} = (V_2, E_2)$ are isomorphic and $|E_1| = |E_2| \ge K$.

3 NP-Completeness of IES and INS

In this section, we prove that IES and INS are NP-complete even for connected outerplanar graphs and two-connected planar graphs. It should be noted that IES is also NP-complete for instances where the graphs are dense ([3]).

3.1 Outerplanar Graphs

Theorem 3.1. *IES is NP-complete for connected outerplanar graphs.*

Proof. We reduce 3-Partition to IES.
Let $A = \{a_1, \dots, a_{3m}\}$ and $B \in \mathbb{N}$ be an instance of 3-Partition.
We construct a graph G as follows (see Figure 2): The left subgraph L consists

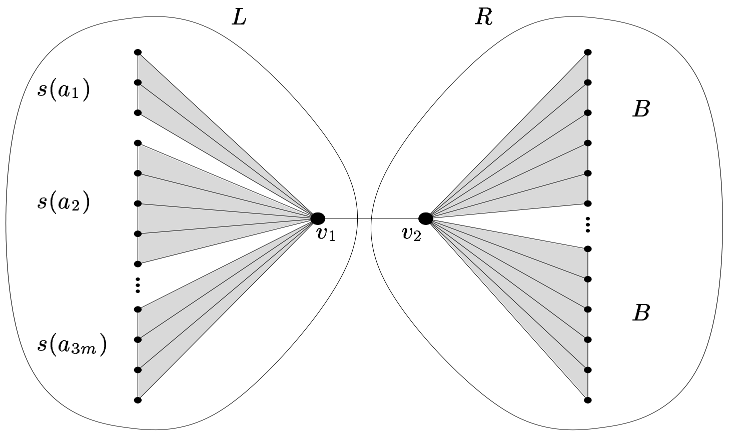

Fig. 2: Reduction to IES for connected outerplanar graphs

of $3m$ fans, where the i-th fan is a chain of $s(a_i)$ nodes, all connected to v_1. The right subgraph R has m fans each consisting of a chain of B nodes connected to v_2. There is an edge between v_1 and v_2. Note that the constructed graph is outerplanar connected.
In the following proof, the i-th fan of L is called $s(a_i)$-fan, the fans of R are called B-fans.

Let $K = 2Bm - 3m$.

Claim. IES has a solution for G and K if and only if A has a 3-Partition.

Proof.
$\Leftarrow$: *Let $A_1, \ldots, A_m$ be a 3-Partition of A.*
Define $H_1 = L$ and $H_2 = R$ and a bijection $f : H_1 \to H_2$, such that v_1 is mapped on v_2 and for all $a \in A_j$ $(1 \le j \le m)$ the $s(a)$-fans are mapped completely onto the j-th B-fan. Hence, H_1 and H_2 have at least

$$\sum_{a \in A} (s(a) - 1) + s(a) = 2Bm - 3m = K$$

common edges and IES has a solution with $K = 2Bm - 3m$.
$\Rightarrow$: *Let H_1 and H_2 be a solution of IES and $K = 2Bm - 3m$.*
(a) *Then we can prove that one of the graphs is equal to L and the other one is equal to R.*
First, $v_1 \in H_1$ and $v_2 \in H_2$ (w.l.o.g.). Assume that v_1 and v_2 belong to the same subgraph, say H_1. By the construction of G, $deg(v_1) = deg(v_2) = Bm + 1$ and $deg(v) \le 3$ for all $v \in V \setminus \{v_1, v_2\}$.
Therefore mapping v_1 and v_2 on two nodes of H_2 implies a loss of at least $2 \cdot (Bm+1-3)$ edges. Then H_1 and H_2 have each at most $\left\lfloor \frac{|E| - 2\cdot(Bm-2)}{2} \right\rfloor$

edges with $|E| = 4Bm - 4m + 1$*. Since for all* $B > 1$*:*

$$\left\lfloor \frac{|E| - 2 \cdot (Bm - 2)}{2} \right\rfloor < K - 3$$

we have a contradiction to the number of edges in every subgraph.
Thus, v_1 *and* v_2 *do not belong to the same subgraph and are mapped onto each other.*
It remains to show that H_1 *has no node of* R *and* H_2 *no node of* L*. Such a mapping would lead to a loss of at least 4 edges which is again a contradiction to* $|E_1| = |E_2| > K$*.*
Hence $H_1 = L$ *and* $H_2 = R$*.*

