An Algorithm for Maximum Common Subgraph of Planar Triangulation Graphs

Yao Lu[1], Horst Bunke[2], and Cheng-Lin Liu[1]

[1] National Lab of Pattern Recognition
Institute of Automation, Chinese Academy of Sciences
yaolubrain@gmail.com, liucl@nlpr.ia.ac.cn
[2] Institute of Computer Science and Applied Mathematics (IAM),
University of Bern
bunke@iam.unibe.ch

Abstract. We propose a new fast algorithm for solving the Maximum Common Subgraph (MCS) problem. MCS is an NP-complete problem. In this paper, we focus on a special class of graphs, i.e. Planar Triangulation Graphs, which are commonly used in computer vision, pattern recognition and graphics. By exploiting the properties of Planar Triangulation Graphs and restricting the problem to connected MCS, for two such graphs of size n and m and their maximum common subgraph of size k, our algorithm solves the MCS problem approximately with time complexity $O(nmk)$.

Keywords: Planar Triangulation Graphs, Delauney Triangulation, Maximum Common Subgraph.

1 Introduction

Images and many other objects can be represented as graphs. The graph representation of an object characterizes its local features and their spatial relationship. Its theoretical properties, applications and efficient algorithms have been studied for decades [1]. Maximum Common Subgraph (MCS) is an important problem in pattern recognition [2]. It incorporates graph isomorphism and subgraph isomorphism as special cases. It has applications in computer vision and pattern recognition such as video indexing [3] and document classification [4]. However, MCS is known to be NP-complete in general. In order to obtain an efficient algorithm of practical value, we need to specialize the problem and/or look for approximate solutions.

In this paper, we focus on a special class of graphs, i.e. Planar Triangulation Graphs. Planar Triangulation Graphs are commonly used in pattern recognition, computer vision, and graphics. Perhaps the best known procedure to obtain such a graph is Delaunay triangulation. It has important properties such as sparseness, locality, and avoiding skinny angles in the triangulation. It has been shown that not much spatial information is lost after Delaunay triangulation of a set of points [5]. Moreover, Delaunay triangulation can be efficiently computed

W.G. Kropatsch et al. (Eds.): GbRPR 2013, LNCS 7877, pp. 162–171, 2013.

with time complexity $O(n \log n)$ by various methods [6]. And there are graph matching algorithms specialized for Delaunay triangulation graphs [7, 8].

The main contribution of in this paper is a new algorithm for the MCS problem of Planar Triangulation Graphs that has a practical high execution speed. By exploiting the properties of Planar Triangulation Graphs and restricting the problem to connected MCS, we are able to derive an algorithm with time complexity $O(nmk)$, for approximately solving the MCS problem of two Planar Triangulation Graphs of size n and m and their maximum common subgraph of size k. Given two graphs, our algorithm will return a common subgraph, but it is not guaranteed that it is a maximum common subgraph. However, our experimental verification showed that most of the common subgraphs returned in our experiments are in fact maximum common subgraphs, and those that are not are missing only a small fraction of nodes and edges.

2 Connected Maximum Common Subgraphs

2.1 Basic Definitions

Definition 1. *A graph is an ordered pair $G = (V, E)$ comprising a set V of nodes together with a set $E \subseteq V \times V$ of edges.*

Remark. The graphs we assume in this paper are unweighted and undirected graphs without node or edge attributes.

