Report on the First Contest on Graph Matching Algorithms for Pattern Search in Biological Databases

Vincenzo Carletti,[1], Pasquale Foggia[1](✉), Mario Vento[1], and Xiaoyi Jiang[2]

[1] Department of Information Engineering, Electrical Engineering and Applied Mathematics University of Salerno, Fisciano, Italy
{mvento,pfoggia,vcarletti}@unisa.it

[2] Department of Computer Science
University of Münster, Münster, Germany
xjiang@uni-muenster.de

Abstract. Graphs are a powerful data structure that can be applied to several problems in bioinformatics, and efficient graph matching is often a tool required for several applications that try to extract useful information from large databases of graphs. While graph matching is in general a NP-complete problem, several algorithms exist that can be fast enough on practical graphs. However, there is no single algorithm that is able to outperform the others on every kind of graphs, and so it is of paramount importance to assess the algorithms on graphs coming from the actual problem domain. To this aim, we have organized the first edition of the Contest on Graph Matching Algorithms for Pattern Search in Biological Databases, hosted by the ICPR2014 Conference, so as to provide an opportunity for comparing state-of-the-art matching algorithms on a new graph database built using several kinds of real-world graphs found in bioinformatics applications. The participating algorithms were evaluated with respect to both their computation time and their memory usage. This paper will describe the contest task and databases, will provide a brief outline of the participating algorithms, and will present the results of the contest.

1 Introduction

Graphs are a powerful data structure that can be applied to several problems in bioinformatics. For instance, a lot of biological data can be represented as graphs: molecule structures, protein secondary structures, protein interaction networks. Since all these data are stored in huge biological data banks, one of the most important challenges for bioinformatics is the development of efficient Pattern Recognition tools to retrieve relevant information. Among these, the search for patterns inside a biological database can be formulated as a graph matching problem [1,9,12].

Graph matching within a biological database is a very computationally intensive problem, especially because of the size of the graphs involved. For instance,

© Springer International Publishing Switzerland 2015
C.-L. Liu et al. (Eds.): GbRPR 2015, LNCS 9069, pp. 178–187, 2015.
DOI: 10.1007/978-3-319-18224-7_18

if we consider a generic protein database, while a pattern can be a small graph of ten nodes, a complete protein graph, where the pattern is to be searched, can have several thousands of nodes and edges. So the number of candidate solutions for all the proteins can be huge.

To reduce the computational complexity, which in the worst case is exponential [5], some researchers have put their efforts in the development of more or less complex heuristics, that can decrease the matching time in the average case, at least for some classes of graphs, but still retain the exponential worst case; others have devoted their attention to inexact, suboptimal matching algorithms, that achieve a polynomial time, but do not ensure the accuracy of the found solutions, which may or may not be acceptable depending on the application. Besides the time complexity, space requirements of the algorithms also are an important concern when working with the large graphs of bioinformatics applications.

For this reason, it is extremely useful, for the researchers applying graph matching to bioinformatics, to obtain a benchmarking of matching algorithms that is not based on performance measurements taken on generic graphs, but exploits real data from bioinformatics databases, and at the same time attempts to highlight which are the conditions that give an advantage or a disadvantage to each of the algorithms.

This is the motivation behind the organization First International Contest on Graph Matching Algorithms for Pattern Search in Biological Databases (Biograph2014), hosted in August 2014 by the International Conference on Pattern Recogniton (ICPR2014). We, as the organizers of the contest, prepared a large dataset containing three different kinds of graphs extracted from bioinformatics databases. The participants to the contest were invited to submit their subgraph isomorphism algorithms, that we have tested measuring both the running time and the used memory. This initiative was endorsed by the Technical Committee #15 (Graph-based Representations in Pattern Recognition) of the International Association for Pattern Recognition (IAPR), since this Technical Committee has a long-standing tradition of promoting and supporting benchmarking initiatives for graph-based algorithms.

This report provides a description of the contest and of the dataset, and presents the results obtained by participant algorithms.

2 The Biograph2014 Contest

2.1 Description of the Contest Task

The competition task was to find the occurrences of a smaller *pattern graph* within a larger *target graph*. Such a task is often part of bioinformatics applications; for instance, in *motif discovery* [8,10] a repeating substructure within a set of larger structures is searched for, under the assumption that a commonly occurring substructure is likely to have some sort of biological significance.

More specifically, the algorithms were required to find all the *subgraph isomorphisms* between the pattern and the target graphs. A subgraph isomorphism is a mapping between the nodes of the pattern and the nodes of the target that is injective, is consistent with the node/edge labels, and preserves the edge structure of the graphs (for a formal definition of subgraph isomorphism see [5]).