(b) In order to show that 3-Partition has a solution, it is sufficient to show that exactly three $s(a_i)$*-fans must be mapped completely onto one* B*-fan. Since* H_1 *has* K *edges, every* $s(a_i)$*-fan must be completely mapped onto one* B*-fan, otherwise we loose at least one edge.*
Since $\frac{B}{4} < s(a) < \frac{B}{2}$*, exactly three* $s(a_i)$*-fans must be mapped onto one* B*-fan. Hence the 3-Partition has a solution which is given uniquely by the mapping of nodes and fans.* □

Theorem 3.2. *INS is NP-complete for connected outerplanar graphs.*

Proof. We reduce 3-Partition to INS. Since the proof is very similar to the previous one, we only show how to construct the graph G *(see Figure 3). The left*

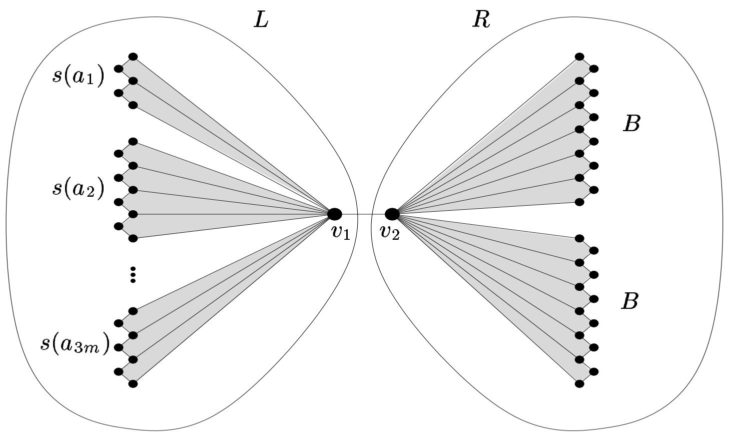

Fig. 3: Reduction to INS for connected outerplanar graphs

subgraph L *consists of* $3m$ *fans, where the* i*-th fan is a chain of* $s(a_i) + (s(a_i) - 1)$ *nodes and every second node is connected to* v_1*. The right subgraph* R *has* m *fans where each fan has exactly* $B + (B - 1)$ *nodes where every second node is connected to* v_2*. The nodes* v_1 *and* v_2 *are adjacent. The value of* K *is* $3Bm - 6m$*.*
By the reasoning as above it can be shown that INS has a solution for G *and* K *if and only if* A *has a 3-Partition.* □

3.2 Planar Graphs

IES and INS are NP-complete for connected planar graphs since outerplanar graphs are a subset of planar graphs. Next we show, that they are also NP-complete for 2-connected planar graphs. Again, the proof is similar to the first one. Therefore, we only present the idea of the proof of IES. The proof of INS is left to the reader.

Theorem 3.3. *IES is NP-complete for 2-connected planar graphs.*

Proof. We reduce 3-Partition to IES.

We construct a graph G (see Figure 4). The left subgraph L consists of $3m$ fans,

