Journal of Bioinformatics and Computational Biology
Vol. 8, No. 2 (2010) 199–218

DOI: 10.1142/S021972001000477X

Imperial College Press
www.icpress.co.uk

SIGMA: A SET-COVER-BASED INEXACT GRAPH MATCHING ALGORITHM*

MISAEL MONGIOVÌ†, RAFFAELE DI NATALE‡, ROSALBA GIUGNO§, ALFREDO PULVIRENTI¶ and ALFREDO FERRO‖

Dipartimento di Matematica ed Informatica, Università di Catania
V.le A. Doria, 6, Catania, 95125, Italy
†*mongiovi@dmi.unict.it*
‡*dinatale@dmi.unict.it*
§*giugno@dmi.unict.it*
¶*apulvirenti@dmi.unict.it*
‖*ferro@dmi.unict.it*

RODED SHARAN

Blavatnik School of Computer Science, Tel Aviv University
Tel Aviv, 69978, Israel
roded@tau.ac.il

Received 20 July 2009
Revised 15 October 2009
Accepted 15 October 2009

Network querying is a growing domain with vast applications ranging from screening compounds against a database of known molecules to matching sub-networks across species. Graph indexing is a powerful method for searching a large database of graphs. Most graph indexing methods to date tackle the exact matching (isomorphism) problem, limiting their applicability to specific instances in which such matches exist. Here we provide a novel graph indexing method to cope with the more general, inexact matching problem. Our method, SIGMA, builds on approximating a variant of the set-cover problem that concerns overlapping multi-sets. We extensively test our method and compare it to a baseline method and to the state-of-the-art Grafil. We show that SIGMA outperforms both, providing higher pruning power in all the tested scenarios.

Keywords: Indexing; graph matching; network querying.

1. Introduction

Data in many biological domains are represented as graphs, where nodes correspond to molecules and edges connect related molecules. Mining such data to

*A preliminary version of this paper appeared as Mongiovì *et al.*[1] in the *Proceedings of the CSB 2009 Conference.*
†Corresponding author.

search for specific subgraphs is a fundamental step in identifying similarities among molecules, molecular networks etc. For example, querying for protein pathways within a collection of protein-protein interaction networks can identify matching pathways that are conserved in evolution and assist in the functional annotation of proteins and the prediction of their interactions.

Graph indexing is a common technique for performing searches in large databases. In a pre-processing phase, each graph of the database is analyzed in order to extract and store its features (composing the graph index). These could be either all the paths up to a certain length,[2–6] trees[7] or general subgraphs.[8,9] These indices are then used by a filtering phase to prune graphs that cannot contain instances of the query. The remaining candidates are finally verified in a matching phase through a subgraph matching algorithm.[10]

While many graph indexing algorithms have been suggested for the exact search (subgraph isomorphism) problem, very few algorithms exist for inexact search. In the most basic variant of the problem, the goal is to allow matches that are isomorphic to the query up to a few edge indels. Since edge insertions (i.e. extra edges in the match that do not have counterparts in the query) can be discarded while only improving the quality of the match, the core of the problem is handling edge deletions. More general variants allow label mismatches, node insertions and deletions and so on.

Molecular compounds, for instance, can be represented as graphs where atoms are vertices and bounds are edges. Molecules which share part of a given molecular structure often have similar chemical properties. Here inexact matching may assist in the identification of drugs which are active against some pathologies or have side effect, when the molecular structure responsible for a particular activity or side effect is known. Figure 1 shows that antidepressive molecules such as L-tryptophan share compounds with alkaloids, amines isolated from plants, including poisons such as strychnine and with powerful hallucinogenic drugs such as LSD. The shared parts

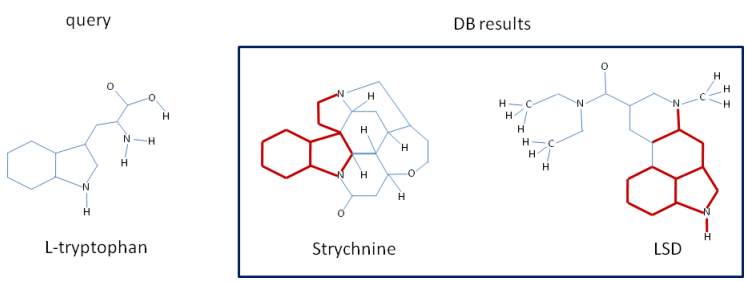

Fig. 1. An example of inexact matching on molecular compounds. The compounds are represented as graphs where vertices are atoms and are labeled with their element symbol (unlabeled vertices correspond to C atoms), and edges are bonds (double bounds are represented as single edges). The red-colored part of strychnine and LSD matches a part of the Tryptophan structure. Finding this match allows to identify compounds which share chemical properties.

are highlighted. By deleting 7 edges from L-tryptophan, the remaining compound has a match in strychnine, while 5 deletions are needed to find a match in LSD. In Ref. 11 it is shown that both L-tryptophan and LSD are involved in serotonin syndrome and that strychnine poisoning produces similar symptoms, being involved in differential diagnosis.

To tackle the inexact matching problem, several systems[5,6] apply exact search techniques to queries that contain wildcard-nodes that can match any node and wildcard-paths, which are paths of any length that connect the two nodes. Indexing is used to filter out graphs in the database that do not contain the subparts of the query that are completely specified. A shortcoming of this approach is the need to specify in advance the parts of the query that may change.

Grafil[12] has been the first attempt to realize indexing for inexact searches. It transforms the edge deletions into feature misses in the query graph, and uses an upper bound on the maximum number of allowed feature misses for graph filtering. Grafil in fact clusters the features according to their selectivity and applies a multi-filter strategy, where each filter uses a distinct cluster and the filtering results are combined. SAGA[13] is a more flexible indexing system, which can handle also node insertions and deletions. Key to the algorithm is a distance measure between graphs. Fragments of the query are compared to database fragments using the distance measure. Matching fragments are then assembled into larger matches using a clique detection heuristic and, finally, candidate matches are evaluated. The SAGA algorithm was successfully applied to mine biological pathways, but its distance metric limits its applicability in other domains in which one seeks direct control over the number of edge deletions introduced. Closure-Tree[14] is another tool for inexact matching which focuses on the edit distance between the query and its candidate matches. However, for efficiency reasons, the edit distance computations are approximate and, hence, the tool can miss true matches.

In this paper we present the Set-cover-based Inexact Graph Matching Algorithm (SIGMA), an efficient feature-based filtering algorithm for inexact graph matching. The algorithm is based on associating a feature set with each edge of the query and looking for collections of such sets whose removal will allow exact matching of the query with a given graph. This translates into the problem of covering the missing features of the graph with overlapping multisets. We formulate this variant of set cover and provide a greedy approximation for it. We extensively test SIGMA in a simulated setting, querying small molecules against a database of molecular compounds. We compare it to a baseline filtering method and to the state-of-the-art Grafil; we show that SIGMA exhibits consistently higher filtering power, where the difference grows with the size of the query.

