A Database of Graphs for Isomorphism and Sub-Graph Isomorphism Benchmarking

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

Abstract
Despite of the fact that graph based methods are gaining more and more popularity in different scientific areas, it has to be considered that the choice of an appropriate algorithm for a given application is still the most crucial task.
The lack of a large database of graphs makes the task of comparing the performance of different graph matching algorithms difficult, and often the selection of an algorithm is made on the basis of a few experimental data available on it.
In this paper we describe a database containing 72,800 couples of simple graphs especially devised for comparing the performance of isomorphism and graph-subgraph isomorphism algorithms. The 72,800 couples are split into 18,200 couples of isomorphic graphs and 54,600 couples of graphs with a sub-graph isomorphism mapping among them. The graphs include different categories as Randomly Connected Graphs, Regular Meshes (2D, 3D and 4D), Bounded Valence Graphs, Irregular Meshes and Irregular Bounded Valence Graphs. The size of the graphs ranges from few dozens to about 1000 nodes.

1. Introduction

Graphs are data structures endowed with such an expressive power to make their use profitable in the most disparate areas. By attributing a suitable meaning to nodes and branches of a graph, it is possible to achieve complete and univocal representations, of interest within many application domains, ranging from scientific to humanistic areas. Just to cite a few examples, graphs have been profitably used for representing syntactic components and their relationships in the sentences of a language [1], so as for describing the structure of chemical compounds or for dealing with traffic problems [2].
Recently, graphs have excited more and more interest within the scientific community active in the fields of Pattern Recognition and Computer Vision, and the applications employing graphs have multiplied [3-5].
In the last three decades, plenty of algorithms for graph matching, using either optimal or approximate methods, have been proposed with the specific aim of

reducing computational complexity of the matching process. In fact, despite the well-established mathematical background of the graph theory and the smartness of this powerful data structure, the problem of matching two graphs still remains affected by the high computational complexity. So, the problem of matching graphs, yet containing a few hundred of nodes, generally lead to time and memory requirements overcoming the capacity of the most powerful computer systems.
Among all the problems defined on graphs, the exact isomorphism and the sub-graph isomorphism detection play a key role, and are used in a variety of real applications. Despite the relative oldness of the problem, even today the literature is enriching with a variety of algorithms for solving the above mentioned matching problems. Far from giving an exhaustive list of the graph matching algorithms proposed up to now, some known algorithms are described in [6-12].
Notwithstanding, at the knowledge of the authors, only a few papers face the problem of comparing them in terms of key performance indices (memory and time requirements, maximum graph size, etc.) [13,14]. So, it seems that the habit of the authors of proposing more and more new algorithms is prevailing against the need of comparing the performances of the large variety of the known algorithms in a objective way. This lack causes that the users of graph-based approaches can only use qualitative criteria to select the algorithm which seems to better fit with the application constraints.
While it seems difficult to understand why up to now no serious attempt have been made for comparing graph matching algorithms, it can be surely recognized the need of a standard database of graphs specifically designed for the above mentioned aims.
In this paper we describe a database containing 72,800 couples of simple graphs usable for comparing the performance of isomorphism and graph-subgraph isomorphism algorithms. The 72,800 couples are made up of 18,200 couples of isomorphic graphs and 54,600 couples of graphs with a sub-graph isomorphism among them. The graphs include different categories, as *Randomly Connected Graphs*, *Regular Meshes (2D, 3D and 4D)*, *Bounded Valence Graph*, *Irregular meshes and Irregular Bounded Valence Graphs* (the meaning of irregularity is detailed in the following of the paper). The graph size ranges from few dozens to about 1000 nodes.