Subgraph isomorphism is, as previously said, an NP-complete problem. For this reason, the contest was open also to inexact algorithms. However, only exact algorithms were submitted.

2.2 Performance Measures

We have evaluated the algorithms submitted to the contest according to three criteria:

1. CPU time needed for solving the problem;
2. used memory;
3. accuracy of the found solutions.

Regarding the time, we have performed the measurement by instrumenting the source code of the programs, so as to be able to consider only the time spent during the matching, and not the loading of the input graphs. For each pattern/target pair we have measured both the time required to find the first solution, and the time for all the solutions. The time needed for finding the first solution reflects the needs of applications where it is only required whether the searched substructure is present or not in the target; the time for all the solutions instead is a more appropriate criterion for applications where it is also important to find the position of the substructure within the target.

For the memory, we have used the Valgrind dynamic analysis tool [11] in order to measure the peak of the memory usage for each algorithm. This is an important factor that has to be considered when the application has to work with very large graphs, such as protein structures; while in the literature there are matching algorithms requiring a memory space that grows quadratically with respect to the number of nodes, their usage is not practical for many bioinformatics applications.

The accuracy measurements were introduced to take into account inexact algorithms. In particular, we have measured the number of missed solutions. Since all the algorithms submitted to the contest were exact, we have used these accuracy measures only to check for errors in their implementations. In the end, all the tested implementations have provided correct solutions, and so we will not discuss further these measures.

2.3 The Dataset

We have prepared three sets of graphs representing different structures used in biological problems:

- *Molecules dataset*, containing the chemical structures of different small organic compounds taken from the PubChem database hosted by the National Center for Biotecnology Information (NCBI) [3]; these are small graphs (up to 99 nodes), and the edge density is very low (about 2 edges per node). In this dataset, nodes represent atoms, and edges represent chemical bonds.

– *Proteins dataset*, containing the chemical structures of proteins and protein backbones taken from the Protein DataBank (PDB) [2]; these are very large graphs (up to 10081 nodes), and the edge density is very low (about 2 edges per node). Also in this dataset, nodes represent atoms, and edges represent chemical bonds.
– *Contact maps dataset*, containing the contact maps extracted from proteins in the PDB, by means of the CMView tool [14]. A contact map represents the adjacency relation in the 3D structure of a protein between the aminoacids belonging to different chains. The corresponding graphs are medium sized (up to 733 nodes), but the density is somewhat higher (about 20 edges per node). In this case, nodes represent amino acids in the protein chains, and edges represent the spatial adjacency relationship.

For each of the three datasets, some pattern graphs have been extracted by randomly chosing connected subgraphs of the target graphs. The characteristics of the datasets are summarized in Table 1.

Table 1. Characteristics of the three datasets of the contest

dataset	target graphs	nodes	edges/node	node labels	pattern graphs	pattern nodes
Molecules	10000	8–99	≈ 2	4–5	50	4–64
Proteins	300	535–10081	≈ 2	4–5	60	8–256
Contact maps	300	99–733	≈ 20	21	60	8–256

The dataset has been made publicly available at the link [7].

3 Contest Participants

Six algorithms were submitted to the Biograph2014 contest. In the following we will discuss briefly the characteristics of each of them. Table 2 contains a summary of these algorithms.

Regarding the overall structure of the presented algorithms, they can be roughly divided into two categories: algorithms based on backtracking, and algorithms based on constraint propagation. Backtracking algorithms incrementally build a solution, adding at each step a pair of nodes to the current mapping, after checking if the addition is consistent with the problem constraints and with algorithm-specific heuristics; if they reach a point where no other pair can be added, they *backtrack*, undoing the previous assignment and trying a different one. Algorithms based on constraint propagation, instead, are based on a formulation of graph matching as a Constraint Satisfaction Problem (CSP); they start by computing the *domain* for each node of the pattern, i.e. the set of target nodes that are compatible with it; then the domains are iteratively reduced by propagating constraints on the structure of the mapping, until only a few candidate matchings remain, that can be easily enumerated. Of course, the two approaches can be combined, by first applying constraint propagation to reduce the number of possible matchings, and then, if there is still more than one possible solution, by using backtracking to explore the reduced search space.