To demonstrate the utility of SIGMA in a real biological setting, we apply it to query yeast and human protein complexes. While there are previous methods for protein complex querying, such as Torque[15] and QNet,[16] this is the first application of a graph indexing technique for this task. In contrast to the previous methods, SIGMA aims to find matches that are topologically similar to the query, and does

not assume homeomorphism of the two topologies (as in QNet) or that the exact topology is not important (as in Torque).

Our contribution is three-fold:

(i) We define a new effective pruning rule for inexact matching based on *multiset multi-cover*, a variant of the well known set-cover problem.
(ii) We provide a tight greedy approximation for multiset multi-cover, which is crucial for efficient and effective pruning.
(iii) We evaluate the performance of the proposed method, compared to a state-of-the-art approach, over a molecular compound dataset. In addition, we apply our method in a systematic comparison of protein complexes from yeast and human.

The paper is organized as follows: Section 2 provides the basic definitions of graph indexing. Section 3 derives new pruning rules for inexact matching that are based on several variants of the set cover problem. Finally, experimental results and a comparison to Grafil are presented in Sec. 4.

2. Preliminaries

An undirected labeled graph (in the following, simply a graph) is a 4-tuple $G = (V, E, \Sigma, l)$ where V is the set of vertices, E is the set of edges, Σ is the alphabet of labels and $l : V \rightarrow \Sigma$ is a function which maps each vertex to a label. We denote as $V(G)$ the set of vertices of G and by $E(G)$ the set of edges of G. We say that a graph G_1 is *subgraph* of G_2, denoted $G_1 \subseteq G_2$, if $V_1 \subseteq V_2$ and $E_1 \subseteq E_2$.

Given two graphs $G_1 = (V_1, E_1, \Sigma, l)$, $G_2 = (V_2, E_2, \Sigma, l)$ an *isomorphism* (that respects the labels) between G_1 and G_2 is a bijection $\phi : V_1 \rightarrow V_2$ so that:

- $(u, v) \in E_1 \Leftrightarrow (\phi(u), \phi(v)) \in E_2$
- $l(u) = l(\phi(u)), \forall\, u \in V_1$

A *subgraph isomorphism* between G_1 and G_2 is an isomorphism between G_1 and a subgraph of G_2. We say that a graph G_1 admits an *exact match* in G_2 if there exist a subgraph isomorphism between G_1 and G_2. We say that a graph G_1 admits an *inexact match* in G_2 with r *deletions* if there exists a subgraph isomorphism between a graph G_r obtained from G_1 by removing arbitrarily r edges, and G_2. We say also that G_1 is contained in G_2 with r deletions.

We define a multiset as a pair (A, m) where A is a set and m is a function from A to the set $\mathbb{N}$ of natural numbers. We say that $m(a)$ is the multiplicity of the element a. Given a set U, we say that $A' = (A, m)$ is a multiset of U if $A \subseteq U$. For simplicity, we extend the function $m()$ to all element of U by setting $m(u) = 0$ for each $u \in U - A$. We define the cardinality of a multiset $A' = (A, m)$ as $|A'| = \sum_{a \in A} m(a)$

Let $A' = (A, m)$ and $B' = (B, n)$ be two multisets. We define the difference $A' - B'$ as the set $C' = (C, p)$ where $C = \{c \in A | m(c) > n(c)\}$ and $p(c) = m(c) - n(c)$

for each element $c \in C$. We define the intersection $A' \cap B'$ as the set $C' = (C, p)$ where $C = A \cap B$ and $p(c) = min(m(c), n(c))$ for each element $c \in C$. We define the union $A' \cup B'$ as the set $C' = (C, p)$ where $C = A \cup B$ and $p(c) = m(c) + n(c)$ for each element $c \in C$. We say that $A' \subseteq B'$ if for each $a \in A$ we have $a \in B$ and $m(a) \leq n(a)$.

Given a multiset C and two multisets $A, B \subseteq C$, it is easy to verify that the following relations hold:

- $C - (C - A) = A$
- $C - A \subseteq C - B \Leftrightarrow B \subseteq A$

2.1. *Filtering techniques for exact matching*

Given a database $D = \{G_1, G_2, \ldots, G_n\}$ of graphs, performing an exact graph query Q in D calls for finding all graphs G in D such that $Q \subseteq G$.

Since checking all graphs of D is very expensive, a feature-based indexing system applies a filter-and-verification framework which allows to prune the graphs of the databases which cannot contain the query. A feature is a small graph which allows to discriminate, by checking its inclusion, the graphs which could contain the query from the graphs that cannot contain it. We denote as $\mathcal{F}$ the set of all possible features. The choice of $\mathcal{F}$ depends on the particular system used. In this paper we refer to a generic set of features.

Basically, graph-based graphs indexing systems are based on the observation that for a query Q to admit a match in the graph G, it is necessary that each feature of $\mathcal{F}$ contained in Q is also contained in G. More precisely, when we say that a feature f is contained in G we mean that there exists an isomorphism between f and a subgraph of G. The pruning is performed by the following phases:

- **Pre-processing:** This phase is off-line and is independent from the query. Each graph of the database is examined in order to extract all features of $\mathcal{F}$ which are contained in the graph. The set of features of all graphs are recorded in a data structure called *graph index*.
- **Filtering:** The given query Q is examined in order to extract a set of features contained in Q. A candidate graph set is computed comparing the extracted set of features against the corresponding sets in the graph index.
- **Matching:** Each candidate graph is examined in order to verify if there are matches between the query and the graph.

