The Graph Database CD

SIVALab (*http://amalfi.dis.unina.it*) March 2001

This document provides a short description of the material found on the Graph Database CD, a huge collection of graphs of different kinds and sizes, generated for the purpose of establishing a common test ground for benchmarking graph matching algorithms.

Contents

1 Introduction

The *Graph Database* is a huge collection of graphs realized by the *Intelligent Systems and Artificial Vision Lab.* (*SIVALab*) of the University of Naples "Federico II". The aim of this collection is to provide the graph research community with a standard test ground for the benchmarking of graph matching algorithms.

The graphs have been randomly generated according to six different generation models, each involving different possible parameter settings. As a result, 84 diverse kinds of graphs are contained in the database. Each type is represented by thousands of pairs of graphs for which an isomorphism or a graph-subgraph isomorphism relation holds, for a total of 143,600 graphs.

In section 2.5 (Papers included in the CD) the reader will find links to more detailed information about the adopted generation models; the purpose of this document is of illustrating the contents of the database distribution CD, explaining the meaning and the format of the various files contained in it.

Notice that a browsable version of this document is available in the HTML file *grapdb.html*

2 CD contents

Data on CD is organized in the following directories:

Prefix	Description
`iso`	Pairs of isomorphic graphs
`si2`	Pairs of graph-subgraph isomorphic graphs; the subgraphsize is 20% of the full graph
`si4`	Pairs of graph-subgraph isomorphic graphs; the subgraphsize is 40% of the full graph
`si6`	Pairs of graph-subgraph isomorphic graphs; the subgraphsize is 60% of the full graph

Table 1: File name prefixes

`doc`

The database documentation (the file you are reading now) in printable and browsable format.

`graphs`

The files containing the actual graphs composing the database; for a more detailed description of these files and of their format see section 2.1 (The graph files). In the same directory are stored the ground-truth files, described in section 2.2 (The ground truth files).

`vflib-2.0`

The graph matching library VFLIB, vers. 2.0. More details are provided in section 2.3 (The VFLIB graph matching library).

`generators`

The sources of the programs used to generate the database, described in section 2.4 (Graph generation software).

`papers`

Articles describing the database and the VFLIB graph matching library; an overview of these papers is presented in section 2.5 (Papers included in the CD)

2.1 The graph files

The `graphs` directory contains 84 `.zip` files in which the actual graph files are stored in a compressed format. You will need a decompression utility able to deal with the `.zip` format in order to extract the graph files, if you have not yet installed one. For instance, you can try *WinZip* <`http://www.winzip.com`> for the MS Windows platform or *InfoZip* <`ftp://ftp.freesoftware.com/pub/infozip/`> for Unix, MacOS and other operating systems.

The name of the `.zip` files reflect the kind of graphs that are stored in the corresponding archives; in particular, the name is made of a prefix, which indicates the type of matching that holds between the graphs, and an identifier of the type of graphs (see section 2.5 (Papers included in the CD) for more details on the graph types). In Table 1 (File name prefixes) there is a list of prefixes with the corresponding meaning; Table 2 (Graph type identifiers) contains the description of the identifiers used for the types.

The prefix and the type identifier are separated by an underscore ("_"). For example, the file `si2_m2D.zip` has the prefix `si2`, meaning that it contains couple of graphs for which there is a graph-subgraph isomorphism relation, and the subgraph contain 20% of the nodes of the full graph; the type identifier is `m2D`, corresponding to bidimensional meshes with no additional random edges.

Identifier	**Description**
`r001`	randomly generated graphs with eta = 0.001
`r005`	randomly generated graphs with eta = 0.05
`r01`	randomly generated graphs with eta = 0.01
`m2D`	bi-dimensional meshes with rho = 0
`m2Dr2`	bi-dimensional meshes with rho = 0.2
`m2Dr4`	bi-dimensional meshes with rho = 0.4
`m2Dr6`	bi-dimensional meshes with rho = 0.6
`m3D`	tri-dimensional meshes with rho = 0
`m3Dr2`	tri-dimensional meshes with rho = 0.2
`m3Dr4`	tri-dimensional meshes with rho = 0.4
`m3Dr6`	tri-dimensional meshes with rho = 0.6
`m4D`	quadri-dimensional meshes with rho = 0
`m4Dr2`	quadri-dimensional meshes with rho = 0.2
`m4Dr4`	quadri-dimensional meshes with rho = 0.4
`m4Dr6`	quadri-dimensional meshes with rho = 0.6
`b03`	bounded-valence graphs with valence = 3
`b03m`	*modified* bounded-valence graphs with valence = 3
`b06`	bounded-valence graphs with valence = 6
`b06m`	*modified* bounded-valence graphs with valence = 6
`b09`	bounded-valence graphs with valence = 9
`b09m`	*modified* bounded-valence graphs with valence = 9

Table 2: Graph type identifiers

Group	Corresponding types
rand	r001, r005, r01
m2D	m2D, m2Dr2, m2Dr4, m2Dr6
m3D	m3D, m3Dr2, m3Dr4, m3Dr6
m4D	m4D, m4Dr2, m4Dr4, m4Dr6
bvg	b03, b03m, b06, b06m, b09, b09m

Table 3: Type groups

The files inside each `.zip` archive are divided into subdirectories, using three levels of grouping. The first level correspond to the type of matching, and the directories are named using the prefixes of Table 1 (File name prefixes). The second level groups together similar graph types; Table 3 (Type groups) shows the names of the directories, and the graph types they contain. Finally, the third level corresponds to graph types, and the directories are named after Table 2 (Graph type identifiers). For example, the graphs in the file `si4_r01.zip` get expanded into the directory `si4/rand/r01/`.