2. The Database

Generally, the unavailability of a standard database for performance evaluation issues can be attributed to the difficulty of determining what is generally required within the community asking for it. In fact, the database should be so wide to cover almost all the need of the community and at the same time should have a size allowing a simple distribution on the Web and/or by suitable media.
Often, the difficulty of characterizing the need of the potential users of the database and the tediousness of collecting the samples to be included in the database discourage the researchers from the important task of building a new database.
This problem is even more critical with reference to a collection of graphs: in fact,

being them a mean for representing object of very different nature, they are used in very disparate application domains with so many different uses that a complete and exhaustive analysis revealed to be very difficult. Anyway, the criteria adopted in the design of the database, preliminarily described in [14], have been aimed to have available a collection which:

i) provides a wide variety of graphs so as to cover the highest number of different applicative areas. It is worth noting that in some contexts the graphs exhibit structural properties which can simplify the matching algorithms (f.i. planar graphs, mesh connected graphs, and so on),
ii) contains couples of graphs satisfying different morphisms (f.i. isomorphism, graph sub-graph isomorphism, and so on),
iii) includes graphs with different size parameters as number of nodes and edges, density of edges.

In general, two main approaches can be followed for generating a database; the easier way is that of starting from graphs obtained from real data, but alternatively the database can be obtained synthetically, by a suitably designed computer program.
Although the first approach allows us to obtain rather realistic graphs, it is generally more and more expensive as it require the collection of real data and the selection of the set of algorithms to be used for obtaining graphs from data. The obtained graphs are dependent on the considered domain and the considered pre-processing algorithm, reducing significantly the generality of the database and its reusability in other contexts. On the contrary, the artificial generation of graphs is not only simpler and faster than collecting graphs from real applications, but also allows the controlled variation of various critical parameters of the underlying graph population, such as the average number of nodes, average number of edges per node, number of different labels and so on.
Starting from these considerations, we decided to generate the database synthetically, so obtaining, in relatively short times, a quite large database of graphs, easily expandable in the future.
The choice of the kind of graphs to include in the database started with an analysis of the graphs mainly used by members of the IAPR-TC15 community. Table 1 reports a classification of various kinds of graphs used within the Pattern Recognition field, recently proposed in [14].
The database is structured in couples of graphs. Two categories of couples of graphs have been introduced. Couples made of isomorphic graphs and couples made of graphs in which the second graph is a sub-graph of the first one. A total of 72,800 couples of graphs have been generated. These couples are split into 18,200 couples of isomorphic graphs and 54,600 sub-graph isomorphic couples.

Kind of graphs	Unlabeled Graphs Graphs with labels and/or attributes Random Graphs Hypergraphs
Constraints on graph structure	None Planar Bounded Valence Graphs Trees Others
Size of graphs	Small (less than 100 nodes) Medium (100 to 1000 nodes) Large (more than 1000 nodes)
Density of edges	Low (same order as number of nodes) Medium High (quadratic in order of nodes)
Type of node and branch attributes	None (i.e. Unlabeled Graphs) Symbolic Numeric Structured

Fig. 1: A classification of graphs.

Each category of couples is made of graphs differing for the structure and the size. In particular, the following kinds of graphs have been considered:

- ***Randomly Connected Graphs*** with different values of the edge density η;
- ***Regular Meshes***, with different dimensionality: 2D, 3D and 4D;
- ***Irregular Meshes*** with different degrees of irregularity ρ;
- ***Bounded Valence Graphs***, with valence equal to 3,6 and 9;
- ***Irregular Bounded Valence Graphs*** with a degree of irregularity α;

For the exact definition of η, ρ and α, see the sections 2.1, 2.3 and 2.5 respectively. Each obtained category contains couples of graphs of different size, ranging from few dozens of nodes to about 1000 nodes.
Table 2 reports the categories of the generated couples of graphs in the case of isomorphism.
In case of graph sub-graph isomorphism, for each category of graphs, we have generated couples of graphs in which the two graphs (satisfying the graph sub-graph condition) have three predefined size ratio σ; in particular we have considered for σ the values: $\sigma \in (0.2, 0.4, 0.6)$. So, each category of graphs contains a number of couples three times higher than that obtained in case of isomorphism.

