Matching matchings

Gábor Bacsó
Laboratory of Parallel and Distributed Systems, MTA SZTAKI
bacso.gabor@sztaki.mta.hu

Anita Keszler
Distributed Events Analysis Research Group, MTA SZTAKI
keszler.anita@sztaki.mta.hu

Zsolt Tuza
Rényi Institute, Budapest; University of Pannonia, Veszprém
tuza@dcs.uni-pannon.hu

***Abstract*—This paper presents the first steps toward a graph comparison method based on matching matchings, or in other words, comparison of independent edge sets in graphs. The novelty of our approach is to use matchings for calculating distance of graphs in case of edge-colored graphs. This idea can be used as a preprocessing step of graph querying applications, to speed up exact and inexact graph matching methods. We introduce the notion of colored matchings and prove some interesting properties of colored matchings in edge colored complete graphs and complete bipartite graphs in case of two colors.**

I. Introduction

Graph based representation has become one of the main directions of modeling in pattern recognition during the last few decades. The main reason of the growing interest in graph based modeling and algorithms is the variety of available graph models leading to expressive and compact data representations. Another motivation is that many graph based pattern recognition methods have low computational cost. For example graph cut based methods [22], [18]) or minimum weight spanning tree based algorithms ([16], [15]) are applied often in computer vision.

Graph comparison is a frequently appearing problem in graph based pattern recognition applications. Graph comparison or as it is often called *graph matching* is an essential part of algorithms applied in image retrieval, or in comparison of molecular compounds, just to mention some application areas. Due to its high importance in theoretical approaches and engineering applications as well, several papers have investigated this topic, see [6].

The main drawback of matching graphs is the computational complexity, since most problems related to this topic belong to the NP-complete problem class.

The idea is that the objects (fingerprints [25], business processes [8], molecular compounds, shapes, etc) are represented by graphs, and the comparison of these objects is done by comparing the corresponding graphs.

As mentioned, matching graphs is a hard problem from algorithmic point of view. Two types of graph matching are usually distinguished: exact and inexact matching. Exact matching is also called graph isomorphism. In case of inexact matching, we do not require the two graphs to be the same, just *similar enough*. This is the reason why these algorithms are often referred to as error tolerant or approximate graph matchings.

The exact subgraph matching for arbitrary graphs is NP-complete [13]. An experimental comparison on the running time of some exact graph matching methods is presented in [11]. However, in case of special graph classes, for example planar graphs, there exist algorithms with polynomial running time [17]. We remark here that the following statement is an old conjecture: the general isomorphism problem is neither polynomial nor NP-complete (it is in NP, of course).

Although several approaches are also known for speeding up isomorphism testing as well - for example a heuristic based method in [21] or [14] using random walks -, in general for arbitrary graphs inexact graph matching methods have become more popular. These methods also have to deal with computational complexity issues (see [2]), but in case of real datasets and applications flexibility and error tolerance are required.

Depending on the application the applied inexact graph matching methods are also varied. In case of image comparison or object categorization simple structures, such as trees are compared (see [23]). Image processing tasks are typical examples for the case when the shape of the graphs can also be important, since vertices have coordinates (see [3]).

However, the most frequently applied approaches are to compare graphs using a distance measure based on graph edit distance ([29], [28]) or a maximum common subgraph ([10]) In case of these metrics, the position of the vertices is irrelevant.

A detailed survey on graph edit distance is presented in [12]. Despite the number of papers that are concerned with this topic, very few contributions can be found in the literature about learning the parameters that control the matching [26], [19].

In [4] the authors analyze the connection between the two distance measures.

Our suggestion is to define a distance function between graphs based on a special type of maximum common subgraph searching: finding the maximum common matching in edge colored graphs.

The paper is organized as follows. In Section II we present some basic definitions and notation. Section III presents our idea of comparing graphs by matching matchings: subsection III-A contains our suggestion in case of graphs without edge colors subsection III-B analyzes the case of edge-colored graphs. Some interesting properties of 2-edge-colored complete and complete bipartite graphs are presented in Section IV. The suggested algorithm for finding colored matchings in l-edge-colored graphs is introduced in Section V with some remarks on special graph classes. Section VI presents test results on evaluating the usefulness of comparing matchings. Section VII concludes our work and also points out to our future goals.