Table 2. The algorithms submitted to the contest

name	authors	ref.	structure	heuristics
L2G	I. Almasri		Backtracking	Dynamic node ordering, Minimum Remaining Value
LAD	C. Solnon	[13]	Constraint propagation	AllDifferent constraint
FC	C. Solnon	[13]	Constraint propagation	ForwardChecking(Edges), ForwardChecking(Diff)
PJ	J.-C. Janodet, F. Papadopoulos		Backtracking	Static node ordering on smallest domain, k-successors and k-predecessors
RI	V. Bonnici, R. Giugno	[4]	Backtracking	Static node ordering on connections to matched nodes
RI-DS	V. Bonnici, R. Giugno	[4]	Backtracking/ Constraint propagation	Static node ordering on connections to matched nodes, Domain precomputation based on labels and degrees

4 Contest Results

In the following we will present and discuss the results obtained by the contest participants. In order to provide a baseline value for the performance, we have added the time and memory measurements for the well known VF2 algorithm [6], which was considered very fast when it was introduced, and is still very commonly used.

The experiments have been conducted on a Linux-based system having twelve Intel Xeon 3.20Ghz cores. The system RAM was 32 GBytes.

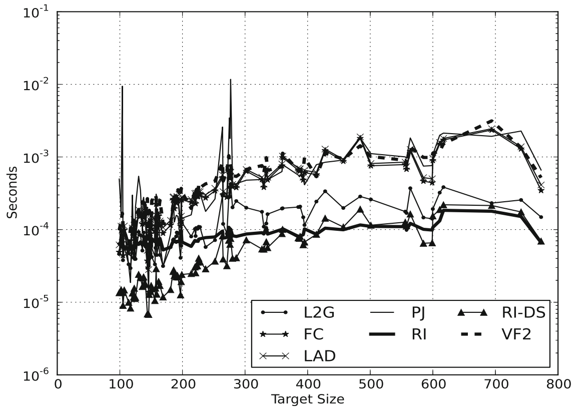
Figure 1 shows the average matching times for finding all the possible matchings between a pattern, with respect to the number of target nodes. On small and medium graphs (the Molecules and the Contact Maps datasets), RI and RI-DS are the fastest algorithms, with a slight advantage for RI-DS when the graphs are small. After these two algorithms come FC, LAD and L2G, with L2G at an advantage on medium graphs. Finally, PJ is the slowest, with a similar performance to the older VF2 algorithm.

Passing to the very large graphs of the Proteins dataset there is a slight change in the ranking. In this case, RI is definitely able to outperform the others by almost an order of magnitude to the closest competitors. L2G and RI-DS come second, with very similar performance. After them, at more than one order of magnitude, follow the remaining algorithms, with similar times.

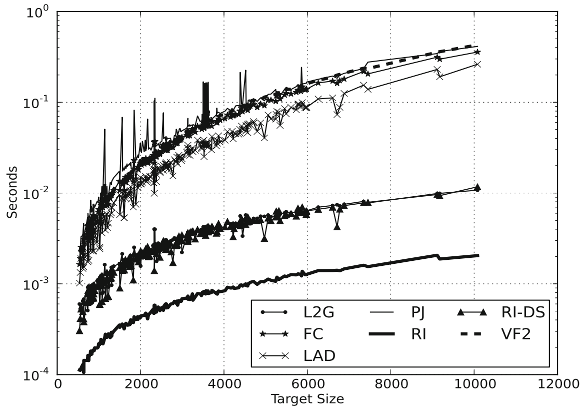
This behavior is confirmed by Table 3, which reports for each dataset the total time spent by each algorithm for finding all the solutions.

Thus, for these kinds of graphs, a safe choice would be the RI algorithm, which is the fastest on very large graphs, but is not very far from the fastest (RI-DS) on small and medium graphs.

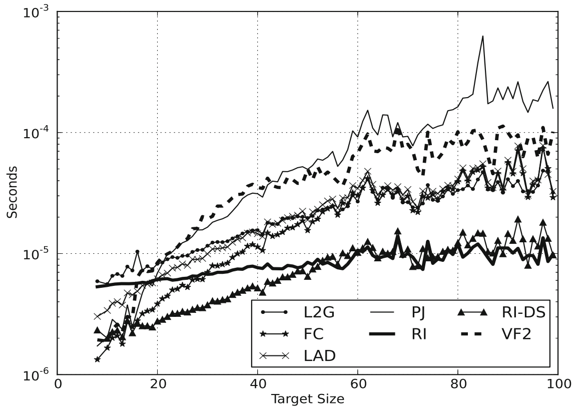
We have also analyzed the times for finding only the first solution for each pattern/target pair. Table 4 reports this information. While the fastest algorithms



Contact Maps



Proteins



Molecules

Fig. 1. Matching time versus target size for finding all the solutions

Table 3. Total matching times (seconds) for the search of all solutions. Boldface values are the best times for each database.