Definition 2. *A subgraph of a graph G is a graph whose node set is a subset of that of G, and whose adjacency relation is a subset of that of G restricted to this subset.*

Definition 3. *An isomorphism of graphs G and H is a bijective function between the node sets of G and H, $f : V(G) \to V(H)$ such that any two nodes u and v of G are adjacent in G if and only if $f(u)$ and $f(v)$ are adjacent in H.*

Definition 4. *A subgraph isomorphism from G to H is an injective function $f : V(G) \to V(H)$ such that if there exists a subgraph S of H and f is a graph isomorphism from G to S.*

Definition 5. *Let G, G_1, and G_2 be graphs. G is a common subgraph of G_1 and G_2 if there exists a subgraph isomorphisms from G to G_1 and from G to G_2.*

Definition 6. *A common subgraph G of G_1 and G_2 is maximal if there exists no other common subgraph G' of G_1 and G_2 that has more edges than G.*

Remark. This definition is given in [15] as Maximum Common Edge Subgraph. The work described in this paper will be based on this kind of MCS. It is preferred over Maximum Common Induced Subgraph for the reasons explained in [10, 15].

Definition 7. *A graph is connected if there is a path from any node to any other node in the graph.*

Finding connected MCS is in general NP-complete since subgraph isomorphism remains NP-complete even for connected graphs of bounded treewidth [9], which is a special case of connected MCS.

2.2 Related Work

Many exact algorithms for finding MCS have been proposed in the literature. Two early examples are [10] and [11], which are based on backtrack search and maximum clique detection in an association graph, respectively. Later algorithms are described in [12–14]. All these algorithms have exponential time complexity. Moreover, there are algorithms specialized for chemical structures [15], which can achieve faster solutions in this domain than general MCS algorithms.

In this paper, we specialize on Planar Triangulation Graphs, a class of graphs which are widely used in computer vision, pattern recognition and graphics.

3 Planar Triangulation Graphs

A Planar Triangulation Graph is obtained by triangulation of points in a plane. See [6, 16] for references on triangulation. In Fig.1, we show an example obtained by Delaunay triangulating a set of points in the two-dimensional plane.

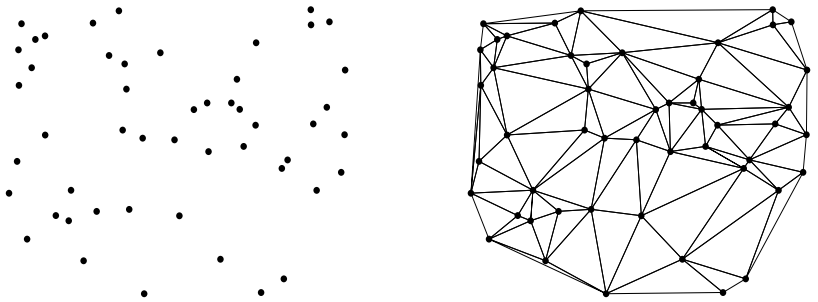


Fig. 1. Delaunay Triangulation of a set of points in the two-dimensional plane

3.1 Properties

Planar triangulation graphs have two important properties [6].

Property 1. *Each triangle of a Planar Triangulation Graph has at most three adjacent triangles.*

Property 2. *A Planar Triangulation Graph of size n has $O(n)$ triangles and $O(n)$ edges.*

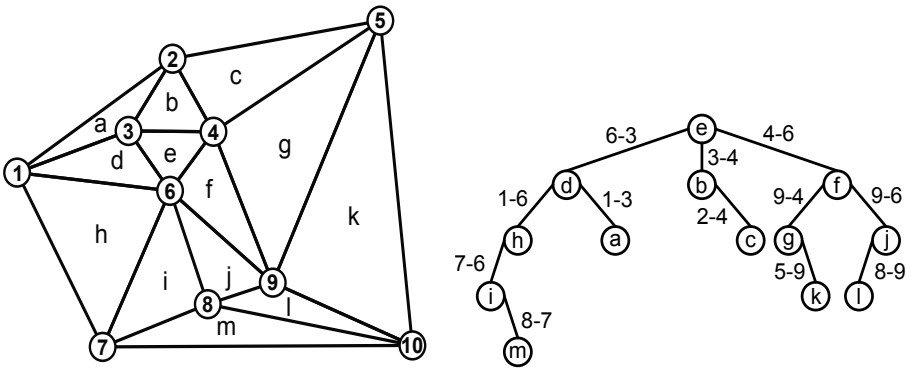
These properties allow us to design a fast algorithm for connected MCS of Planar Triangulation Graphs, as will be shown in the following sections.