	Kind of Graph	No. of Couples	Parameters	Size
ISOMORPHISM (18,200 couples of graphs)	**Randomly Connected Graphs** (3000 couples)	1000	η=0.01	From 20 to 1000
		1000	η=0.05	From 20 to 1000
		1000	η=0.1	From 20 to 1000
	Regular Mesh (2300 couples)	1000	2D-mesh	From 16 to 1024
		800	3D-mesh	From 27 to 1000
		500	4D-mesh	From 16 to 1296
	Modified Mesh (6900 couples)	3000	Irregular 2D-mesh $\rho \in \{0.2, 0.4, 0.6\}$	From 16 to 1024
		2400	Irregular 3D-mesh $\rho \in \{0.2, 0.4, 0.6\}$	From 27 to 1000
		1500	Irregular 4D-mesh $\rho \in \{0.2, 0.4, 0.6\}$	From 16 to 1296
	Bounded Valence Graphs (3000 couples)	1000	valence = 3	From 20 to 1000
		1000	valence = 6	From 20 to 1000
		1000	valence = 9	From 20 to 1000
	Irregular Bounded Valence Graphs (3000 couples)	1000	valence = 3 α=0.1	From 20 to 1000
		1000	valence = 6 α=0.1	From 20 to 1000
		1000	valence = 9 α=0.1	From 20 to 1000

Table 2: The graphs generated for the isomorphism case.

2.1 Randomly Connected Graphs

Randomly Connected Graphs are graphs in which the edges connect nodes without any structural regularity. It is assumed that the probability of an edge connecting two nodes of the graph is independent on the nodes themselves. To generate these graphs we have adopted the same model proposed in [10]: it fixes the value h of the probability that an edge is present between two distinct nodes n and n ¢ The probability distribution is assumed to be uniform.

According to the meaning of h, if N is the total number of nodes of the graph, the number of its edges will be equal to $hN*(N-1)$. However, if this number is not sufficient to obtain a connected graph (i.e. at least N-1 edges), further edges are suitably added until the graph being generated becomes connected. See Fig.1 for an

example.

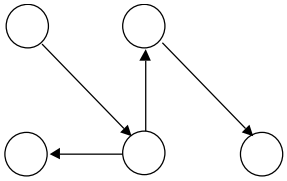
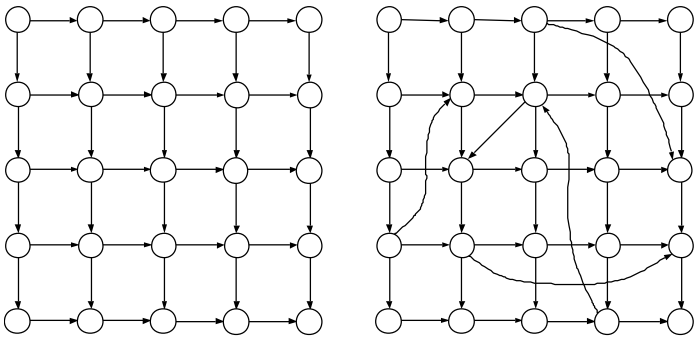


Fig. 1: An example of a r*andomly connected graph* with N=5 and h=0.2. The number of edges is equal to $hN*(N-1)$, i.e. 4.

2.2 Regular Meshes

It is generally agreed that graphs having a regular structure represent a worst case for general graph matching algorithms (i.e. algorithms working on any kind of graphs) [10]. This problem determined the born of specialized graph matching methods, with kind of algorithms able to efficient perform the matching for given graph structures. The database includes, as regular graphs, the mesh connected graphs (2D, 3D and 4D).



a) **b)**

Fig. 2: a) A 2D regular mesh graph with size 5x5 **b)** an irregular mesh (using r=0.2) obtained by adding to the regular mesh Nr (25·0.2=5) further edges. For each added edge, the starting and the ending nodes are randomly determined according to an uniform distribution.

The considered 2D mesh are graphs in which each node (except those belonging to the border of the mesh) is connected with its 4 neighborhood nodes (see Fig. 2a). Similarly, each node of a 3D and 4D graph has respectively connections with its 6 and 8 neighborhood nodes.
In all cases the meshes are open, i.e. the border nodes are connected only with internal nodes.

2.3 Irregular Meshes

Irregular Mesh-Connected graphs are made of a regular mesh with the addition of random edges uniformly distributed. The number of added branches is rN, where r is a constant greater than 0, and N is the number of nodes of the graph. Note that, the closer r is to 0, the more symmetric the graphs are. Figure 2b) shows a sample graph generated according to this model.