The feature-based condition for Q to be contained in G can be expressed as a pruning rule. We denote as H_G the set of features contained in the graph G. Given a query Q, the graph G can be discarded if $H_Q \nsubseteq H_G$. To apply this pruning rule we only check the existence of a subgraph isomorphism between features and graphs. Given a feature f and a graph G there can be several distinct subgraphs of G which admit an isomorphism with the feature f. Each subgraph of G which admits an isomorphism with f is referred as a distinct *feature occurrence* of f

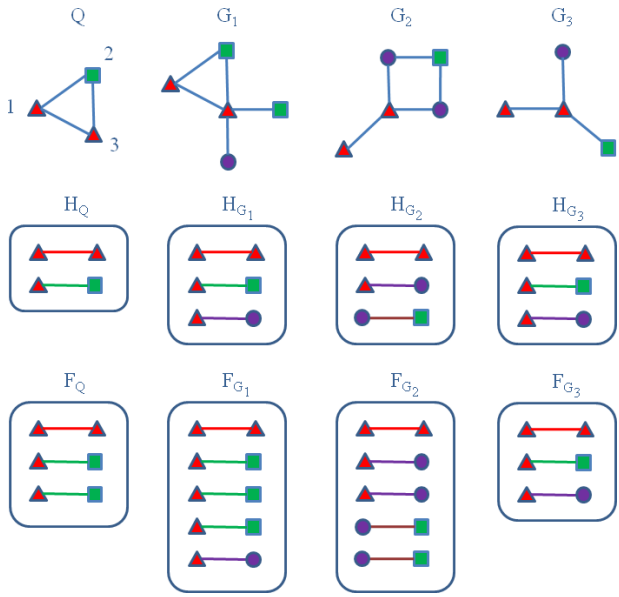

Fig. 2. An example of exact matching in a database of graphs. Here we consider as features simply edges (graphs with size 1). The first row shows the query graph Q and the database of graphs $\{G_1, G_2, G_3\}$. The second and third rows report respectively, the sets of features and the multisets of features associated to each graph. The multiplicity of multisets take into account the number of feature occurrences. For instance, the query Q contains two occurrences of the feature triangle-square, one over the nodes 1-2 and the other over the nodes 3-2. In this example the query Q is contained in the graph G_1 but not in the graphs G_2 and G_3. G_2 can be discarded by the filtering process because the feature triangle-square is not contained in H_{G_2}. G_3 can be discarded taking into account the number of occurrences by observing that the feature triangle-square have two occurrences in the query and only one in the graph.

in G. The pruning power can be increased by considering the number of feature occurrences. We denote as F_G the multiset of features of the graph G which associate to each feature, the number of occurrences of it in the graph. For the query Q to be contained in the graph G, the number of occurrences of each feature in Q must be lower or equal to the number of occurrences of the corresponding feature in G. This means that we can discard the graph G if $F_Q \nsubseteq F_G$.

For example, the query Q in Fig. 2 matches with the graph G_1 but not with G_2 and G_3. It contains one occurrence of the feature triangle-triangle and two occurrences of the feature triangle-square. G_2 can be discarded by observing $H_Q \nsubseteq H_{G_2}$. By considering the number of feature occurrences, G_3 can also be discarded, since $F_Q \nsubseteq F_{G_3}$.

3. A Filtering Technique for Inexact Matching

In this section we develop effective pruning rules for inexact matching. We focus on the following problem: Given a query Q and a graph G, does Q admit an inexact

match in G with at most r deletions? The scheme that we develop is based on associating a feature set F_e with each edge e of the query (i.e. the set of features that contain this edge) and looking for collections of such sets whose removal will allow exact matching of the query with G. The resulting problem can be formulated as a set cover problem: given a set Y (of features of Q which are missing in G) and a family S of sets (of features associated to each edge), find the smallest subfamily Γ of S that covers Y, i.e. $\bigcup_{X\in\Gamma} X \supseteq Y$.

Such a subfamily represents a set of query edges whose deletion assure that all the features of Q are contained in G. If a subfamily Γ of size r does not exist, we can assume that if we delete r edges in all possible ways, we can always find at least one feature of the query which is not contained in the graph, therefore the graph can be discarded.

We can strengthen the above formulation by considering the multiplicity of feature occurrences. Let $E_\gamma \subseteq E(Q)$ be a subset of the query edges. We denote as F_Q the multiset of features of Q and as F_{E_γ} the multiset of features which contain at least one of the edges in E_γ. If Q admits an inexact match in G with r deletions, there must exist an r-size edge set E_γ such that $F_Q - F_{E_\gamma} \subseteq F_G$. Hence the following pruning rule can be inferred:

Pruning rule 1. Given a query Q with r allowed deletions, a graph G can be discarded if for each $E_\gamma \subseteq E(Q)$ with $|E_\gamma| = r$ we have

$$F_{E_\gamma} \not\supseteq F_Q - F_G$$

Clearly this pruning rule cannot be applied efficiently because the number of possible r-subsets of $E(Q)$ grows exponentially with r, and the rule must be verified for all the graphs in the database. Instead, we resort to a multiset cover approach and define a new pruning rule based on a greedy algorithm.

In the multiset multi-cover problem $Y = (Y', m_Y)$ is a multiset and S is a family of multisets. Each element (feature) f of Y has a multiplicity $m_Y(f)$ which specifies the number of times f has to be covered, and it occurs in each set X of S with a given multiplicity $m_X(f)$. The goal is to find the minimum-size set Γ such as $\bigcup_{X\in\Gamma} X \supseteq Y$, i.e. for each $f \in Y'$, $\sum_{X\in\Gamma} m_X(f) \geq m_Y(f)$. In its general formulation, a set of S can be chosen several times (Γ is a multiset too). In what follows we consider the further constraint that each set of S can be chosen at most once. In our case, the multiset to be covered is $Y = F_Q - F_G$, and the collection of covering multisets is $S = \{F_e\}_{e\in E(Q)}$. If Y admits no multiset multi-cover of size r then G can be discarded (see Fig. 3).

Set-cover is known to be NP-complete,[17] but can be solved by a simple greedy heuristic with approximation ratio $H(max\{|X| : X \in S\})$, where $H(n) = 1 + 1/2 + \cdots + 1/n$.[17,18] The more general multiset multi-cover problem was shown to admit the same approximation ratio.[19] Figure 4 describes a greedy heuristic for the multiset multi-cover problem. At each iteration, the algorithm chooses the multiset X of the family S which maximizes the number of newly covered feature occurrences of Y. The chosen set is added to the cover, and its elements are removed from Y.

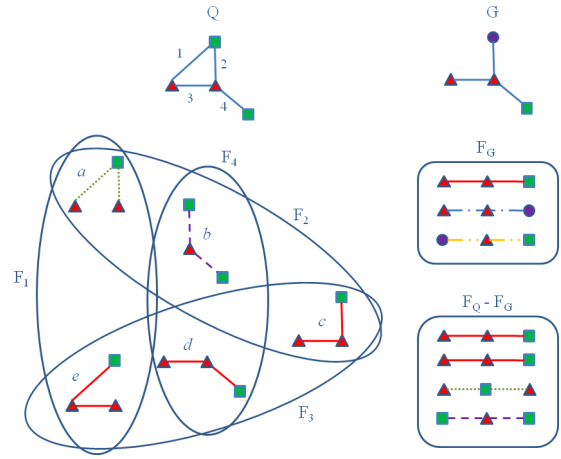

Fig. 3. An example of a query Q and a graph G which contains a copy of Q with two deletions. We consider as features all connected subgraphs containing exactly two edges. Left: Q and all the feature occurrences it contains (F_Q). The line type of feature occurrences is chosen according to the feature they correspond to. Each set F_i indicates all the feature occurrences that contain the edge i. Right: G, its multiset of features (F_G) and the multiset of missing features ($F_Q - F_G$). The minimum cover of $F_Q - F_G$ by the family $\{F_1, F_2, F_3, F_4\}$ is of cardinality 2, implying that at least two deletions are needed for a match. $\{F_1, F_2\}$ is a possible cover, implying that G is a candidate to match Q with edges 1 and 2 deleted.

