A Performance Comparison of Five Algorithms for Graph Isomorphism

P. Foggia, C.Sansone, M. Vento
Dipartimento di Informatica e Sistemistica
Via Claudio, 21 - I 80125 - Napoli, Italy
{foggiapa, carlosan, vento}@unina.it

Abstract
Despite the significant number of isomorphism algorithms presented in the literature, till now no efforts have been done to characterize their different performance in terms of matching time. It is not clear how the behavior of those algorithms varies as the type and the size of the graphs to be matched varies in case of real applications.
In this paper we present a benchmarking activity for characterize the performance of a bunch of algorithms for exact isomorphism: to this purpose we use a database of graphs specifically developed for this task.

1. Introduction

The exact graph matching problem is of interest in a variety of different pattern recognition contexts. In fact, graphs are used to support structural descriptions as well as for low level image representations.
As it is well known, among the different types of graph matching subgraph isomorphism is a NP-complete problem [10], while it is still an open question if also graph isomorphism is a NP-complete problem. As a consequence, time requirements of brute force matching algorithms increase exponentially with the size of the input graphs, restricting the applicability of graph based techniques to problems implying graphs with a small number of nodes and edges.
Therefore, algorithms having time requirements suited for matching large graphs, have been a subject of research during the last three decades.
Some of them reduce the computational complexity of the matching process by imposing topological restrictions on the graphs. We can mention algorithms for finding the isomorphism between planar graphs [11], trees [1] or, more generally, bounded valence graphs [14].
An alternative approach to reducing matching complexity is that of using an adequate representation of the searching process and pruning unprofitable paths in the search space. In this way, no restrictions must be imposed on the structure of the input graphs and the obtained algorithms can be generally applied.
One of the pioneer papers ascribable to this area [6], illustrates an isomorphism algorithm which performs suitable transformations on the input graphs, in order to

find a different representation for which the matching is computationally more convenient. However, it has been shown [15] that the conjecture on which this method is based is not always true.

A procedure that significantly reduces the size of the search space is the backtracking algorithm proposed by Ullmann in [21]. This algorithm is devised for both graph isomorphism and subgraph isomorphism and is still today one of the most commonly used for exact graph matching. This is confirmed by the fact that in Messmer [16] it is compared with other algorithms and it results the more convenient in terms of matching time in case of one-to-one matching.

Another backtracking algorithm is the one presented in [20] by Schmidt and Druffel. It uses the information contained in the distance matrix representation of a graph to establish an initial partition of the graph nodes. This distance matrix information are then used in a backtracking procedure to reduce the search tree of possible mappings.

A more recent algorithm, known as VF, is based on a depth-first search strategy, with a set of rules to efficiently prune the search tree. Such rules in case of isomorphism are shown in [5].

Another possible approach to the isomorphism problem is the one presented in [2]; instead of reducing the complexity of matching two graphs, the authors attempt to reduce the overall computational cost when matching a sample graph against a large set of prototypes. The method performs the matching in quadratic time with the size of the input graph and independently on the number of prototypes. It is obviously convenient in applications requiring the matching of a graph against a database, but the memory required to store the pre-processed database grows exponentially with the size of the graphs, making the method suitable only for small graphs. So one of the authors concludes in [16] that in case of one-to-one matching other algorithms (in particular, in [9] the Ullmann's one is cited) are more suitable.

As regards graph isomorphism algorithms, it is also necessary to mention the McKay's Nauty algorithm [17]. It is based on a set of transformations that lead to reduce the graphs to be matched to a canonical form for which is very simple to test the presence of an isomorphism. Even if Nauty is considered to be one of the fastest graph isomorphism algorithm available, it has been shown that there are categories of graphs for which it employs exponential time in order to find an isomorphism [19].

All the above cited algorithms are exact ones, i.e. they find the exact solution to the matching problem, if any. Besides them, other techniques (as those based on non-deterministic paradigms [4, 7, 13]), able to find approximate solutions to the matching problem have been proposed, especially in the recent past. We do not explicitly consider them in this paper, since they are really so powerful to reduce the complexity (in most cases) from exponential to polynomial, but, as said, they do not guarantee finding an exact and optimal solution.