II. Definitions and notation

A simple undirected graph is an ordered pair $G=(V,E)$, where $V=v_1,v_2,..,v_n$ denotes the set of vertices, and $E \subseteq V \times V$ denotes the set of edges. The edge between vertex v_i and v_j is denoted by $(v_i,v_j)=e_{ij}$. A vertex v is incident to edge e, if $v \in e$. The number of vertices is called the order of the graph. Complete graph (or clique) K_n on n vertices is a graph where each vertex pair is connected: $\forall v_i,v_j \in V$, $(v_i,v_j) \in E$. A bipartite graph is a triplet $G=(A,B,E)$. A graph is bipartite if its set of vertices V can be divided into two disjoint sets A,B, such that each edge in E connects a vertex in A to a vertex in B. **Remark** For disconnected bipartite graph, A and B are not unique. The complete bipartite graph $K_{m,n}$, is a bipartite graph, where $|A|=m$, $|B|=n$ and each vertex in A is connected to each vertex in B. In an arbitrary graph two edges are independent, if they do not have a common vertex. A matching is a set of pairwise independent edges. If every vertex of the graph is incident to exactly one edge of the matching, it is called a perfect matching. For further introduction to graph theory and algorithm complexity, see for example [7].

III. Comparing matchings of two graphs

A. Comparing matchings of graphs without edge colors

Finding the largest common subgraph of two graphs is in general an NP-hard problem. Our suggestion is to modify (or specialize) the idea of finding the largest common subgraph to finding the largest common matching of two graphs.

Matchings are an appropriate choice for comparing graphs without colors, since it is relatively easy to find a maximum sized matching. There are polynomial methods for finding the largest (or maximum) matching in a bipartite graph, and in non-bipartite graphs as well (Edmonds-algorithm [9]). These algorithms are also applicable in case of weighted graphs.

Although graphs with maximum matchings of the same size can differ in structure, this measure is suitable to run pre-filtering in graph comparison applications. Recently, the size of the available input datasets have increased rapidly in several areas applying graph-based modeling (web analysis, protein-protein interaction networks, etc.). This naturally requires the development of efficient graph storing and searching techniques. For example graph indexing and querying receives more and more attention, see [31] or [27]. Testing relatively easily computable features of graphs help reducing the search space (branch-and-bound or tree pruning techniques). In our case, a pruning condition is the size of the matching in the query graph and the ones in the graph database. Comparing a simple structural property can speed up exact and inexact graph matching techniques as well.

Let the distance between two graphs be derived from the difference of the size of their maximum matchings. That is, let G_1 and G_2 be two arbitrary graphs. The distance between these graphs is the following:

$$D(G_1,G_2)=abs(|M_1|-|M_2|) \tag{1}$$

where $|M_i|$ is the size of the maximum matching in graph G_i.

B. Comparing matchings of edge colored graphs

Investigation of matching in graphs is an extensively studied topic, however the main directions of research take graphs into consideration without edge colors. One of the novel aspects of our approach is to compare colored matchings as well.

Definition 1. *(In this work) an edge colored - or edge labeled graph (V,E,c) is a graph such that color $c(e_{ij})$ is the color assigned to edge e_{ij}.*

Note that the usual definition contains the following additional condition: edges having a common vertex can not have the same color (proper coloring). The definition here is drastically different.

Edge colored graphs offer more possibilities for comparing matchings, or calculating the distance of graphs based on matchings, than the ones without edge colors. The first idea is to extend Equation 1., to handle more colors, see Equation 2.

$$D1_{color}(G_1,G_2)=\sqrt{\sum_{i=1}^{n_c} w_i(|M_{c_i,1}|-|M_{c_i,2}|)^2} \tag{2}$$

where n_c is the number of colors, c_i is the i^{th} color. $|M_{c_i,j}|$ is the size of the maximum matching in the subgraph of G_j containing only the edges with color c_i. If it is necessary, the colors can also be weighted.

The advantage of this distance calculating method is that the colors are handled separately. The same polynomial algorithm is suitable to find the maximum matching for each color, as in case of graphs without colors on the edges.

However, the drawback is that we gain quite a little information on the correspondence between the edges with different colors. Our suggestion is to use a distance function, that takes into consideration matchings with mixed coloring.

Definition 2. *A colored matching* $(c_1, c_2, ..., c_{n_c}) = (e_1, e_2, ..., e_{n_c})$ *is a matching of* e_i *edges with color* c_i*. For example (yellow,green)=(1,3) is a matching of one yellow and three green edges.*

This definition is somewhat similar to the definition of rainbow matchings [20] (or heterochromatic matchings [30]), however in these type of matchings, no two edges have the same color. In other words a rainbow matching is a $(c_1, c_2, ..., c_{n_c}) = (e_1, e_2, ..., e_{n_c})$ colored matching, where $\forall e_i \leq 1$.