Greedy-Multiset-Multicover(Y, S)
$\Gamma \leftarrow \phi$
while$Y \neq \phi$ do
$X \leftarrow argmax_{\overline{X} \in S} |\overline{X} \cap Y|$
$Y \leftarrow Y - X$
$\Gamma \leftarrow \Gamma \cup \{X\}$
return Γ

Fig. 4. A greedy algorithm for the multiset multi-cover problem.

For the greedy algorithm to be used effectively for filtering, it is essential to have a tight lower bound of the optimal solution. We prove a tight lower bound below.

Let $Y = (Y', m_Y)$ be the multiset of features to be covered. Let $cost(f, i)$ be a function from $Y' \times \mathbb{N}$ to $\mathbb{R}$, which assigns a cost to each feature occurrence covered by the greedy algorithm. The feature occurrences are ordered by the time they are covered by the algorithm. The cost is assigned at each step (execution of the while loop) of the algorithm, spreading a unit cost over all the feature occurrences which are being covered, i.e. each feature occurrence is assigned a cost $1/c$, where c is the number of newly covered occurrences. Formally the function *cost* is defined as follows: Let $new\_cov(f, s)$ be the number of newly covered occurrences of f at the

step s and $cov(f, s)$ be the total number of covered occurrences of f after the step s, i.e. $cov(f, s) = \sum_{t=1\cdots s} new\_cov(f, t)$. The function *cost* is defined as

$$cost(f, i) = \begin{cases} \dfrac{1}{\sum\limits_{f \in Y'} new\_cov(f, s)} & \text{if } cov(f, s-1) < i \leq cov(f, s) \\ \\ 0 & \text{otherwise} \end{cases}$$

Let Γ be the cover returned by the greedy algorithm, Γ^* the exact cover and $r_X(f) = min(m_X(f), m_Y(f))$. The following theorem bounds the size of the cover returned by the greedy algorithm.

Theorem 1. *Let* $\alpha(f) = cost(f, m_Y(f))$ *and* $\beta = \sum_{f \in Y'} \sum_{i=1}^{m_Y(f)} (\alpha(f) - cost(f, i))$ *then,*

$$|\Gamma^*| \geq \min_{\Gamma' \subseteq S : \sum_{(X, m_X) \in \Gamma'} \sum_{f \in X} r_X(f)\alpha(f) - \beta \geq |\Gamma|} |\Gamma'|$$

Proof. We show that

$$\sum_{(X, m_X) \in \Gamma^*} \sum_{f \in X} r_X(f)\alpha(f) - \beta \geq |\Gamma|$$

The claim follows since $\Gamma^* \subseteq S$ and each element of a set is always greater than or equal to the minimum over that set.

The total cost assigned to all the feature occurrences is equal to $|\Gamma|$. Thus:

$$\begin{aligned} |\Gamma| &= \sum_{f \in Y'} \sum_{i=1}^{m_Y(f)} cost(f, i) \\ &= \sum_{f \in Y'} m_Y(f) \cdot cost(f, m_Y(f)) \\ &\quad - \sum_{f \in Y'} \sum_{i=1}^{m_Y(f)} (cost(f, m_Y(f)) - cost(f, i)) \\ &= \sum_{f \in Y'} m_Y(f)\alpha(f) - \beta \\ &\leq \sum_{(X, m_X) \in \Gamma^*} \sum_{f \in X} r_X(f)\alpha(f) - \beta. \end{aligned}$$

□

By the above theorem, we obtain the following pruning rule:

Pruning rule 2. Given a query Q with r allowed deletions and a graph G. Let $|\Gamma|$ be the cover returned by the greedy algorithm when executed on $F_G - F_Q$. G can be discarded if

$$r < \min_{\Gamma' \subseteq S : \sum_{(X, m_X) \in \Gamma'} \sum_{f \in X} r_X(f)\alpha(f) - \beta \geq |\Gamma|} |\Gamma'|$$

The right side can be easily computed by ranking the sets of S by the score $\sum_{f \in X} r_X(f)\alpha(f)$ in decreasing order, and taking them one by one until the sum of the scores is greater than or equal to $|\Gamma| + \beta$.

3.1. *An attempt to increase the filtering power*

Using multisets alone does not capture interdependencies between them, i.e. two multisets of features may include the same feature occurrence but in the cover we may count it twice (see Fig. 5).

To this end we introduce a new variant of the set-cover problem, which we call *Multi-cover by Overlapping Multisets* (MOM). Let U be a set of elements (feature occurrences), $\mathcal{F}$ a set of features and f a function that associates with each element of U a feature from $\mathcal{F}$. Given $A \subseteq U$, we define the covering of A, denoted as $Cov_f(A)$, as the multiset $D' = (D, m)$ of $\mathcal{F}$ so that $D = \{f(a) | a \in A\}$ and $m(d) = |\{a \in A | f(a) = d\}|$. We define the MOM problem as follows: For a multiset Y of $\mathcal{F}$ and given a family S of subsets of U, find the smallest subfamily Γ of S so that $Cov_f(\bigcup_{X \in \Gamma} X) \supseteq Y$.

Note that in Fig. 5 the minimum cover for MOM is $\{F_1, F_6, F_7\}$, so G is not a candidate to match Q with at most two deletions.