Algorithm	Contact Maps	Proteins	Molecules
L2G	2.14	58.34	7.38
FC	6.25	908.43	5.59
LAD	6.77	579.21	7.11
PJ	14.54	1169.06	16.02
RI	1.41	**11.62**	3.64
RI-DS	**0.79**	54.51	**2.59**
VF2	7.98	892.66	14.57

Table 4. Total matching times (seconds) for the search of the first solution. Boldface values are the best times for each database.

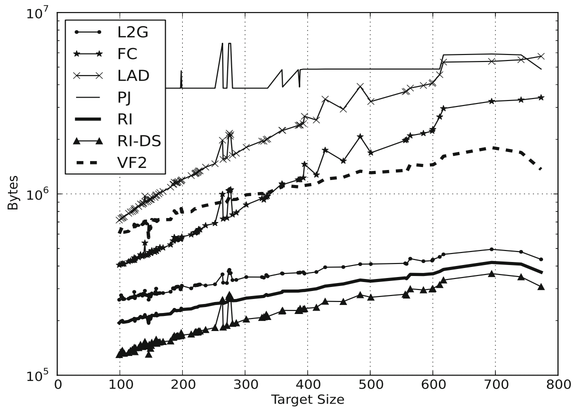
Algorithm	Contact Maps	Proteins	Molecules
L2G	1.94	37.74	6.14
FC	6.23	651.29	5.16
LAD	6.76	569.03	6.87
PJ	13.24	711.35	13.07
RI	1.38	**8.09**	3.51
RI-DS	**0.78**	53.66	**2.48**
VF2	7.62	570.19	12.43

remain the same, it is interesting to note that the reduction in computation time depends not only on the dataset (as it is expected, since different datasets have on the average a different number of matching solutions for each pattern/target), but also on the algorithm: for instance, on the Proteins dataset, PJ has a reduction of 40% of the total matching time, while RI-DS has a reduction of only 3%.

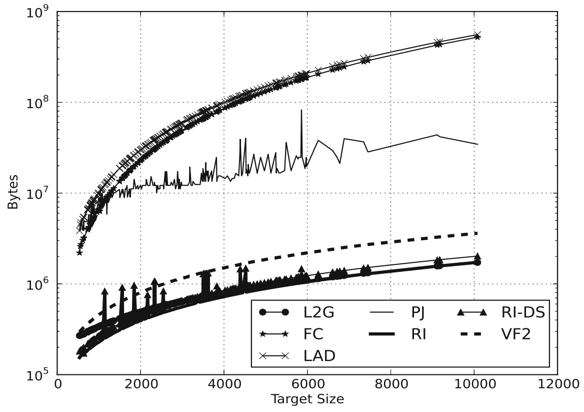
Figure 2 shows the memory usage plotted against the number of target nodes. For small and medium graphs, it can be seen that the most convenient algorithms are RI and RI-DS; if graphs are medium sized, also L2G has a competitive memory usage. It is also worth noting that while PJ has the highest memory occupation, on these graphs its trend is almost constant. LAD, FC and VF2 have an intermediate behavior, and are similar to each other, although it can be seen that the first two have a higher growth rate.

When we pass to the very large graphs, the best algorithms are RI and L2G, which have an almost identical memory occupation. RI-DS uses slightly more memory than the first two, and then comes VF2, with a higher curve but a very similar growth rate. After these algorithms there is PJ, with a larger memory occupation but a similar growth rate. The worst performers, with respect to memory, are LAD and FC, which not only use two orders of magnitude more memory, but also have a significantly faster growth.

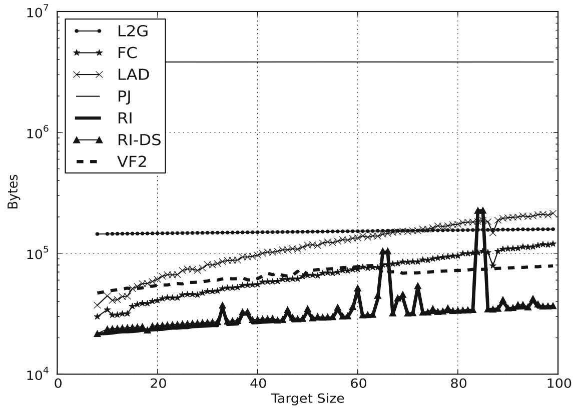
Table 5 shows the maximum memory usage for each algorithm for matching an entire dataset, that confirms the previous observations. However it is worth



Contact Maps



Proteins



Molecules

Fig. 2. Used memory versus target size for the search of all the solutions

Table 5. Maximum used memory (kbytes) for the search of all solutions. Boldface values are the best for each database.