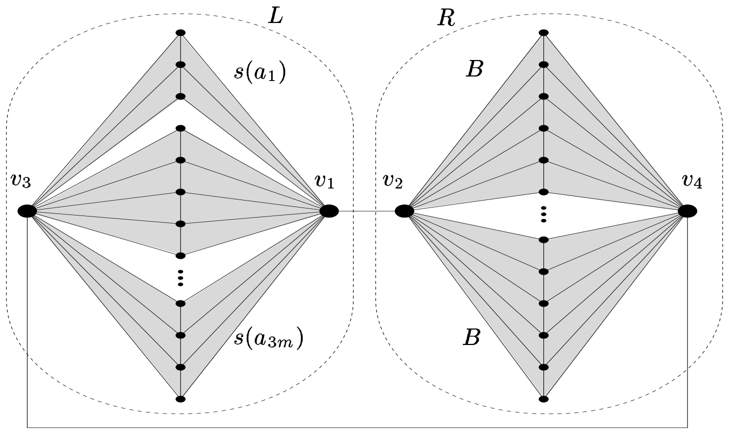

Fig. 4: Reduction to IES for 2-connected planar graphs

where the i-th fan is a chain of $s(a_i)$ nodes, each connected to v_1 and v_3. The right subgraph R consists of m fans where each fan has exactly B nodes, each connected to v_2 and v_4. The nodes v_1 and v_2 are adjacent as are v_3 and v_4. Note that G is 2-connected planar. K is chosen as $3Bm - 3m$.

It can be shown that IES has a solution if and only if A has a 3-Partition. In addition to theorem 3.1, we have to prove that v_1 must be mapped to v_2 and v_3 to v_4 (or v_1 to v_4 and v_3 to v_2); otherwise there is a contradiction to the number of edges in the subgraphs.

□

4 Isomorphic Subtrees

In this section we specialize the Isomorphic Subgraphs problem to trees. We consider free and rooted trees, ordered and unordered trees. For rooted ordered trees the Isomorphic Subgraphs problem is solved by a transformation of trees into strings over pairs of parenthesis and a reduction to the largest identical non-overlapping well-nested substring problem. The other types of trees are reduced to rooted ordered trees.

Let $T = (V, E, r)$ be a tree with root r. A *subtree rooted at r' $T' = (V', E', r')$* consists of all of the descendants of r' (including r' itself) and their connecting edges.

A part of the following algorithm will be the computation of the longest identical non-overlapping well-nested substrings. The *longest identical non-overlapping well-nested substrings* of a string w are the well-nested substrings $w[i, j]$ and $w[p, q]$ with $w[i, j] = w[p, q]$ and $j < p$ of maximal length. We prove in Lemma 4.1 that they can be computed in linear time on suffix trees. If the context is clear, we will refer to the longest identical non-overlapping well-nested substrings by *identical substrings.*

Lemma 4.1. *Let w be a string over some alphabet $\mathcal{A}$. Then the identical substrings can be computed in linear time.*

Proof. *Let T be the suffix tree of w. Let $V^{int} = \{v_1, \ldots, v_m\}$ be the set of internal nodes of T.*
For every $v_i \in V^{int}$ compute the tuple (v_i, s, P) such that s is the concatenation of the labels from the root to v_i and $P = \{p_1, \ldots, p_k\}$ is the set of starting positions of s which is equal to the labels of the leaves of the subtree with root v_i. Insert the tuple into the list L which is sorted by a descending $|s|$ if $|s| \leq \left\lfloor \frac{|w|}{2} \right\rfloor$. Note that strings with $|s| = 1$ are omitted.

Take the first tuple from L. Determine a possible intersection of the intervals $[p_i, p_i + |s| - 1]$ $(i \in \{1, \ldots, k\})$. If there is a p_i and a p_j such that $[p_i, p_i + |s| - 1] \cap [p_j, p_j + |s| - 1] = \emptyset$ and $w[p_i, p_i + |s| - 1]$ and $w[p_j, p_j + |s| - 1]$ are well-nested, the strings $w[p_i, p_i + |s| - 1]$ and $w[p_j, p_j + |s| - 1]$ are the identical substrings. Otherwise take the next tuple out of L and check the conditions again.

As mentioned before the suffix tree can be determined in linear time in the length of the string. The number of internal nodes are at most $|w| - 1$. Therefore the number of tuples is restricted to $O(|w|)$. If $|P| > 2$, $P = \{p_1, \ldots, p_k\}$ should be a sorted list such that we only need to compare p_1 and p_k. If these intervals overlap, the innermost intervals do not fit either and the tuple can be omitted. Note that the sorting of L and P can be done in linear time using Bucket Sort. Thus, the computation of the identical substrings of w is in $O(|w|)$. □

Let T be a rooted ordered tree. The isomorphic subtrees T_1 and T_2 of T are computed in three steps. First of all, the tree T is transformed into a well-nested string $s(T) \subseteq \{(,)\}^*$ over pairs of parenthesis using depth first search. Note that every node of the tree is represented by exactly one pair of parenthesis and the isomorphic subtrees are identical well-nested substrings of $s(T)$. After that, the identical substrings $s_1(T)$ and $s_2(T)$ of $s(T)$ are computed with the algorithm given in Lemma 4.1. As every pair of parenthesis represents exactly one node, the isomorphic subtrees can easily be retrieved from the identical substrings by scanning the substrings and its node information.