The name of the graph files in each `.zip` archive are obtained from the name of the `.zip` file by adding a suffix and an extension. The suffix, separated by an underscore ("_") from the graph type identifier, indicates the number of nodes in the graphs. The first letter of the suffix is "`s`" for *small* graphs (i.e. up to 100 nodes) and "`m`" for *medium-sized* graphs (i.e. above 100 nodes but below 2000); after this letter, there is the decimal representation of the number of nodes. The file extension (separated by a period "." from the rest of the file name) is made of a letter, which can be "`A`" or "`B`", and two digits, forming a number from `00` to `99`, that identifies one of the pairs of matching graphs. Within the pair, the graph with the "`A`" in the filename extension is the subgraph, for the subgraph-isomorphism matching types.

For example, a graph file name could be:

```
iso_r01_m200.A27
```

meaning that the graph is the first of the pair #27 of isomorphic randomly-generated graphs with eta=0.01 and having 200 nodes.

2.1.1 The graph file format

The graphs are stored in a compact binary format, one graph per file. The file is composed of 16 bit words, which are represented using the so-called *little-endian* convention, i.e. the least significant byte of the word is stored first.

The first word contains the number of nodes in the graph; this means that this format can deal with graphs of up to 65535 nodes (however, actual graphs in the database are quite smaller, up to 1024 nodes).

Then, for each node, the file contains the list of edges coming out of the node itself. The list is represented by a word encoding its length, followed by a word for each edge, representing the destination node of the edge. Node numeration is 0-based, so the first node of the graph has index 0.

The following C code shows how a graph file can be read into an adjacency matrix; the code assumes that the input file has been opened using the *binary* mode.

```
/* WARNING: for simplicity this code does not check for End-Of-File! */
unsigned short read_word(FILE *in)
  { unsigned char b1, b2;

    b1=getc(in); /* Least-significant Byte */
    b2=getc(in); /* Most-significant Byte */

    return b1 | (b2 << 8);
  }

/* This function assumes that the adjacency matrix is statically allocated.
 * The return value is the number of nodes in the graph.
 */
int read_graph(FILE *in, char matrix[MAX][MAX])
  { int nodes;
    int edges;
    int target;
    int i, j;

    /* Read the number of nodes */
    nodes = read_word(in);

    /* Clean-up the matrix */
    for(i=0; i<nodes; i++)
      for(j=0; j<nodes; j++)
        matrix[i][j]=0;

    /* For each node i ... */
    for(i=0; i<nodes; i++)
      {
        /* Read the number of edges coming out of node i */
        edges=read_word(in);

        /* For each edge out of node i... */
        for(j=0; j<edges; j++)
          {
            /* Read the destination node of the edge */
            target = read_word(in);

            /* Insert the edge in the adjacency matrix */
            matrix[i][target] = 1;
          }
      }

    return nodes;
  }
```

2.2 The ground truth files

For each `.zip` file in the `graphs` directory there is a corresponding `.gtr` file containing the ground truth regarding the pairs of graphs in the `.zip` archive. For example, the ground truth of the file `iso_b03.zip` is

in `iso_b03.gtr`.

A ground truth file is a text file containing a line for each pair of graphs in the `.zip` file; the line has two fields, separated by a TAB character (ASCII 9), and is terminated by a LF character (ASCII 10). The first field is the name of the first graph of the pair, while the second field is the number of matchings found.

For example, a line of the `iso_b03.gtr` file could be:

```
iso_b03_m1000.A00	1
```

to signify that between the graphs `iso_b03_m1000.A00` and `iso_b03_m1000.B00` there is only one isomorphism.

All the graph pairs in the `iso_*` files are present in the corresponding ground truth files. However, for the graph-subgraph isomorphisms, only a subset of the pairs have been accounted for in the ground truth files, due to the considerably larger computation time needed to generate this information.

2.3 The VFLIB graph matching library

In the directory `vflib-2.0` we have include the VFLIB graph matching library. This library implements several graph-matching algorithms, both for isomorphism and graph-subgraph isomorphism. In the `vflib-2.0/doc` directory we have provided a detailed *documentation* for the library installation and use.