3.2 Breadth-First Traversal of Triangles

In this subsection, we describe a method for traversing the triangles of a Planar Triangulation Graph. Note that it is a traversal of *triangles* rather than of *nodes*.

Starting from a given ordered triangle of a Planar Triangulation Graph with a triangle visiting order (e.g. clock-wise), there is a unique breadth-first traversal of triangles of the graph. The breadth-first traversal process works as follows. For example, in Fig. 2(a), let us take triangle **e** (with order **6-3-4**) as the root of the traversal. Then the adjacent triangles of the root triangle are visited in clock-wise order. From edge **6-3**, triangle **d** is visited, from edge **3-4**, triangle **b** is visited, and from edge **4-6**, triangle **f** is visited. Then we continue the process with triangle **d**, and so on. Except for the root triangle, for each visit of a triangle, at most one new node is encountered. After the traversal is finished, the visiting order of nodes **1** to **10** is: 4,5,2,3,8,1,7,9,10. The breadth-first traversal of triangles can be represented as a tree, as shown in Fig. 2(b). By property 1, this tree has a special structure: it has at most 3 children at its root node and at most 2 children at the other nodes.

Given a triangle, its adjacent triangles can be found by using a hash table with $O(1)$ operations. And the hash table can be built with time complexity $O(n)$ since there are only $O(n)$ triangles and each triangle has at most three adjacent triangles. Therefore, for a Planar Triangulation Graph of size n, the breadth-first traversal of triangles has time complexity $O(n)$.



(a) Planar Triangulation Graph (b) Tree structure of the traversal of triangles

Fig. 2. Breadth-First Traversal of Triangles

With these special properties and structures, we can derive an efficient heuristic algorithm for finding connected MCS of a given pair of Planar Triangulation Graphs, as shown in the next section.

4 Algorithm

The algorithm works as follows: starting with an arbitrarily chosen pair of ordered triangles as root in graphs G_1 and G_2, do a pair of breadth-first traversals of triangles on both graphs simultaneously. The visiting orders of the nodes of the initial pair triangles are 1, 2, 3. Two triangles of two graphs are matched and added to their traversal if their corresponding nodes have the same node visiting order. If a node has not been visited in the traversal yet, its node visiting order is set to be 0. The process continues until no triangles can be added to the pair of traversals. Finally, the graphs composed of the matched triangle pairs are the MCS. A pseudo-code description of our algorithm is given in Algorithm 1.

As an example, consider finding the MCS of the graphs in Fig. 3(a). In Fig. 3(b), we start with ordered triangles **2-3-4** and **b-c-e** as the roots of the pair of traversals. The visiting orders of nodes **2**, **3**, **4** and nodes **b**, **c**, **e** are 1, 2, 3, respectively. Triangles **2-3-4** and **b-c-e** are matched and added to the traversal since their corresponding nodes have the same node visiting order (all 0 now and 1,2,3 after). From ordered triangle **2-3-4**, triangles **2-3-1**, **3-4-6** and **4-2-5** are visited, in that order. And from ordered triangle **b-c-e**, triangles **b-c-a**, **c-e-f** and **e-b-d** are visited, in that order. Next, triangles **2-3-1** and **b-c-a** are matched since the the condition is satisfied, then triangles **3-4-6** and **c-e-f**, and finally triangles **4-2-5** and **e-b-d** (Fig. 3(c)). However, triangle **4-5-6** cannot be matched to either **d-e-g** or **e-f-g** because node **6**'s node visiting order is 6 and node **g**'s node visiting order is 0. Finally, as shown in Fig. 3(d), the graphs composed of **2-3-4**, **2-3-1**, **3-4-6** and **4-2-5** and of **b-c-e**, **b-c-a**, **c-e-f** and **e-b-d** are the MCS of the two graphs in Fig. 3(a).