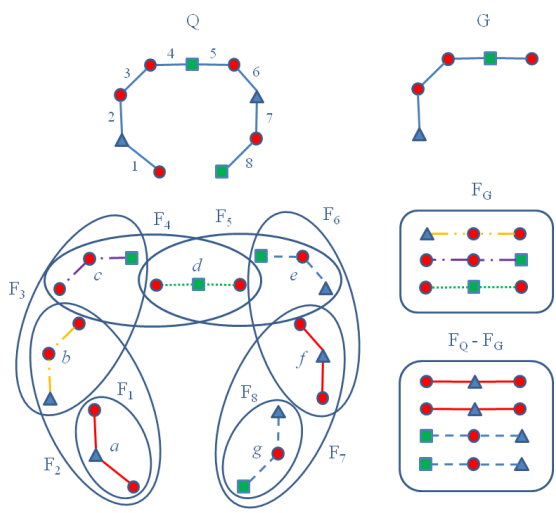

Fig. 5. A graph which cannot be pruned solving the multiset multi-cover problem. Inexact matching with at most two deletions are searched for. Features are subgraphs containing exactly two connected edges. The left side shows the query Q and all its feature occurrences (F_Q). The line type of a feature occurrence is uniquely associated with that feature. Each set F_i indicates all the feature occurrences containing the edge i. The right side shows the target graph G, its multiset of features (F_G) and the multiset of missing features ($F_Q - F_G$). For the multiset multi-cover problem, $\{F_6, F_7\}$ is a cover of $F_Q - F_G$ since the feature f is counted twice. This means that Q is candidate to match G with 2 deletions. Considering f only once (see MOM defined below) the minimum cover would be $\{F_1, F_6, F_7\}$ and G would be discarded.

```
Greedy-MOM(Y, S)
    Z ← U
    Γ ← φ
    while Y ≠ φ do
        X ← argmax_{X̄∈S} |Cov_f(X̄ ∩ Z) ∩ Y)|
        Y ← Y − Cov_f(X ∩ Z)
        Z ← Z − X
        Γ ← Γ ∪ {X}
    return Γ
```

Fig. 6. A greedy algorithm for MOM.

It can be shown that this problem is also NP-hard by reduction from set-cover. A greedy algorithm for it is given in Fig. 6. In the greedy algorithm for MOM in Fig. 6 a further set Z is used to keep track of the covered elements. When a set is added to the cover, its elements are removed from Z in order to avoid considering them twice.

We can now define a new pruning rule based on MOM which is equivalent to pruning rule 1.

Pruning rule 3. Given a query Q with r deletions. Denote as F_e the set of feature occurrences of Q which contain the edge $e \in E(Q)$. A graph G can be discarded if for each $E_\gamma \subseteq E(Q)$ of size r

$$Cov_f\left(\bigcup_{e \in E_\gamma} F_e\right) \not\supseteq F_Q - F_G$$

Since $Cov_f(\bigcup_{e \in E_\gamma} F_e) = F_{E_\gamma}$ we get that

Theorem 2. *Pruning rule* 3.1 *is equivalent to pruning rule* 1.

Theorem 1 and pruning rule 2 apply to the MOM greedy algorithm as well, so the same lower bound can be used to prune the graphs.

4. Experimental Results

To evaluate our filtering methods we applied them to query a large database of molecular compounds and to detect cross-species similarities between protein complexes. We compared our performance to the state-of-the-art Grafil[12] as well as to a baseline filtering method called Edge.[12] The latter simply compares the edges of the query to those of a given graph and discards all graphs that miss (with respect to the query) more edges than the number of allowed deletions. This filtering is in fact equivalent to both our filtering and that of Grafil when considering edge-based features only.

4.1. *Implementation*

Two versions of our tool have been implemented: one is based on the multiset multicover formulation, and the other is based on the MOM formulation. Both tools use Edge as a first pruning step and then apply pruning rule 2. They are compared with our own implementation of Edge and Grafil (which includes Edge as part of the filtering). To perform a uniform analysis, paths of length up to 4 were used as features for all the compared systems. The candidate verification was performed by enumerating all possible subgraphs of the query that can be obtained by deleting any set of r edges, and running an efficient subgraph isomorphism algorithm called VF2[20] over each graph.

4.2. *Benchmark*

We used two query settings. The first, a simulated setting, contained queries of small molecules from the Antiviral Screen Dataset (AIDS).[21] The second, a real setting, contained queries of protein complexes in yeast and human.

The AIDS database contains the topological structures of 42,000 chemical compounds that have been tested for evidence of anti-HIV activity. Each compound of the dataset was converted into a graph where vertices are atoms, edges are bonds between atoms, and the element symbols are used to label the vertices. Multiple bonds were represented by single edges. We obtained a dataset of graphs ranging from 20 to 270 vertices in size. Queries were extracted at random from the AIDS database. The extraction procedure picks a graph and a vertex of that graph at random; it then generates a subgraph starting from the picked vertex and adding edges until a specified size is reached. We generated queries with size ranging between 16 and 48.

The yeast and human protein complex datasets contain graph representations of the set of complexes of each of the species, where vertices correspond to proteins and edges correspond to protein-protein interactions (PPIs). Human complexes were retrieved from CORUM[22] and yeast complexes were retrieved from SGD.[23] The topology of each complex was inferred from PPI data taken from BioGRID.[24] In order to assign labels to the vertices (proteins), we executed an all-pair BLAST on yeast and human proteins, and then clustered them according to the BLAST scores. To this end, we used average-linkage hierarchical clustering with a score cutoff of 40 bits and a maximum cluster size threshold of 500. This procedure yielded 6703 clusters. Each protein was then labeled with the id of the cluster containing it. Removing the complexes with no edges, we obtained a set of 785 human complexes and 284 yeast complexes. We queried the human complexes against the collection of yeast complexes.

4.3. *Results*

We applied all three methods (SIGMA, Grafil and Edge) to the AIDS database with queries of sizes ranging from 16 to 48. We allowed between 1 to 4 deletions and

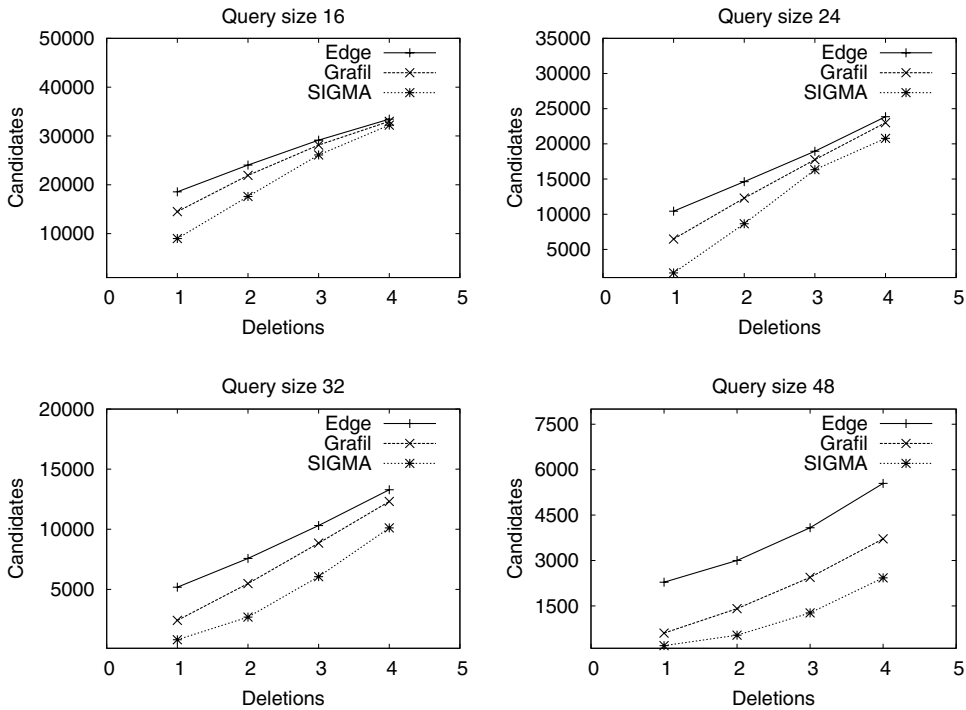

Fig. 7. A comparison of the number of candidates produced by SIGMA, Grafil and Edge. For each query size, the average number of candidates over 100 queries of that size is reported.