Despite the significant number of algorithms presented in the literature, till now almost no efforts have been done to characterize their different performance in terms of matching time. It is not clear the behavior of those algorithms as the type and the size of the graphs to be matched varies in case of real applications.

Some preliminary work has been made in [12] as regards the comparison of inexact graph matching algorithms for Attributed Relational Graphs. However, in this paper the authors use a database made by graphs very limited in size (input graph size ranged from 3 to 9 nodes) and so the obtained results cannot be considered useful for guiding the algorithm's choice in most real applications.

As a consequence, users of graph-based approaches can only utilize qualitative criteria to select the algorithm that seems to better fit their application constraints. So the need of a benchmarking activity arises higher and higher in the Pattern Recognition community, as stated in [3].

In order to made a first step towards the attempt of filling this lack, in this paper we present a benchmarking activity for characterize the performance of a bunch of algorithms for exact isomorphism, and to this purpose we use a database of graphs specifically developed for this task.

The paper is organized as follows: in the next section the algorithms chosen for the comparison are briefly presented; and in section 3 the used database is described. In section 4 we report, for the different categories of graphs of the database the results of the performance analysis. Finally, a discussion highlighting the behavior of the considered algorithms is reported, and some future directions for the benchmarking activity are drawn.

2. Algorithms for Benchmarking

This first benchmarking activity has been carried out on those exact matching algorithms that do not impose particular restrictions on the structure of the input graphs. This category include the Ullmann's algorithm, the algorithm of Schmidt and Druffel (in the following referred as SD), the VF algorithm and the Nauty algorithm. We consider two versions of the VF algorithm: the first one, referred as VF only is based on the implementation reported in [5], while the second one, referred as VF2, uses more effective data structures in order to optimize the matching time. Details on this kind of implementation can be found in [8]. We explicit choose of not considering algorithm that do not guarantee to find an exact solution, as the Corneil and Gottlieb algorithm and all those based on non-deterministic approaches. Also the algorithms proposed by Messmer and Bunke [16, 18] are not considered, as they optimize the one-to-many matching problem, so resulting disadvantaged in comparison with a generic one-to-one matching algorithm.

3. The used Database

The used database was made up of 10,000 couples of isomorphic graphs: it is part of a wider database of synthetically generated graphs, especially developed for benchmarking purposes, and described in details in [9].

In particular, the following kinds of graphs have been considered:

- ***Randomly connected graphs*** (3000 couples);
- ***Regular 2D meshes*** (1000 couples);
- ***Irregular 2D meshes*** (3000 couples); (see after for the meaning of irregular)
- ***Bounded valence graphs*** (3000 couples).

Each category contains couples of graphs of different size, ranging from few dozens to about 1000 nodes (i.e., *small* and *medium* size graphs according to [3]); for each size 100 different couples have been generated.

As regards the *randomly connected graphs*, three different values of the edge density η has been considered (0.01, 0.05 and 0.1) and 1000 couples of graphs of different size have been generated for each value of η. The density η is defined as the probability that an edge is present between two distinct nodes n and n'. The higher the value of η, the more the graph is dense.

An *irregular 2D mesh* has been obtained from a *regular 2D mesh* by the addition of a certain number of edges, each connecting nodes that have been randomly determined according to an uniform distribution. The number of added edges was ρN, where ρ is a constant greater than 0. Three values of ρ has been considered (0.2,0.4 and 0.6) and 1000 couples of graphs of different size generated for each value of ρ.

Finally for *bounded valence graphs*, three different value of the valence v has been considered and, once again, 1000 couples of graphs of different size have been generated for each value of v.