For each such a pair of traversals, it takes $O(k)$ operations assuming the maximum common subgraph of size k. To find the MCS, we have to consider all pairs of ordered triangles as the roots of the pairs of the traversals. By Property 2, there are $O(nm)$ pairs of ordered triangles in total. Consequently, if we take all pairs of ordered triangles as the roots of the traversals, there are $O(nm)$ pairs of traversals. Hence, the overall time complexity of the algorithm is $O(nmk)$.

At first glance, our algorithm looks similar to String Growing algorithm for subgraph isomorphism [17]. But there are two main differences: (1) our algorithm is specialized for Planar Triangulation Graphs while String Growing algorithm is specialized for Region Adjacent Graphs. (2) Our algorithm has worst case time complexity $O(nmk)$ while String Growing algorithm has worst case exponential time complexity.

5 Experiments

In the experiments, n random points in the two-dimensional plane were generated to obtain point set S_1 and triangulated by Delaunay triangulation to obtain graph G_1. m points around the center of S_1 were selected to obtain point set S_2 and then triangulated by Delaunay triangulation to obtain graph G_2. Due to the boundary effect of Delaunay triangulation, G_2 is not necessarily a subgraph of G_1 in general.

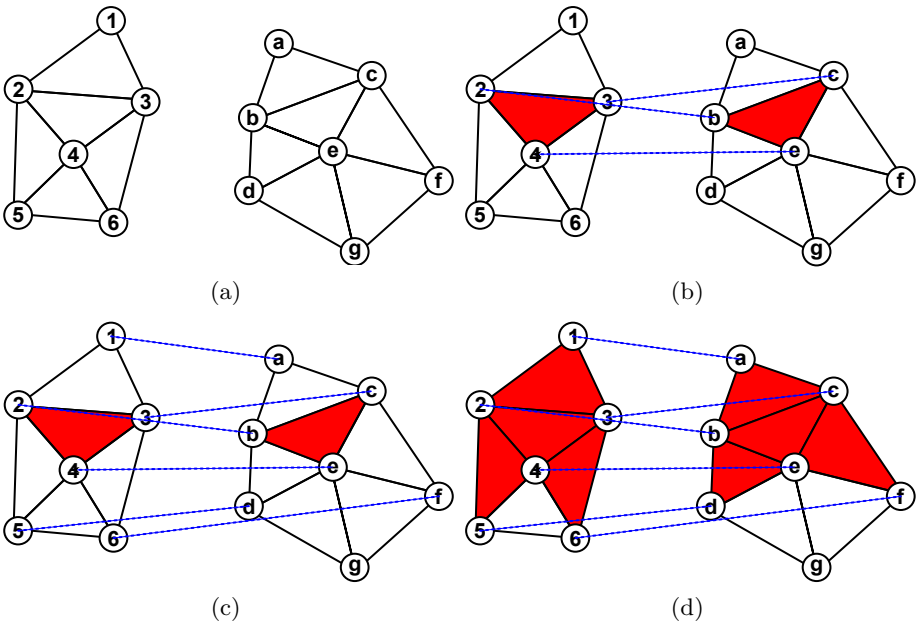


Fig. 3. Illustration of the algorithm

Algorithm 1. MCS of Planar Triangulation Graphs

```
1  foreach Pair of ordered triangles (T_i,T_j) do
2      Initialize empty set M
3      Initialize empty queues Q_1 and Q_2
4      Enqueue(Q_1,T_i)
5      Enqueue(Q_2,T_j)
6      while Q_1 and Q_2 are not empty do
7          T_1 = Dequeue(Q_1)
8          T_2 = Dequeue(Q_2)
9          M = M ∪ {(T_1,T_2)}
10         foreach Pair of triangles (T_1^adj,T_2^adj) adjacent to (T_1,T_2) and not in M
           do
11             if their corresponding nodes have the same node visiting order then
12                 Enqueue(Q_1,T_1^adj)
13                 Enqueue(Q_2,T_2^adj)
14             end
15         end
16     end
17     Record M with its cardinality
18 end
19 return M with the maximum cardinality
```

The graph data was generated in MATLAB. The algorithm is implemented in C++. The experiments were run with Intel Core i5-2400 3.1GHz CPU and 4GB RAM and with a single thread.