tested the filtering power of the different approaches. We tried both variants of our approach, multiset multi-cover and MOM, and got very similar results, hence we report the former only. Compared to multiset multi-cover, MOM tends to generate larger covers, but the computed lower bounds are often less tight. Therefore we did not obtain a significant improvement in pruning power. Moreover, since MOM needs to keep track of each single feature occurrence, the resulting filtering time is higher than the corresponding time obtained by multiset multi-cover. The design of a specific tight lower bound for MOM will be the subject of further investigation. The comparison against Grafil and Edge is depicted in Fig. 7. For a given number of deletions, the average number of candidates over 100 queries is reported. The number of candidates of each query is highly variable, ranging from 1 to the whole dataset. Evidently, SIGMA outperforms the other two methods on all query sizes. The gap tends to increase with the size of the query. A more careful check over each single query has shown that SIGMA outperformed Grafil in more than 95% of the queries.

To evaluate the query processing time and quantify the pruning power, defined as the ratio between the number of verified matches and the number of generated

Table 1. Filtering time and number of candidates (between brackets) obtained by Edge, Grafil and SIGMA over a database of 1000 graphs extracted from the AIDS dataset. The values refer to the average over 10 query executions. All times are expressed in seconds.

Deletions	Edge	Grafil	Sigma
1	0.008 (144.100)	0.056 (96.800)	0.167 (39.700)
2	0.016 (276.600)	0.140 (242.600)	0.327 (142.800)
3	0.049 (371.500)	0.414 (368.700)	0.462 (294.700)
4	0.145 (463.900)	1.136 (461.000)	0.603 (422.100)

Table 2. Number of matches found and overall query time (filtering + matching) performed by Edge, Grafil and SIGMA over a database of 1000 graphs extracted from the AIDS dataset. The values refer to the average over 10 query executions. All times are expressed in seconds.

Deletions	Matches	Edge	Grafil	Sigma
1	8.400	0.860	0.666	0.337
2	36.100	14.010	12.982	9.536
3	106.800	143.737	142.386	129.015
4	181.400	1226.785	1227.513	1176.640

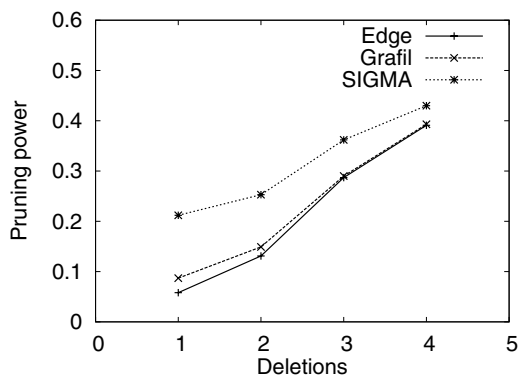

Fig. 8. A comparison of the pruning power of SIGMA, Grafil and Edge.

candidates, we applied an exhaustive search algorithm to part of the data. Specifically, we considered a subset of 1000 compounds and fixed the query size at 16. The results, shown in Tables 1 and 2 and Fig. 8, are expressed as the average over 10 queries. Table 1 reports the filtering time and the number of candidates (between brackets) obtained by the three algorithms. Table 2 reports the number of found matches (number of molecules which contain the query) and the overall query time (filtering + matching) performed by the three algorithms. The pruning power is shown in Fig. 8. On this small dataset, SIGMA exhibits up to fourfold increase in the pruning power.

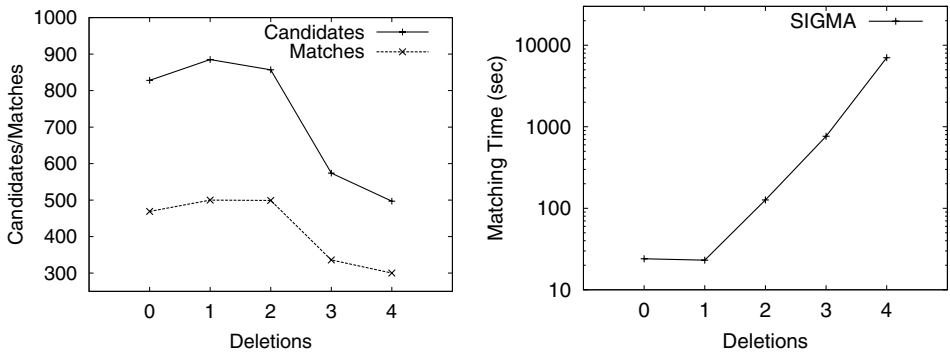

Fig. 9. Performance of SIGMA on a dataset of protein complexes. (a) Reports the number of candidates produced by the algorithm and the number of matches. (b) Reports the query time.

Finally, we applied our algorithm to compare protein complexes between yeast and human. The yeast collection was preprocessed in 93 seconds. Each human complex was then queried against the yeast collection with up to four possible deletions. Figure 9(a) reports the number of matches and candidates found by SIGMA per number of allowed deletions. The number of human-complexes used as queries is 785, the number of yeast-complexes used as targets is 284. During the filtering phase all queries with a number of edges less than 1 have been removed. Figure 9(b) reports the total query time. SIGMA managed to match a total of 336 human protein complexes (1-31 matches per query), obtaining a total of 2104 matches; 439 of the matches were exact and the remaining 1635 were inexact. Some of the most significant matches obtained are reported in Table 3. An exhaustive list can be found in the supplementary material.[25] For example, the "LSm2-8" complex of human matches with the "small nucleolar ribonucleoprotein" complex of yeast with 1 deletion. Figure 10 shows the "LSm2-8" complex of human, the "small nucleolar ribonucleoprotein" complex of yeast and the match between them.

5. Conclusions

We have developed novel graph indexing strategies for inexact graph searches. The resulting tool, called SIGMA, is based on a novel variant of the set-cover problem and a greedy algorithm to approximate its solution.