Although there exist interesting theoretical results in case of matchings of not properly edge-colored graphs (Labeled Maximum/Perfect Matching problem, see [5], [1] or [24]) our work aims to solve problems that to the best of our knowledge were not addressed before. The goal of the Labeled Maximum Matching problem is to find a maximum matching in an edge-colored graph with the maximum (or minimum) number of colors in it.

Our work is more general, since we are interested not only in the number of appearing colors in a matching, but the number of edges corresponding to each color as well. The advantage of this approach is that it gives more information on the structure of the colored matchings.

The comparison of edge-colored graphs and the distance calculation between them is based on the distance between their selected colored matchings. Note that these matchings do not necessarily have the same size. The exact method of comparing colored matchings depends on the application and the role of the colors. The colors are weighted in order to handle different importance of edges.

$$Dist(CM_1, CM_2) = \sqrt{\sum_{i=1}^{n_c} w_i(|c_i : CM_1| - |c_i : CM_2|)^2} \quad (3)$$

where $|c_i : CM_j|$ is the number of edges with color c_i in the colored matching CM_j.

If there are no selected colored matchings to represent the graphs, calculation of the distance becomes more complex. Similarly to graph edit distance calculations, the matchings with the smallest distance should be selected. Of course in this case, the size of the matchings should also be taken into consideration.

IV. Comparing matchings of 2-edge-colored graphs K_n and $K_{m,n}$

In this section we will present some properties of the matchings in complete graphs and complete bipartite graphs using two colors. Analyzing these types of graphs helps us to understand the behavior of more general graph classes. Here, we are interested in exact matching of matchings, that is our assumption is that in the query graph we have found a (y,g) matching of y yellow and g green edges, and we would like whether the given colored matching exists in another given colored graph. As mentioned, here our graphs are complete or complete bipartite graphs. It means we know the type of connection (color) between all pair of vertices.

First, we will present a theorem and a short proof on finding (y,g) matchings in complete graphs with a fixed coloring. Then we introduce a rephrased version of the theorem with a longer proof. Although this proof is more complex than the first one and it also depends on parity, nevertheless it has a strong algorithmic nature, and it reveals important properties of the structure of the edge colored graphs, that will be useful in generalizing our theorem.

Preliminary remark Suppose there is a matching with size $y+g$, containing y yellow and g green edges in a graph G. Obviously, for this property, the following is a necessary condition: there is a yellow matching of size y and a green matching of size g in G separately. The condition $2(y+g) \leq n$ is also necessary. Here we investigate the question: When are these conditions sufficient in the complete graph?

A. 2-edge-colored graphs K_n

Theorem 1. *Let* K_n *be an edge colored complete graph with two colors. We have no constraint for the parity of* n*.*

Furthermore, let M *denote a set of edges, that contains a yellow matching of* y *edges and a green matching of* g *edges, where* $y+g < n/2$*. Furthermore, suppose that among all the sets of edges with this property,* M *has the smallest number of vertices belonging to a green and a yellow matching edge as well. Then,* M *is a* (y,g) *matching.*

Proof. In an edge set with the edge coloring introduced above, let the vertices that are incident with a yellow and a green edge called *bad* vertices. Suppose, there exists a vertex x in M which is bad. Let V_M denote the vertices covered by M. $V_M < n$, since $2 \cdot (y+g) < n$, and $V_M < 2 \cdot (y+g)$, otherwise we have found a (y,g) matching.

- If the number of vertices is even ($n = 2t$): at least 3 vertices remain outside V_M.
  Let v_1 and v_2 denote two of the vertices outside V_M. We do not know the color of the edge between these vertices, but it is not important. If it is yellow, then we remove the yellow matching edge in M incident to x, and substitute it with this yellow edge between v_1 and v_2. (If the (v_1, v_2) edge was green, we remove the green edge incident to x). The result is a M' edge set, that consists of a yellow matching of size y and a green matching of size g. This edge set contains at least one less bad vertex than M, which is a contradiction, since M was chosen to be the one with the least bad vertices.
- If the number of vertices is odd ($n = 2t+1$): at least 2 vertices remain outside V_M, so the previous method is appropriate in this case as well.

The proof is complete. □

B. 2-edge-colored graphs $K_{m,n}$

The method of the proof can also be applied in case of complete bipartite graphs. In this way we obtained the following theorem.