Among the algorithms, there is also the **VF** graph matching algorithm, developed at the *Intelligent Systems and Artificial Vision Lab* of the University of Naples. In section 2.5 (Papers included in the CD) we have included two papers that describe this algorithm and provide a comparative performance evaluation of the different algorithms on the Graph Database.

The library class `GraphBinaryLoader` can be used to load and save graphs using the binary format adopted for the database, which is detailed in subsection 2.1.1 (The graph file format).

2.4 Graph generation software

In the directory `generators` there are the source files of the programs used to generate the graphs composing the database. These programs have been developed and used under *Linux*, and are not completely portable to other operating systems, since they use the `/dev/urandom` pseudo-device for initializing the random number generator.

Each program generates a pair of graphs; we have used shell scripts to call repeatedly the programs with the proper arguments for generating the whole database.

In table 4 (Graph generators) there is an outline of the programs and of their command line arguments.

For each program there is one source file with extension `.cc` . The programs `gen`, `gensub`, `genm2D` and `gensubm2D` need to be linked with the **VFLIB** graph matching library, which is described in section 2.3 (The VFLIB graph matching library). All other files are self-contained.

2.5 Papers included in the CD

In the `papers` directory we have provided three articles giving more detailed information on the database generation and the **VF** graph matching algorithm:

Name	Arguments	Types generated
gen	*nodes edges file1 file2*	iso*r001, iso*r005, iso*r01
gensub	*subgraph-nodes nodes edges file1 file2*	si*r001, si*r005, si*r01
genm2D	*nodes extra-edges file1 file2*	iso*m2D, iso*m2Dr2, iso*m2Dr4, iso*m2Dr6
gensubm2D	*subgraph-nodes nodes extra-edges file1 file2*	si*m2D, si*m2Dr2, si*m2Dr4, si*m2Dr6
genm3D	*nodes extra-edges file1 file2*	iso*m3D, iso*m3Dr2, iso*m3Dr4, iso*m3Dr6
gensubm3D	*subgraph-nodes nodes extra-edges file1 file2*	si*m3D, si*m3Dr2, si*m3Dr4, si*m3Dr6
genm4D	*nodes extra-edges file1 file2*	iso*m4D, iso*m4Dr2, iso*m4Dr4, iso*m4Dr6
gensubm4D	*subgraph-nodes nodes extra-edges file1 file2*	si*m4D, si*m4Dr2, si*m4Dr4, si*m4Dr6
genbvg	*nodes valence file1 file2*	iso*b03, iso*b06, iso*b09
gensubbvg	*subgraph-nodes nodes valence file1 file2*	si*b03, si*b06, si*b09
genbvgm	*nodes valence file1 file2*	iso*b03m, iso*b06m, iso*b09m
gensubbvgm	*subgraph-nodes nodes valence file1 file2*	si*b03m, si*b06m, si*b09m

Table 4: Graph generators

- *A Database of Graphs for Isomorphism and Subgraph Isomorphism Benchmarking* (file: `database.pdf`). This paper describes the criteria and the models used for the generation of the database.
- *An Improved Algorithm for Matching Large Graphs* (file: `vf-algorithm.pdf`). This is a description of the **VF** algorithm (version 2) which is implemented in the VFLIB graph matching library, presented in section 2.3 (The VFLIB graph matching library).
- *A Performance Comparison of Five Algorithms for Graph Isomorphism* (file: `benchmark.pdf`). This paper presents an experimental comparison, using the graph database, of five graph matching algorithms for the isomorphism problem. Four of the considered algorithms are implemented in the VFLIB graph matching library, which is described in section 2.3 (The VFLIB graph matching library). The fifth algorithm is *Nauty*; its implementation, together with some documentation, can be found at *The Nauty Page* <`http://cs.anu.edu.au/~bdm/nauty/`>

These papers have been submitted for acceptance at the 3rd IAPR-TC15 Workshop on Graph-based Representation (*GbR2001*).

In order to view the papers you must have installed a *PDF Viewer*, such as *Acrobat Reader* <`http://www.adobe.com/products/acrobat/readstep2.html`> for Windows, Mac and the major variants of Unix or *xpdf* <`http://www.foolabs.com/xpdf`> for Unix, VMS, OS/2 and BeOS.

3 References

1. H. Bunke, M. Vento. "Benchmarking of Graph Matching Algorithms", Proc. of the 2nd IAPR-TC15 Workshop on Graph-based Representations, Haindorf, 1999.
2. L. P. Cordella, P. Foggia, C. Sansone, M. Vento. "Performance Evaluation of the VF Graph Matching Algorithm", Proc. of the 10th ICIAP, IEEE Comp. Society Press, pp. 1172-1177, 1999.
3. J. R. Ullmann. "An Algorithm for Subgraph Isomorphism", Journal of the ACM, 23, pp. 31-42, 1976.

4. R. Mathon. "Sample Graphs for Isomorphism Testing", Congressus Numerantium, 21, pp. 499-517, 1978.

5. B. D. McKay. "Practical Graph Isomorphism", Congressus Numerantium, 30, pp. 45-87, 1981.