4. Experimental Results

All the considered algorithms were implemented in C++ and run on a Intel Celeron 500 Mhz PC, equipped with 128 MB of RAM. In particular we have implemented both the Ullmann and the SD algorithm on the basis of the papers [21] and [20] respectively. Their source code can be found at the Web site *http://amalfi.dis.unina.it/graph* together with the two version of the VF algorithm, namely VF and VF2. As regards the Nauty algorithm, we used the version 2.0b9 made available by B.D. McKay at the URL: *http://cs.anu.edu.au/~bdm/nauty/*

In the following of the Section the plots giving the matching times as a function of the input graphs size are shown for the five considered algorithms. Times are always reported in seconds in a logarithmic scale

Before presenting them, it is worth noting that some curves do not report the matching time obtained in correspondence with a given size. It happens when the algorithm was not able to find a solution to the isomorphism problem in a time less than half an hour.

4.1 Randomly Connected Graphs

Fig.1 shows the matching time of the five selected algorithms with reference to the *randomly connected graphs*. In particular Fig 1a, 1b and 1c respectively refers to values of η equal to 0.01, 0.5 and 0.1.

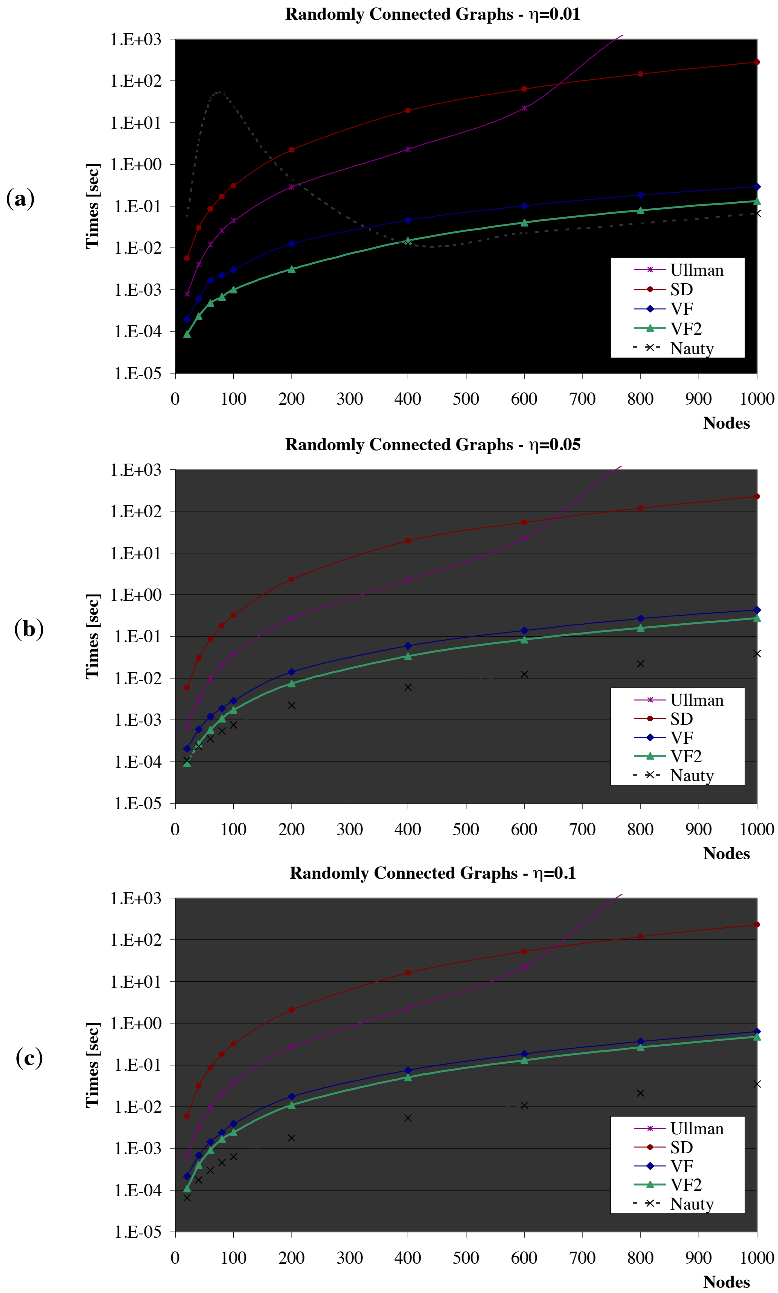

Fig. 1: The performance of the five algorithms on *Randomly Connected Graphs,* as a function of the graph size and for different values of η: (**a**) η= 0.01, (**b**) η= 0.05, (**c**) η= 0.1