In the case of free trees, this are trees with no explicitly given root, finding the largest isomorphic subtrees is done by a reduction to the above algorithm for rooted trees. Every node of the tree is chosen once as a root. After that, the maximum of all results is taken. The runtime of this algorithm is $O(n^2)$ then.

In the case of unordered trees, T is first transformed into a right-heavy tree according to the following rules.
If T_1 and T_2 are subtrees rooted at the sons of some node v, then T_1 is left of T_2 if

1. $height(T_1) < height(T_2)$, or
2. $height(T_1) = height(T_2)$ and $size(T_1) < size(T_2)$, or
3. $height(T_1) = height(T_2)$ and $size(T_1) = size(T_2)$ and $level\_weight(T_1) < level\_weight(T_2)$

where *height* is the length of the longest path to some leaf, *size* is the number of nodes and *level_weight* counts the number of descendants by level and returns the least level with different numbers, the number of descendants of T_1 is less than that of T_2.
After that, the above algorithm for ordered trees can be applied. Since the reordering of the tree is linear, the whole algorithm for unordered trees runs in linear time.

5 Open Questions and Outlook

As our next steps we try to find solutions for IES and INS of 2-connected outerplanar and 3-connected planar graphs. We conjecture that these variants are polynomial solvable.

Additionally, we are working on good heuristics for the NP-complete cases of IES and INS in order to achieve some practical results.
Our first approach to a heuristic for IES is adapted from GRAPH ISOMORPHISM heuristics which are based on so-called graph invariants. These are graph-theoretical properties which remain unchanged under isomorphism.
Since there is normally a remainder in our graph, it is not possible to use these invariants immediately. Instead, we determine the locally best mapping and continue our search with the help of maximal weighted matching algorithms for bipartite graphs.
The rating of a mapping between two nodes v_i and v_j depends for example on the difference of degree, the minimal distance between v_i and v_j and the number of neighbours of v_i and v_j which can be mapped in the next step.

Our idea for the heuristic of INS is similar to the idea of IES except that we operate on faces, not only on nodes. The criteria for good mappings are now for example the number of nodes in the faces or their distance.

Finally, we will try to include the information of isomorphic subgraphs into drawing algorithms. For a better understandability, the isomorphic subgraphs should be drawn identical. For this purpose, we may borrow techniques from the design of hierarchical methods ([1]) and layout graph grammars ([2]). It is also possible to integrate the results into springembedders.
The hierarchical methods are used for drawing one of the isomorphic subgraphs. After that, this subgraph is copied. The integration into the whole graph is done by collapsing the isomorphic subgraphs into two nodes whos sizes are given by the surrounding rectangles. It is possible now to draw the complete graph and unfold the nodes representing the isomorphic subgraphs afterwards.
Another possibility for drawing algorithms is applicable in springembedders where the position of nodes is determined by forces. Instead of moving only one node in every step, the forces of a mapped pair are determined and both nodes are moved. Thus, the isomorphic subgraphs are drawn identically. The advantage of this method is that the remainder is regarded during the determination of the layout of the isomorphic subgraphs in contrast to the copy-paste-method.

The integration of isomorphic subgraphs into drawing algorithms will also lead to some theoretical problems. One can easily find examples, such that there are no planar drawings of planar graphs, if the isomorphic subgraphs are drawn in the same way. Therefore, we will have to characterize those planar graphs which admit such planar drawings. It might be possible to adapt the results of Feng et al. ([5], [4])

Acknowledegment

I would like to thank Prof. F.J. Brandenburg for his various and detailed comments and suggestions, particularly on the proof of ISOMORPHIC SUBGRAPHS of general graphs.