Algorithm	Contact Maps	Proteins	Molecules
L2G	483	**1679**	154
FC	3318	509270	116
LAD	5602	543550	209
PJ	6613	80388	3737
RI	409	1685	219
RI-DS	**354**	1975	221
VF2	1756	3527	**77**

noting that on the Molecules dataset (the smallest graphs), the maximum memory usage of RI and RI-DS is significantly higher than the average value that can be observed from the chart in Fig. 2, because of a few pattern/target pairs (that can be seen as peaks in the chart). For this dataset, if we consider the maximum memory requirement, none of the newer algorithm outperforms the old VF2.

5 Conclusions

In this paper we have presented the first Contest on Contest on Graph Matching Algorithms for Pattern Search in Biological Databases (Biograph2014), hosted at ICPR2014. We have described the contest challenge and the dataset prepared and made publicly available for the contest, and we have presented and discussed the results obtained by the participating algorithms.

We hope that this initiative will have a following in immediate future. First, we wish to encourage other researchers to use the same database [7] for their benchmarking activities, so as to make the results more easily comparable. Second, we are planning to organize a second edition of the contest, using an extended database and possibly a more complex challenge (for instance, the algorithm in the first edition where required to perform a 1-pattern-vs-1-target matching, while it could be interesting to measure the time for a 1-pattern-vs-n-targets matching, which could allow the algorithm to do some kinds of optimizations).

References

1. Aittokallio, T., Schwikowski, B.: Graph-based methods for analysing networks in cell biology. Briefings in Bioinformatics 7(3), 243–255 (2006)
2. Berman, H., Westbrook, J., Feng, Z., Gilliland, G., Bhat, T., Weissig, H., Shindyalov, I., Bourne, P.: The Protein Data Bank. Nucleic Acids Research 28, 235–242 (2000)
3. Bolton, E., Wang, Y., Thyessen, P.A., Bryant, S.H.: PubChem: Integrated platform of small molecules and biological activities. Annual Reports in Computational Chemistry 4(12) (2008)

4. Bonnici, V., Giugno, R., Pulvirenti, A., Shasha, D., Ferro, A.: A subgraph isomorphism algorithm and its application to biochemical data. BMC Bioinformatics 14 (2013)
5. Conte, D., Foggia, P., Sansone, C., Vento, M.: Thirty years of graph matching in Pattern Recognition. IJPRAI 18(3), 265–298 (2004)
6. Cordella, L., Foggia, P., Sansone, C., Vento, M.: A (sub)graph isomorphism algorithm for matching large graphs. IEEE Transactions on Pattern Analysis and Machine Intelligence 26, 1367–1372 (2004)
7. Foggia, P., Vento, M., Jiang, X.: The biograph2014 contest dataset, http://biograph2014.unisa.it
8. Huan, J., Bandyopadhyay, D., Wang, W., Snoeyink, J., Prins, J., Tropsha, A.: Comparing graph representations of protein structure for mining family-specific residue-based packing motif. Journal of Computational Biology 12(6), 657–671 (2005)
9. Kuhl, F.S., Crippen, G.M., Friesen, D.K.: A combinatorial algorithm for calculating ligand binding. Journal of Computational Chemistry 5(1), 24–34 (1984)
10. Lacroix, V., Fernandez, C., Sagot, M.: Motif search in graphs: Application to metabolic networks. Transactions on Computational Biology and Bioinformatics (2006)
11. Nethercote, N., Seward, J.: Valgrind: A framework for heavyweight dynamic binary instrumentation. In: Proceedings of the 2007 ACM SIGPLAN Conference on Programming Language Design and Implementation, PLDI 2007, pp. 89–100. ACM (2007)
12. 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)
13. Solnon, C.: Alldifferent-based filtering for subgraph isomorphism. Artificial Intelligence 174(12-13), 850–864 (2010)
14. Vehlow, C., Stehr, H., Winkelmann, M., Duarte, J.M., Petzold, L., Dinse, J., Lappe, M.: CMView: Interactive contact map visualization and analysis. Bioinformatics (2011), doi:10.1093/bioinformatics/btr163