It can be noted that the two version of VF and the algorithm of Nauty performs always better with respect to SD and Ullmann.
Moreover, VF2 performs always better than VF, while Ullmann is better than SD if the size of the graphs is smaller than 700. After this size, in fact, the Ullmann algorithm is not able to find any isomorphism.
In conclusions, the VF2 algorithm obtain the best performance for graphs of small size and for quite sparse graphs, while for dense graphs the Nauty algorithm obtains the best results.

4.2 2D Meshes

In Fig.2 the performance of the five algorithms on 2D regular meshes are shown.

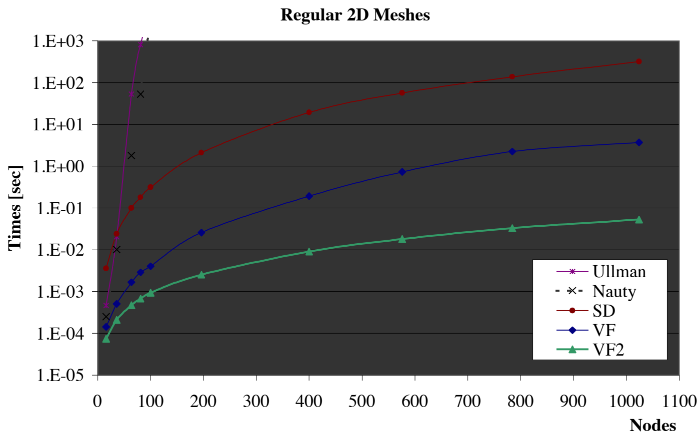

Fig. 2: The performance of the five algorithms on *Regular 2D Meshes* as a function of the graph size.

In this case, as the size of the graphs grows up to one hundred nodes, i.e. for graphs of medium size, both Nauty and Ullmann are not able to find solutions. For any input graph size, the VF2 algorithm is the best one. Moreover, note that the algorithm VF always performs better than SD.
Fig.3 reports the performance of all the algorithms on irregular 2D meshes. In particular, in Fig 3a 2D Meshes with ρ= 0.2 are considered, while in fig. 1b and 1c the considered values of ρ are 0.4 and 0.6 respectively.
The main difference with the case of regular meshes is that the Nauty algorithm is now always able to find a solution, but it still performs worse than VF2. However its behavior is better than those of VF and SD
Moreover, the maximum graph size for which the Ullmann algorithm is still able to find a solution grows proportionally to the irregularity of the graphs. For a value of ρ equal to 0.6, Ullmann can solve the isomorphism problem for graphs with size up to 500 nodes. If it finds a solution the matching time is always better than the one obtained by SD.