5.1 Small Random Graphs

In this set of experiments, our algorithm is compared with an exact MCS algorithm. The exact algorithm works by constructing an association graph and finding the maximum clique of it, as in [11, 12]. The maximum clique corresponds to the MCS. The association graphs were built according to [12]. The maximum clique algorithm we used is a Branch-and-Bound method [18], due to its relatively high speed (still exponential time complexity) and efficient implementation. McGregor's algorithm [10] requires that every node of the smaller graph must be matched to some node of the larger graph. Such requirement is not always satisfied in practice. Therefore, McGregor's algorithm is not included in the comparison. The sizes of graphs range only from 5 to 20, due to the high computational costs of the exact MCS algorithm. Three cases of MCS testing experiments were conducted: 20 nodes vs. 5 nodes, 20 nodes vs. 10 nodes, and 20 nodes vs. 15 nodes. The number of edges of the common subgraphs and runtime of two algorithms averaged over 20 trials are shown in Table 1 and Table 2, respectively. Again, due to the high computational costs of the exact algorithm, only 20 trials in each case were conducted.

Table 1. Average edges of MCS dependent on graph size (nodes)

Algorithm	20 vs. 5	20 vs. 10	20 vs. 15
Exact	7.5	20.25	32.95
Ours	7.4	19.5	32.7

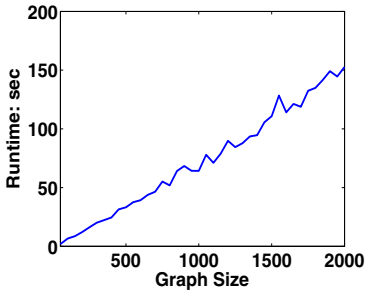
Table 2. Runtime (sec) dependent on graph size (nodes)

Algorithm	20 vs. 5	20 vs. 10	20 vs. 15
Exact	0.002	6.35	2202.123
Ours	0.005	0.032	0.087

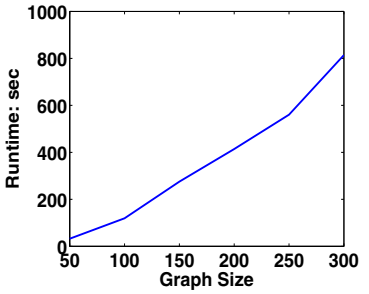
5.2 Large Random Graphs

In this set of experiments, the performance of our algorithm in finding MCS of relatively large graphs is shown. We vary the size of G_1 and G_2 to record the runtime of our algorithm in Fig. 4. The size of graphs range from 50 nodes to 2000 nodes. See Fig. 5 for visualization.

In Fig. 4(a), the size of G_2 is kept constant at 50 nodes, while the size of G_1 varies from 50 to 2000 nodes. In contrast, in Fig. 4(b), the size of G_2 varies from 50 to 300 nodes at constant size of G_1 at 500 nodes. The computation time



(a) Size of G_2 is constantly equal to 50 nodes, while the size of G_1 is varied from 50 to 2000 nodes.



(b) Size of G_1 is constantly equal to 500 nodes, while the size of G_2 is varied from 50 to 300 nodes.

Fig. 4. Runtime (sec) dependent on graph size (nodes)

measured in these experiments (averaged over 10 trials) confirms the theoretical complexity mentioned in Section 4. In Fig. 4(a), a linear increase of the computation time in terms of the size of G_1 of is observed, while the behavior in Fig. 4(b) is superlinear in the size of G_2 since the increase of the size of the smaller graph would also increase the size of the maximum common subgraph.