In extensive tests on a chemical compound database, SIGMA was shown to outperform existing methods for the problem, including the state-of-the-art Grafil. Examining the results in detail, we believe that SIGMA performs better than Grafil because Grafil uses only information about the number of query features that are missing in the graph. In many cases, this criterion is not selective enough. In contrast, SIGMA takes the identity of the features into account, distinguishing between different features, and hence achieves more filtering power. For example, consider

Table 3. Some of the matches obtained querying Human complexes to a database of Yeast complexes. The column Edges refers to the number of edges in the query. The last column reports the number of deletions needed to obtain the match.

Query complex (Human)	Edges	Matching complexes (Yeast)	Deletions
MCM complex	13	MCM complex	0
		DNA replication preinitiation complex	0
		pre-replicative complex	0
18S U11_U12 snRNP	12	ribonucleoprotein complex	2
		small nuclear ribonucleoprotein complex	2
		spliceosome	2
LSm1-7 complex	9	snRNP U6	0
		ribonucleoprotein complex	0
		small nuclear ribonucleoprotein complex	0
		U4 U6 × U5 tri-snRNP complex	0
		spliceosome	0
		snRNP U5	0
		snRNP U1	0
		small nucleolar ribonucleoprotein complex	2
Lsm2-8 complex	8	snRNP U6	0
		small nuclear ribonucleoprotein complex	0
		ribonucleoprotein complex	0
		snRNP U1	0
		U4 U6 × U5 tri-snRNP complex	0
		spliceosome	0
		snRNP U5	0
		small nucleolar ribonucleoprotein complex	1
SMN1-SIP1-SNRP complex	8	ribonucleoprotein complex	1
p27-cyclinE-Cdk2 Ubiquitin E3 ligase(SKP1A-SKP2-CUL1-CKS1B-RBX1) complex	8	ribonucleoprotein complex	3
		preribosome	3
		90S preribosome	4
		transcription factor complex	4
SF3b complex	6	ribonucleoprotein complex	1
		spliceosome	1
		small nuclear ribonucleoprotein complex	2
		snRNP U2	2
12S_U11_snRNP	6	snRNP U5	1
		snRNP U1	1
		ribonucleoprotein complex	1
		small nuclear ribonucleoprotein complex	1
		U4 U6 × U5 tri-snRNP complex	1
		spliceosome	1
		snRNP U5	1
		small nucleolar ribonucleoprotein complex	2

the query in Fig. 11. Compared to the peripheral edges, the central edges are contained in a higher number of feature occurrences, thus they dominate the maximum number of feature misses. As a result, the graph G reported in the figure cannot be discarded by Grafil but is discarded successfully by SIGMA.

Future work will include the management of mismatches and vertex deletions. Although the proposed system can handle vertex deletions by the induced edge

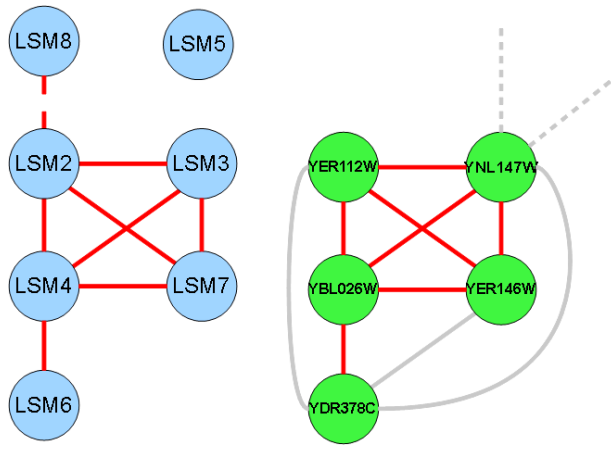

Fig. 10. An example of match between a complex of human and a complex of yeast. The left side represents the human "LSm2-8" complex whereas the right side represents the matching part of the yeast "small nucleolar ribonucleoprotein" complex (composed by 20 nodes and 48 edges). The dashed red line in the left-hand complex represents the missing edge while the red lines in both the left and right hand complexes represent matching edges. Finally, gray lines in the right-hand complex depict edges without a match and the dashed gray lines represent the connections to the remaining part of the yeast complex.

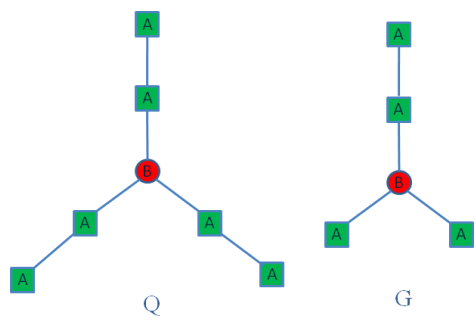

Fig. 11. An example of a graph which is discarded by SIGMA but not by Grafil. We search for the query graph Q with at most 1 deletion, considering paths of length 3 as features. The query contains 3 occurrences of the feature A-A-B and 3 occurrences of A-B-A for a total of 6 feature occurrences. By removing one of the more central edges we miss 3 feature occurrences, while by removing one of the peripheral edges we miss only one feature occurrence. For one allowed deletion, the maximum number of possible feature misses is 3. G misses 2 feature occurrences, thus it cannot be discarded by Grafil. There are no edges of the query which cover the two missing (in G) A-A-B features, thus G is discarded by SIGMA.

deletions, in some applications, the cost of a vertex deletion may not be necessarily related to its degree. In summary, the development of graph indexing methods is essential for efficiently mining biological databases. Methods for inexact matching, like the one reported here, greatly increase the sensitivity of database searches and promise to take a leading role in this area as databases continue to expand.

Acknowledgments

We thank Sharon Bruckner for providing the protein complex datasets and for her help in collecting data. R. Sharan was supported by an Israel Science Foundation grant (No. 385/06). R. Giugno, A. Pulvirenti and A. Ferro were in part supported by PROGETTO FIRB ITALY-ISRAEL grant No. RBIN04BYZ7: Algorithms for Patterns Discovery and Retrieval in discrete structures with applications to Bioinformatics.