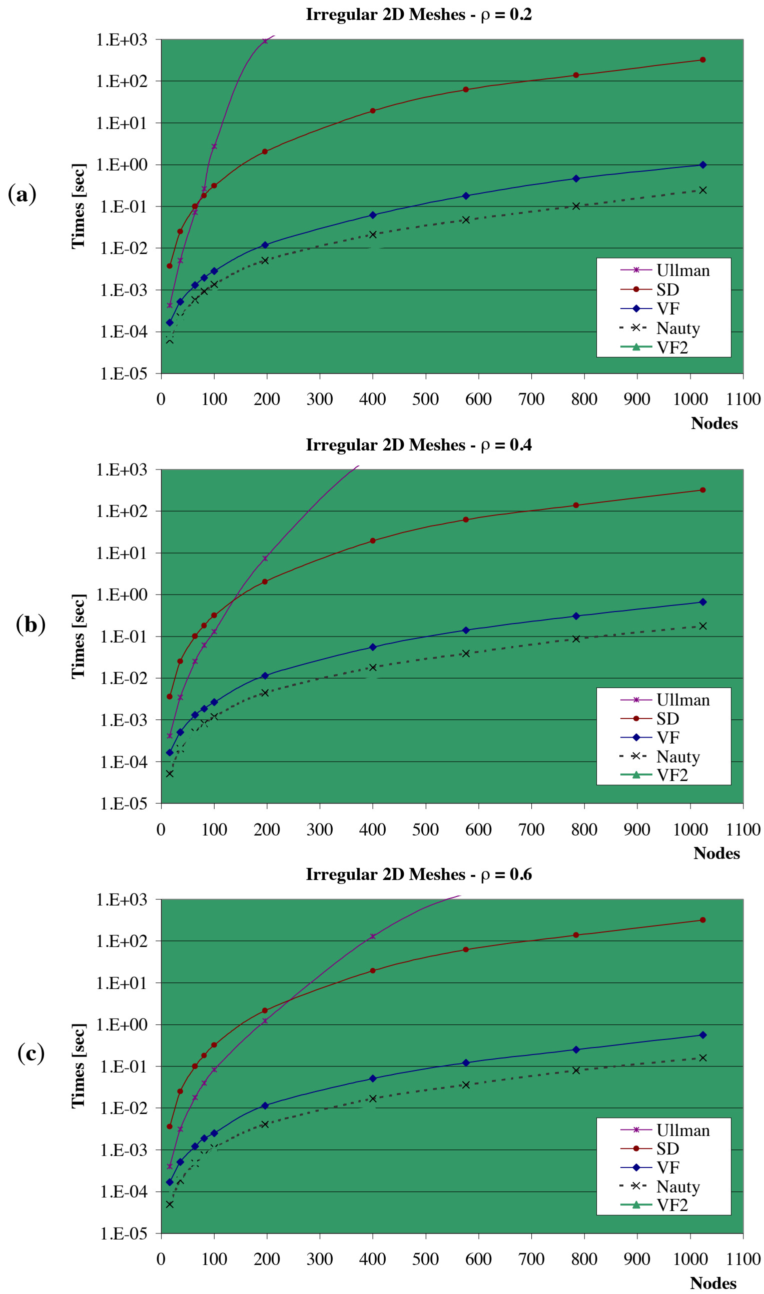

Fig. 3: The performance of the algorithms on *Irregular 2D Meshes* as a function of the graph size and for different values of ρ: (**a**) ρ= 0.2,(**b**) ρ= 0.4 (**c**) ρ= 0.6.