References

[1] W. Bachl. Entwicklung und Implementierung eines Hierarchischen Springembedders. Diplomarbeit, Universität Passau, 1995.

[2] F. J. Brandenburg. Designing graph drawings by layout graph grammars. In *Proc. Graph Drawing 1994*, Lecture Notes in Computer Science 894, pages 416–427. Springer, 1995.

[3] F.J. Brandenburg and S. Wetzel. On pairs of large common subgraphs. unpublished manuscript, http://www.fmi.uni-passau.de/ wetzel/paper/, 1998.

[4] Qing-Wen Feng, Robert F. Cohen, and Peter Eades. How to draw a planar clustered graph. In Ding-Zhu Du and Ming Li, editors, *Computing and Combinatorics COCOON '95*, Lecture Notes in Computer Science 959, pages 21–30. Springer, 1995.

[5] Qing-Wen Feng, Robert F. Cohen, and Peter Eades. Planarity for clustered graphs. In Paul Spirakis, editor, *Algorithms-ESA '95*, Lecture Notes in Computer Science 979, pages 213–226. Springer, 1995.

[6] M.R. Garey and D.S. Johnson. *Computers and Intractability. A Guide to the Theory of NP-Completeness.* W.H. Freeman, 1979.

[7] D. Gusfield. *Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology.* Cambridge University Press, 1997.

[8] R.E. Ladner. On the structure of polynomial time reducibility. *Journal of the ACM*, 22:155–171, 1975.

[9] Giorgio Levi and Fabrizio Luccio. A technique for graph embedding with constraints on node and arc correspondences. *Information Sciences*, 5:1–24, 1973.

[10] Andrzej Lingas. Subgraph isomorphism for biconnected outerplanar graphs in cubic time. *Theoretical Computer Science*, 63:295–302, 1989.

[11] R. J. Lipton, S. C. North, and J. S. Sandberg. A method for drawing graphs. In *Proc. ACM Symposium on Computational Geometry*, pages 153–160. ACM, 1985.

[12] A. Lubiw. Some NP-complete problems similar to graph isomorphism. *SIAM J. Comput.*, 10(1):11–21, 1981.

[13] J. Manning. *Geometric Symmetry in Graphs.* PhD thesis, Department of Computer Science, Purdue University, 1990.

[14] J. Manning and M. J. Atallah. Fast detection and display of symmetry in trees. *Congressus Numerantium*, 64:159–168, 1988.

[15] J. Manning and M.J. Atallah. Fast detection and display of symmetry in outerplanar graphs. *Discrete Applied Mathematics*, 39:13–35, 1992.

[16] H. Purchase. Which aesthetic has the greatest effect on human understanding. In *Proc. Graph Drawing 1997*, Lecture Notes in Computer Science 1353, pages 248–261. Springer, 1997.

[17] H. C. Purchase, R. F. Cohen, and M. James. Validating graph drawing aesthetics. In *Proc. Graph Drawing 1995*, Lecture Notes in Computer Science 1027, pages 435–446. Springer, 1996.

[18] Ronald C. Read and Derek G. Corneil. The graph isomorphism disease. *Journal of Graph Theory*, 1:339–363, 1977.

[19] Steven W. Reyner. An analysis of a good algorithm for the subtree problem. *SIAM J. Comput.*, 6(4):730–732, 1977.

[20] U. Schöning. Graph isomorphism is in the low hierarchy. *Journal. Comput. Syst. Science*, 37:312–323, 1987.

[21] Maciej M. Syslo. The subgraph isomorphism problem for outerplanar graphs. *Theoretical Computer Science*, 17:91–97, 1982.

[22] E. Ukkonen. On-line construction of suffix-trees. In *Algorithmica 14*, pages 249–260, 1995.

[23] P. Weiner. Linear pattern matching algorithms. In *Proc. of the 14th IEEE Symp. on Switching and Automata Theory*, pages 1–11, 1973.

[24] F. Frances Yao. Graph 2-isomorphism is NP-complete. *Information Processing Letters*, 9(2):68–72, August 1979.