Theorem 2. *Let $K_{m,n}$ be an edge colored complete biparite graph with two colors. We have no constraint for the parity of n or m.*

Furthermore, let M denote a set of edges, that contains a yellow matching of y edges and a green matching of g edges, where $y+g < min(m,n)$. Furthermore, suppose that among all the sets of edges with this property, M has the smallest number of vertices belonging to a green and a yellow matching edge as well. Then, M is a (y,g) matching.

The next two subsections present the detailed proof of the rephrased version of Theorem 1. with respect to the parity of n.

C. 2-edge-colored graphs K_n with odd number of vertices

Theorem 3. *Let K_n be an edge colored complete graph with two colors. Furthermore, let the number of vertices be $n = 2t+1$. If there is a yellow matching of size y and green matching of size g separately in K_n so that $y+g \leq t$, then there is a matching with size $y+g$, containing y yellow and g green edges.*

Proof. We know that there exists a yellow matching with size y, moreover, we can find it in polynomial time. Denote this yellow matching with Y. On the remaining vertices we can select some additional edges to the matching with green color. Let us denote this green matching with G', and its size with g'. If $g' = g$, we would have found a (y,g) matching. So let us suppose that $g' < g$. We will prove that if $g' < g$, then G' can be amended with one more green edge, so that we gain a $g'+1+y$ sized matching with $g'+1$ green, and y yellow edges.

There are at least 3 vertices remaining in K_n that are contained neither by Y, nor by G'. The explanation is the following. Since n is odd at least one vertex was left out of the matchings. Besides that, note that $y+g' < t$, so Y and G' contain at most $\leq 2\cdot t-2$ vertices together. Let us denote these remaining vertices with X. Note that all the edges between the vertices in X are yellow, otherwise a green edge could have been selected to increase the size of G', see Fig.1(a).

The other important fact is that all the edges between the vertices in $V(X)$ and $V(Y)$ respectively, are also yellow. (These are the sets of endpoints of the matchings.) The explanation is the following. Let us denote 3 arbitrary vertices in X by v_1, v_2, v_3. Suppose there is a green edge between a $w \in V(Y)$ and $v_1 \in V(X)$ see Fig.1(b). The size of the G' matching can be increased by this green edge. The yellow matching edge with w end vertex can be replaced by the yellow edge between v_2 and v_3, see Fig.1(c).

For the next step we will use the information that there is a green matching with size g in K_n, and we are able to find one in polynomial time. Denote this by G''. Suppose we keep only the edges of G' and G'' in the graph. Furthermore we delete the edges that both matchings contain. Thus, the remaining graph consists of two types of green edges forming alternating paths and circles.

Since $|G''| = g > |G'| = g'$, there exists at least one path with more edges of G'' than of G'. Let P denote one of the alternating paths with this property.

Obviously, the end vertices of P can not be in G'.

Now we will examine the possible positions of the end vertices of P:

- Both end vertices are in X. This way we could have found a larger green matching than G', by replacing the edges of G'' with the ones of G'. This is a contradiction, since we have selected G' to be the maximum sized green matching that amends Y.
- One end vertex is in X, the other one is in Y. By keeping the edges of G'' instead of G' in the alternating path P, we will gain a larger green matching. However, we use one vertex that was the end vertex of a yellow edge in Y. But we are able to replace this edge by one in X the same way as illustrated on Fig.1(c). See the example on Fig.2(a).
- Both end vertices are in Y. If they are in the same yellow matching edge, then we will replace it, as in the previous case (see Fig.2(b)). If the end vertices of the path belong to two yellow matching edges, by increasing the green matching with one, we will lose two yellow matching edges. Since we have proved that between the vertices of Y and X all the edges are yellow, and there are more than two vertices in X, we can restore the yellow matching by replacing the lost yellow edges (see Fig.2(c)).

All the cases have been examined. Thus we have proved that if $|G'| = g' < g$, then there exists one more green edge to amend the matching with. That is, until we reach a matching of y yellow and g green edges, we can always improve the matching. □

D. 2-edge-colored graphs K_n with even number of vertices

Theorem 4. *Let K_n be an edge colored complete graph with two colors. Furthermore, let the number of vertices be $n = 2t$. If there is a yellow matching of size y and a green matching of size g separately in K_n so that $y+g < t$, then there is a matching with size $y+g$, containing y yellow and g green edges.*

Proof. First of all, note that all matchings in K_n of size $< t$ can be extended to a matching of size t. Similarly to the proof of Theorem 3., we know that there exists a yellow matching of size y. However, if the largest yellow