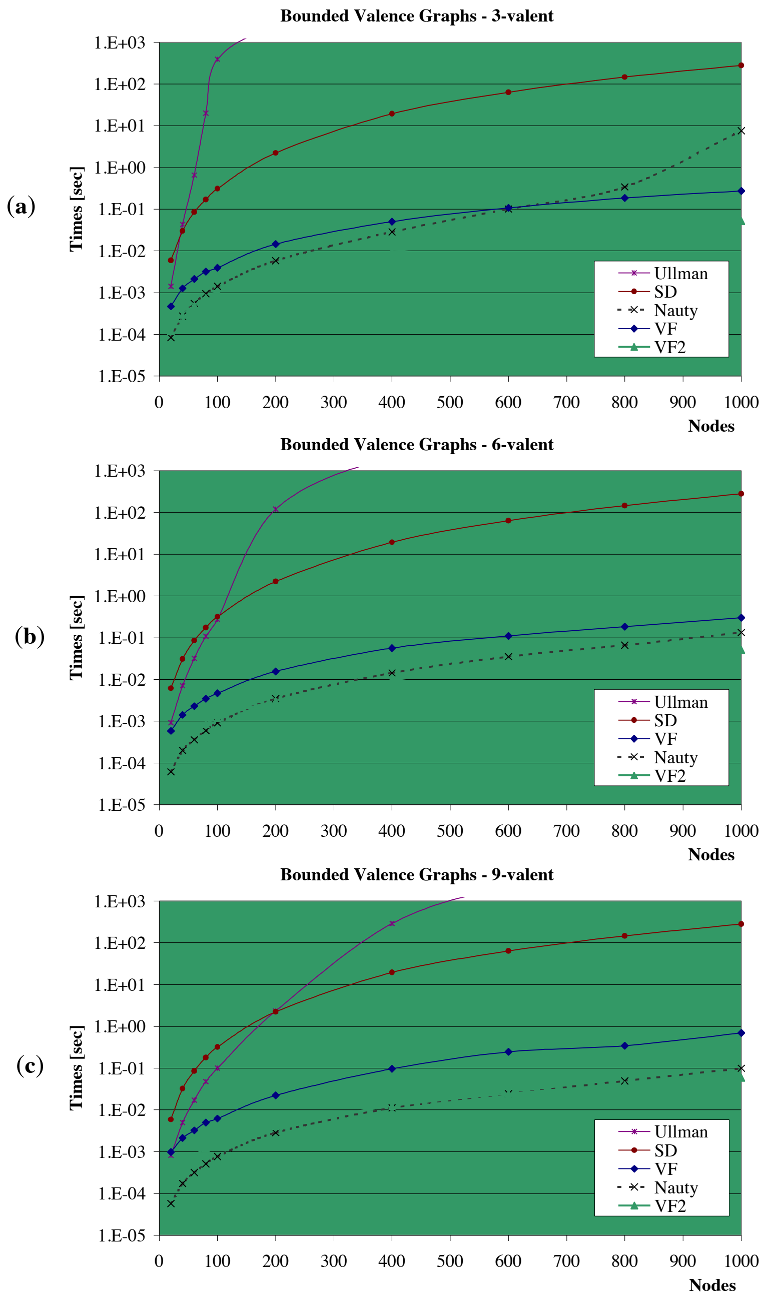

Fig. 4: The performance of the algorithms on *Bounded Valence Graphs* as a function of the graph size and for different values of the valence v: (**a**) $v = 3$, (**b**) $v = 6$, (**c**) $v = 9$.

4.3 Bounded Valence Graphs

Finally, in Fig.4 the performance of the algorithms on *bounded valence graphs* are shown. In this case the considered values of the valence *v* are 3, 6 and 9, as reported in Fig. 4a, 4b and 4c respectively.
Also in this case the Ullmann algorithm is not always able to find a solution; if it happens, however, its matching time is smaller than the one of SD and higher than those obtained by the other three algorithms. SD performs always worse than VF, VF2 and Nauty, while VF is always worse than VF2.
As regards the comparison between VF2 and Nauty, the first algorithm performs always better for a value of *v* equal to 3 (in this case, for quite large graphs, VF also is better than Nauty), while for higher values of *v* the matching times obtained by Nauty for quite small graph are smaller than those obtained by VF2. Anyway, independently of the considered value of *v*, as the number of the nodes of the input graphs is bigger than 600, VF2 is the best algorithm.

5. Discussion and conclusions

In this paper we have presented a preliminary benchmarking activity for assessing the performance of some widely used exact graph matching algorithms. The comparison has been carried out on a database of artificially generated graph.
As it could be expected, it does not exist an algorithm that is definitively better than all the others.
In particular, for randomly connected graphs, the Nauty algorithm is the better if the graphs are quite dense and/or of quite large size. For smaller and quite sparse graphs, on the contrary, VF2 performs better.
On more regular graphs, i.e. on 2D meshes, VF2 is definitely the best algorithm: in this case the Nauty algorithm is even not able to find a solution for graphs bigger than few dozens of nodes.
In case of bounded valence graph, if the valence is small, VF2 is always the best algorithm, while for bigger values of the valence the Nauty algorithm is more convenient if the size of the graphs is small.
Finally, it is also worth noting that SD, VF and VF2 are the only algorithms that have always been able to find a solution to the isomorphism problem in our tests, independently of the size and the kind of the graphs to be matched.
Future steps of this benchmarking activity will involve the comparison of other categories of graph matching algorithms, and the extension to other kind of matching problems, as the monomorphism and the graph-subgraph isomorphism.

6. References

[1] A.V. Aho, J.E. Hopcroft, J.D. Ullman, *The design and analysis of computer algorithms*, Addison Wesley, 1974.