References

1. Mongiovì M, Di Natale R, Giugno R, Pulvirenti A, Ferro A, Sharan R, A Set-cover-based approach for inexact graph matching, in *Proc 8th Annual International Conference on Computational Systems Bioinformatics* (*CSB2009*), 2009.
2. Giugno R, Shasha D, GraphGrep: A fast and universal method for querying graphs, in *Proc Int Conf Pattern Recognition* (*ICPR*), pp. 112–115, 2002.
3. James CA, Weininger D, Delany J, Daylight theory manual-Daylight 4.71, 2000.
4. Kelley B, Frowns, http://frowns.sourceforge.net/, 2002.
5. Shasha D, Wang JTL, Giugno R, Algorithmics and applications of tree and graph searching, in *Proc ACM Symposium on Principles of Database Systems* (*PODS*), pp. 39–52, 2002.
6. Ferro A, Giugno R, Mongiovì M, Pulvirenti A, Skripin D, Shasha D, GraphFind: Enhancing graph searching by low support data mining techniques, *BMC Bioinformatics* **9**, 2008.
7. Zhang S, Hu M, Yang J, TreePi: A novel graph indexing method, in *Proc IEEE 23rd Int Conf Data Engineering*, pp. 181–192, 2007.
8. Cheng J, Ke Y, Ng W, Lu A, Fg-index: Towards verification-free query processing on graph databases, in *Proc ACM SIGMOD Int Conf Management of Data*, pp. 857–872, 2007.
9. Yan X, Yu PS, Han J, Graph indexing based on discriminative frequent structure analysis, *ACM Transactions on Database Systems* **30**:960–993, 2005.
10. 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.
11. Bijl D, The serotonin syndrome, *The Netherlands Journal of Medicine* **62**:309–313, 2004.
12. Yan X, Yu PS, Han J, Substructure similarity search in graph databases, in *Proc ACM SIGMOD Int Conf Management of Data*, pp. 766–777, 2005.
13. Tian Y, McEachin RC, Santos C, States DJ, Patel JM, Saga: A subgraph matching tool for biological graphs, *Bioinformatics* **23**:232–239, 2007.
14. He H, Singh AK, Closure-Tree: An index structure for graph queries, in *Proc 22nd Int Conf Data Engineering* (*ICDE'06*), 2006.
15. Bruckner S, Hüffner F, Karp RM, Shamir R, Sharan R, Torque: Topology-free querying of protein interaction networks, *Nucl Acids Res* **37**:106–108, 2009.
16. Dost B, Shlomi T, Gupta N, Ruppin E, Bafna V, Sharan R, Qnet: A tool for querying protein interaction networks, *J Comput Biol* **15**:913–925, 2008.
17. Karp RM, Reducibility among combinatorial problems, *Complexity of Computer Computations* 85–103, 1972.
18. Johnson DS, Approximation algorithms for combinatorial problems, *J Comput System Sci* 256–278, 1974.

19. Rajagopalan S, Vazirani VV, Primal-dual RNC approximation algorithms for (multi)-set (multi)-cover and covering integer programs, in *Proc 34th Annual Symposium on Foundations of Computer Science*, IEEE Computer Society, Palo Alto, CA, USA, pp. 322–331, 1993.
20. Cordella LP, Foggia P, Sansone C, Vento M, An improved algorithm for matching large graphs, in *Proc 3rd IAPR TC-15 Workshop on Graph-based Representations in Pattern Recognition*, pp. 149–159, 2001.
21. NCI DTP Antiviral Screen data, http://dtp.nci.nih.gov/docs/aids/aids_data.html.
22. Ruepp A, Brauner B, Dunger-Kaltenbach I, Frishman G, Montrone C, Stransky M, Waegele B, Schmidt T, Doudieu ON, Stumpflen V, Mewes HW, Corum: The comprehensive resource of mammalian protein complexes, *Nucleic Acids Res* **36**, 2008.
23. Saccharomyces genome database, http://www.yeastgenome.org/, 2008.
24. Stark C, Breitkreutz BJ, Reguly T, Boucher L, Breitkreutz A, Tyers M, BioGRID: A general repository for interaction datasets, *Nucleic Acids Res* **34**, 2006.
25. Supplementary material, http://ferrolab.dmi.unict.it/sigma.html.

Misael Mongiovì received his M.Sc. degree in computer science in 2003, and his Ph.D. in 2007 from the Department of Mathematics and Computer Science at the University of Catania headed by Prof. Alfredo Ferro.

He has been working for the Research and Development Department of Proteo S.p.A. (Catania) taking part in several projects and leading some of them. Currently he is at the University of Catania as a postdoctoral fellow. His research interests lie in the field of data engineering, graph algorithms and bioinformatics.

Raffaele Di Natale received his B.Sc. degree in computer science from University of Catania, Italy in 1997. From 1997 to 2008 his main activities concerned projecting and developing software and in the last years, teaching computer science too. He is a Ph.D. student in BioInformatics at the Department of Biomedical Sciences and the Department of Mathematics and Computer Science of the Catania University.

Rosalba Giugno is an Assistant Professor at the Department of Mathematics and Computer Science at the University of Catania, Italy. She received her B.Sc. degree in computer science from Catania University in 1998 and the Ph.D. in computer science from Catania University in 2003. She has been a visiting researcher at Cornell University, Maryland University and New York University. Her research interests include data mining on structured data and algorithms for bioinformatics.

Alfredo Pulvirenti is an Assistant Professor at the Department of Mathematics and Computer Science at the University of Catania. He received his B.Sc. degree in computer science from Catania University, Italy, in 1999 and the Ph.D. in computer science from Catania University in 2003. He has been a visiting researcher at New York University. His research interests include data mining and machine learning, and algorithms for bioinformatics (sequences and structures).

Alfredo Ferro received his B.Sc. degree in mathematics from Catania University, Italy, in 1973 and a Ph.D. in computer science from New York University in 1981 (Jay Krakauer Award for the best dissertation in the field of sciences at NYU). He is currently professor of computer science at Catania University and has been director of graduate studies in computer science for several years. Since 1989, he has been the director of the International School for Computer Science Researchers (Lipari School http://lipari.cs.unict.it). He is the co-director of the International School on Computational Biology and BioInformatics (http://lipari.cs.unict.it/bio-info/). His research interests include bioinformatics, algorithms for large dataset management, data mining, computational logic and networking.

Roded Sharan obtained his M.Sc. degree from the Hebrew University of Jerusalem, Israel and his Ph.D. from the School of Computer Science, Tel Aviv University, Israel. His doctoral studies under the guidance of Prof. Ron Shamir and later his post-doctoral research work with Prof. Richard Karp at the University of California, Berkeley shaped his interests in bioinformatics, especially in the field of biological networks. He was then offered a senior lecturer position at Tel Aviv University, to where he returned as an Alon Fellow. Subsequently, he was awarded the Raymond and Beverly Sackler Career Development Chair and the Krill Prize from the Wolf Foundation. Today he is an Associate Professor at the Blavatnik School of Computer Science at Tel Aviv University and heads a research group that focuses on the analysis of biological networks. Prof. Sharan has published numerous scientific papers on bioinformatics and graph algorithms. His current research interests include comparative and integrative analysis of biological networks, systems medicine and transcriptional regulation.