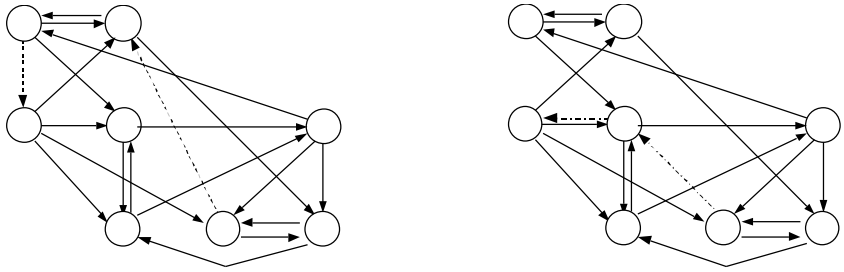
2.4 Bounded Valence Graphs

Bounded Valence Graphs are graphs in which every node has a number of edges (among ongoing and outgoing) lower than a given threshold, called Valence. A particular case occurs when the number of edges are equal for all the nodes; in this case the graph is commonly called fixed valence graph.
The database includes graphs with a fixed valence, that have been generated by inserting random edges (using an uniform distribution) with the constraint that the valence of a node cannot exceed a selected value; edge insertion continues until all the nodes reach the desired valence. Fig. 3 a) shows an example of this kind of graph.
It is worth noting that it is impossible to have fixed valence graphs with an odd number of nodes and an odd valence, but in our database we have only considered graphs with an even number of nodes.

2.5 Irregular Bounded Valence Graphs

In order to introduce some irregularity in the bounded valence graphs, we have introduced the *Irregular Bounded Valence Graphs***.** In these latter, the average valence of the nodes (that is, the ratio between the number of edges and the number of nodes) is still bounded, but the single nodes may have a valence which is quite different from the average, and which is not bounded by a constant value. To this aim, first a fixed valence graph is generated. Then, a certain number of edges are moved from the nodes they are attached to, to other nodes. The number of movements is equal to $M=0.1 \cdot N \cdot V$, where N is the number of nodes and V is the valence. This is equivalent to say that 10% of all the edges are moved.
The edges to be moved are chosen according to a random distribution with uniform probability. However, the new endpoints to which these edges are connected are not chosen uniformly, since this choice would affect only very slightly the overall variance of the valence of the nodes. Instead, after a random permutation of the nodes, the moved edges are distributed among the nodes using a probability distribution in which the node whose index is i has a probability of receiving an edge evaluated as $a \exp(-b\, i)$ where a and b depend on the number N of nodes, and satisfy the following constraints: i**)** the sum of the probabilities of the nodes of the graph is equal to 1 and **ii)** the probability of the node i multiplied by the number of edges to be moved is equal to $0.5\sqrt{N}$. Using this distribution the maximum valence of the

resulting graph will not be independent of *N*, and so special-purpose algorithms for bounded valence graphs cannot be employed, even though the graph is isomorphic for 90% of its edges to a fixed valence graph.

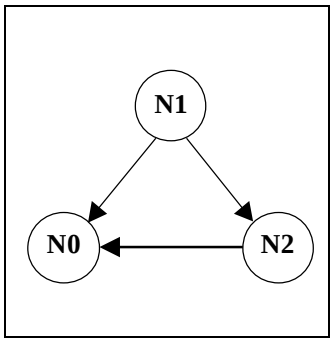


a) **b)**

Fig. 3: a) A fixed valence graph of *N*=8 nodes and valence=5. **b)** an irregular bounded valence graph, obtained from the graph a) by moving the two dashed edges (i.e. 10% of the 20 edges of the graph) according to the procedure illustrated in the text.

3. Implementation Notes

The graphs composing the database are stored in a binary format. Each graph is contained in a file, and the files are grouped into directories according to the kind of graphs and to the value of the parameters used in the generation.
In order to reduce the total disk space, each directory has been stored in a compressed archive using the ZIP format, which can be read by several programs freely available for the major operating systems.
Each graph file is composed by a sequence of 16-bit words; the words are encoded in little-endian format (e.g. LSB first). The first word represents the number of nodes in the graph. Then, for each node, there is a word encoding the number of branches coming out of that node, followed by a sequence of words encoding the endpoints of those branches. See the Fig. 4 for an example.
In addition to the graphs, we have also made available the sources of the programs used for their generation, which are coded in C++. Basically, we have a program for each type of graph. It generates a pair of isomorphic graphs (or a graph and one of its subgraphs) using the parameters passed as command-line arguments.