[2] H. Bunke, B.T. Messmer, Efficient Attributed Graph Matching and its Application to Image Analysis, in *Image Analysis and Processing*. (C. Braccini, L. De Floriani, G.

Vernazza, eds), Springer, Berlin Heidelberg, pp. 45-55, 1995.

[3] H. Bunke, M.Vento, Benchmarking of Graph Matching Algorithms Proc. 2nd IAPR-TC15 Workshop on Graph-based Representations, Handhorf, 1999.

[4] W.J. Christmas, J. Kittler, and M. Petrou, Structural Matching in Computer Vision Using Probabilistic Relaxation, *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 17, no. 8, pp. 749-764,1995.

[5] L.P. Cordella, P. Foggia, C. Sansone, M. Vento, Evaluating Performance of the VF Graph Matching Algorithm, *Proc. of the 10th International Conference on Image Analysis and Processing*, IEEE Computer Society Press, pp. 1172-1177, 1999.

[6] D.G. Corneil, C.C. Gotlieb, An efficient algorithm for graph isomorphism, *Journal of the Association for Computing Machinery*, 17, pp. 51-64, 1970.

[7] K.A. De Jong, W.M. Spears, Using genetic algorithms to solve NP-complete problems, in *Genetic Algorithms*, (J.D. Schaffer, ed.), Morgan Kaufmann, Los Altus, CA., pp. 124-132, 1989.

[8] P. Foggia C. Sansone, M.Vento, An Improved Algorithm for Matching Large Graphs, submitted for publication to the *3rd IAPR-TC-15 International Workshop on Graph-based Representations*, Italy, 2001.

[9] P. Foggia, C. Sansone, M.Vento, A Database of Graphs for Isomorphism and Sub-Graph Isomorphism Benchmarking, submitted for publication to *the 3rd IAPR TC-15 International Workshop on Graph-based Representations*, Italy, 2001.

[10] M.R. Garey, D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, Freeman & co., New York, 1979.

[11] J. Hopcroft, J. Wong, Linear time algorithm for isomorphism of planar graphs, *Proc. 6th Annual ACM Symp. Theory of Computing*, pp. 172-184, 1974.

[12] H. Kälviäinen E. Oja, Comparisons of Attributed Graph Matching Algorithms for Computer Vision, Tech. Report. 1990.

[13] P. Kuner, B. Ueberreiter, Pattern recognition by graph matching - combinatorial versus continuous optimization, *Int. J. Pattern Recognition and Artif. Intell.*, vol. 2, no. 3, pp. 527-542, 1988.

[14] E. M. Luks, Isomorphism of Graphs of bounded valence can be tested in polynomial time, *Journal of Computer System Science*, pp. 42-65, 1982.

[15] R. Mathon, Sample graphs for isomorphism testing, *Congressus Numerantium, 21*, pp. 499-517, 1978.

[16] B. T. Messmer, *Efficient Graph Matching Algorithms for Preprocessed Model Graphs*, Ph.D. Thesis, Institute of Computer Science and Applied Mathematics, University of Bern, 1996.

[17] B.D. McKay, Practical Graph Isomorphism, *Congressus Numerantium, 30*, pp. 45-87, 1981.

[18] B.T. Messmer, H. Bunke, A decision tree approach to graph and subgraph isomorphism detection, *Pattern Recognition*, vol. 32, pp. 1979-1998,1999.

[19] T. Miyazaki, The complexity of McKay's canonical labeling algorithm, in *Groups and Computation, II* (L. Finkelstein and W.M. Kantor, eds.), Amer. Math. Soc., Providence,

RI, pp. 239-256, 1997.

[20] D.C. Schmidt, L.E. Druffel, A Fast Backtracking Algorithm to Test Directed Graphs for Isomorphism Using Distance Matrices, *Journal of the Association for Computing Machinery*, 23, pp. 433-445, 1976.

[21] J.R. Ullmann, An Algorithm for Subgraph Isomorphism, *Journal of the Association for Computing Machinery*, vol. 23, pp. 31-42, 1976.