To the knowledge of the authors, there exists no other specialized algorithm for computing connected MCS of Planar Triangulation Graphs. Therefore there exists no direct competitor against which the proposed algorithm could be evaluated. Of course one could benchmark the new algorithm against other algorithms that were developed for general graphs. Here we note, however, that the algorithms in [10, 12–14] were tested on much smaller graphs ($\leq$ 100 nodes) than the ones considered in this set of experiments. Therefore, it is computationally prohibitive to run comparison experiments.

6 Conclusion and Future Work

We present a fast algorithm for approximately solving the MCS problem of Planar Triangulation Graphs. In its present version, the algorithm can only cope with unweighted graphs without node attributes. However, it is straightforward to include weights on edges and attributes on nodes. Theoretical analysis on the quality of the approximation and more systematic experimental comparison with other MCS algorithms, such as one using approximate maximum clique detection methods [19], will be explored in the future. Also the application of the proposed algorithm to Planar Triangulation Graphs obtained from real images will be an interesting topic to be explored in future research.

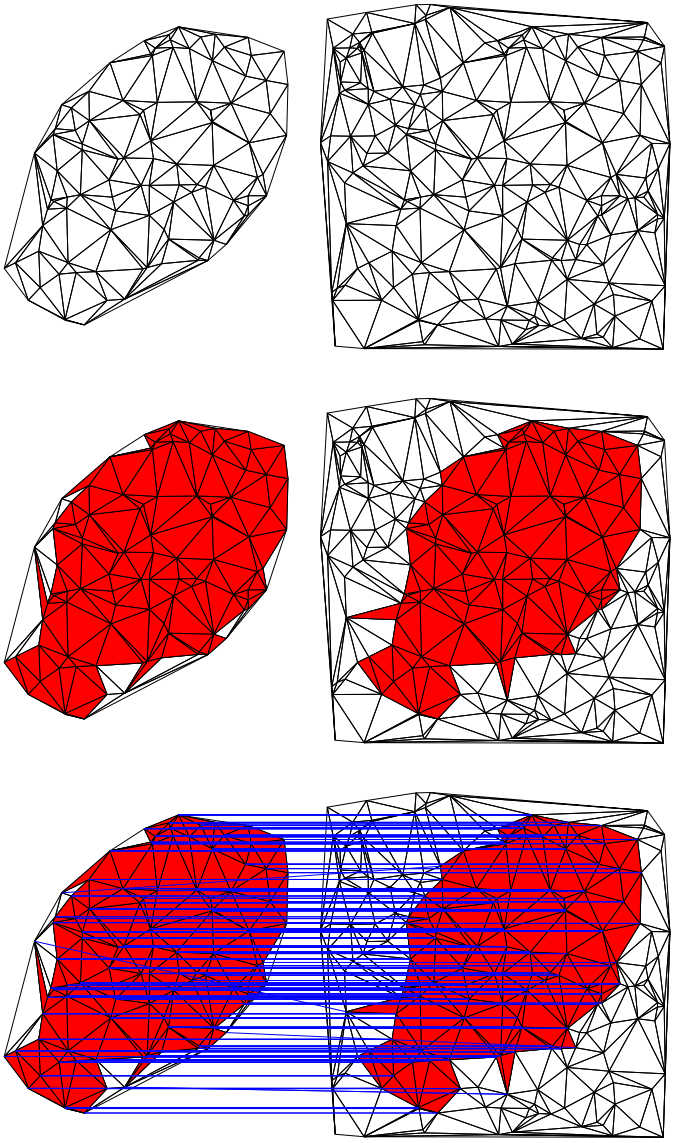


Fig. 5. 100 nodes vs. 200 nodes: 39 sec

References

1. Conte, D., Foggia, P., Sansone, C., Vento, M.: Thirty Years of Graph Matching in Pattern Recognition. International Journal of Pattern Recognition and Aritificial Intelligence 18(3), 265–298 (2004)
2. Bunke, H., Shearer, K.: A Graph Distance Metric Based on the Maximal Common Subgraph. Pattern Recognition Letters 19(3), 255–259 (1998)
3. Shearer, K., Bunke, H., Venkatesh, S.: Video Indexing and Similarity Retrieval by Largest Common Subgraph Detection Using Decision Trees. Pattern Recognition 34(5), 1075–1091 (2001)
4. Schenker, A., Last, M., Bunke, H., Kandel, A.: Classification of Web Documents Using Graph Matching. International Journal of Pattern Recognition and Artificial Intelligence 18(03), 475–496 (2004)
5. Dobkin, D., Friedman, S., Supowit, K.: Delaunay Graphs Are Almost as Good as Complete Graphs. Discrete and Computational Geometry 5(4), 399–407 (1990)
6. Berg, M., Cheong, O., Kreveld, M., Overmars, M.: Computational geometry: algorithms and applications. Springer (2008)