0003 Number of nodes
  0000 Number of edges outgoing from the node N0
  0002 Number of edges outgoing from N1
    0000 Destination node of the first edge of N1
        (edge N1 => N0)
    0002 Destination node of the second edge of N1
        (edge N1 => N2)
  0001 Number of edges outgoing from N2
    0000 Destination node of the only edge of N1
        (edge N2 => N0)

a)

b)

Fig. 4: a) A graph of the database, and **b)** its hexadecimal code, according to the defined format. The nodes N0, N1 and N2 are coded as 0000, 0001 and 0002 in hexadecimal format.

The generation is performed according to the following steps:

1) first, an adjacency matrix is generated according to the selected graph type and the corresponding parameters;
2) two random permutations of the nodes are generated, and, if a subgraph is required, a randomly chosen subset of nodes is removed from the second permutation together with all the related branches;
3) the two permuted graphs are saved in the above described format.

For the generation of the whole database, a Unix shell script has been written. The database has been generated on a computer with a 350MHz Pentium III processor and 128 MB of RAM. The underlying operating system was Linux 2.2.14, and the C++ compiler was egcs release 1.1.2. The total time required by the generation has been of about 7 hours, and the overall size of the compressed archive is about 420 Mbytes. The database is completed with a simplified ground truth; for each couple of graphs (both in the case of isomorphism and/or sub-graph isomorphism) the number of existing solutions is reported.

4. Conclusions

In this paper we present a database of synthetically generated graphs, especially devised for the benchmarking of graph matching algorithms, in particular exact isomorphism and sub-graph isomorphism.

The database is available at the URL: *http://amalfi.dis.unina.it/graph*, together with the sources of the programs (written in C++) used for generating and reading the graphs. It contains 72,800 couples of simple graphs (isomorphic and sub-graph isomorphic) belonging to different categories, and with sizes ranging from few nodes to about 1000 nodes. For each category a set of parameters allow to control the

generated graphs. So, in case of peculiar need, not covered by the database in its present form, the user can generate further graphs by using the provided generating functions.
The authors are working on the expansion of the database by including other categories of graphs.

References

[1] H. Ehrig, Introduction to Graph Grammars with Applications to Semantic Networks, *Comput. Math. Appl*. 23, pp. 557-572, 1992.

[2] D.H. Rouvray, A.T. Balaban, Chemichal Applications of Graph Theory, in *Applications of Graph Theory*, Academic Press, New York, pp.177-221, 1979.

[3] E.K. Wong, Model matching in robot vision by sub-graph isomorphism, *Pattern Recognition*, vol. 25, no. 3, pp. 287-304, 1992.

[4] K. Shearer, H. Bunke S. Venkatesh, D. Kieronska, Graph Matching for Video Indexing, *Computing* suppl. 12, pp. 53-62, 1998.

[5] M. Abdulrahim, M. Misra, A Graph Isomorphism Algorithm for Object Recognition, *Pattern Analysis and Applic.*, vol. 1, no. 3, pp. 189-201, 1998.

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

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

[8] C.M. Hoffman, *Group-theoretic Algorithms and Graph Isomorphism*, Springer, Berlin, 1998.

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

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

[11] 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, pp. 45-55, 1995.

[12] L.P. Cordella, P. Foggia, C. Sansone, M. Vento, Performance evaluation of the VF Graph Matching Algoritmh, *Proc. of the 10th ICIAP*, IEEE Computer Society Press, pp. 1172-1177, 1999.

[13] P. Foggia, C. Sansone, M. Vento, A performance comparison of five algorithms for graph isomorphism, *Proc. of the 3rd IAPR-TC15 Workshop on Graph-based Representations*, Italy, 2001.

[14] H. Bunke, M.Vento, Benchmarking of Graph Matching Algorithms, *Proc. 2nd IAPR-TC15 Workshop on Graph-based Representations*, Haindorf, pp. 109-114, 1999.