7. Finch, A., Wilson, R., Hancock, E.: Matching Delaunay Graphs. Pattern Recognition 30(1), 123–140 (1997)
8. Shin, D., Tjahjadi, T.: Similarity Invariant Delaunay Graph Matching. In: da Vitoria Lobo, N., Kasparis, T., Roli, F., Kwok, J.T., Georgiopoulos, M., Anagnostopoulos, G.C., Loog, M. (eds.) SSPR & SPR 2008. LNCS, vol. 5342, pp. 25–34. Springer, Heidelberg (2008)
9. Matouěk, J., Thomas, R.: On the Complexity of Finding Iso-and other morphisms for Partial k-trees. Discrete Mathematics 108(1), 343–364 (1992)
10. McGregor, J.: Backtrack Search Algorithms and the Maximal Common Subgraph Problem. Software Practice and Experience 12(1), 23–34 (1982)
11. Levi, G.: A Note on the Derivation of Maximal Common Subgraphs of Two Directed or Undirected Graphs. Calcolo 9(4), 341–352 (1972)
12. Koch, I.: Enumerating All Connected Maximal Common Subgraphs in Two Graphs. Theoretical Computer Science 250(1), 1–30 (2001)
13. Bunke, H., Foggia, P., Guidobaldi, C., Sansone, C., Vento, M.: A Comparison of Algorithms for Maximum Common Subgraph on Randomly Connected Graphs. In: Caelli, T.M., Amin, A., Duin, R.P.W., Kamel, M.S., de Ridder, D. (eds.) SSPR & SPR 2002. LNCS, vol. 2396, pp. 123–132. Springer, Heidelberg (2002)
14. Conte, D., Foggia, P., Vento, M.: Challenging Complexity of Maximum Common Subgraph Detection Algorithms: A Performance Analysis of Three Algorithms on a Wide Database of Graphs. Journal of Graph Algorithms and Applications 11(1), 99–143 (2007)
15. Raymond, J., Willett, P.: Maximum Common Subgraph Isomorphism Algorithms for the Matching of Chemical Structures. Journal of Computer-Aided Molecular Design 16(7), 521–533 (2002)
16. Bern, M., Eppstein, D.: Mesh Generation And Optimal Triangulation. Computing in Euclidean Geometry 1, 23–90 (1992)
17. Lladós, J., Martí, E., Villanueva, J.: Symbol Recognition by Error-Tolerant Subgraph Matching between Region Adjacency Graphs. IEEE Trans. Pattern Analysis and Machine Intelligence 23(10), 1137–1143 (2001)
18. Konc, J., Janezic, D.: An Improved Branch and Bound Algorithm for the Maximum Clique Problem. Communications in Mathematical and in Computer Chemistry/MATCH 58(3), 569–590 (2007)
19. Pelillo, M., Torsello, A.: Payoff-Monotonic Game Dynamics and the Maximum Clique Problem. Neural Computation 18(5), 1215–1258 (2006)