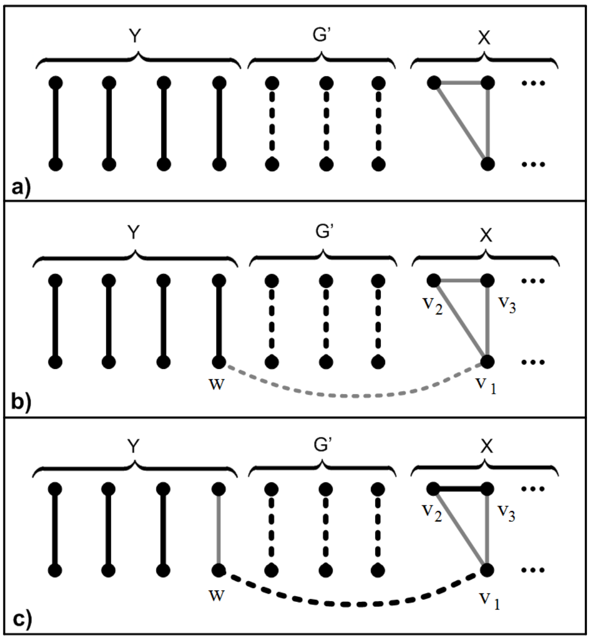


Figure 1: Edges of the matchings are colored black, the other edges are colored grey. a) Y and G': the two matchings, remaining vertex set: X. b) An example: green edge between Y and X. c) Modified matching with y yellow and $g'+1$ green edges.

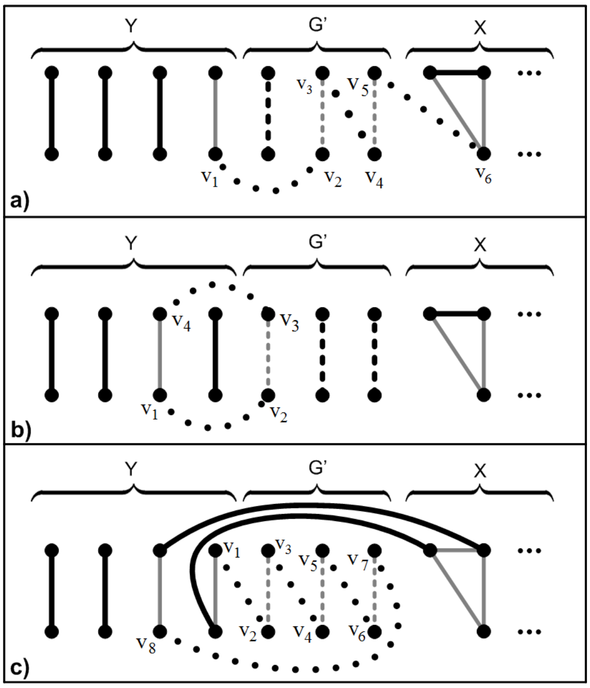


Figure 2: Edges of the matchings are colored black, the other edges are colored grey. a) $v_1,..,v_6$: alternating path with one end in X and one in Y. b) $v_1,..,v_4$ alternating path with end vertices corresponding to one edge in Y. c) $v_1,..,v_8$ alternating path with end vertices corresponding to two edges in Y.

matching in K_n has only y edges, we would be done, since the additional edges to the perfect matching would be necessarily green.

Otherwise, there exists a yellow matching of size $y+1$, which can also be found in polynomial time. Denote this matching by Y. Its role is not the same as above. Let G' denote the largest green matching on the leftover vertices. The size of this matching will be denoted by g', it is smaller than g, similarly as above.

Again, similarly to the proof of Theorem 3., there are remaining vertices, with yellow edges between them (vertex set X), and their number is at least 2. We also know that, in the whole graph, there exists a green matching of size g, denote this by G''. Let P be an alternating path between the edges of G' and G'', as it was in the proof of Theorem 3.. The case partition of the position of the end vertices of P is also analogous with the mentioned proof:

- The two end vertices are in $V(X)$. This way we would have found a green matching of size larger than g', which is a contradiction.
- One of the end vertices (v_1) is in $V(X)$, the other one (v_k) is in $V(Y)$. By replacing the edges of the green matching G' with G'', we gain one green edge, and lose one yellow (the one with v_k as end vertex). But still we have y yellow matching edges.
- If both of the end vertices are in Y, then similarly to the case of odd number of vertices, the Y matching will be decreased by one or two edges. Since X contains at least two vertices, connected by a yellow edge, there is at least one edge to increase the yellow matching with. The size of Y was $y+1$, so at least y yellow edges remain.

We proved that if the G' matching contained less than g edges, we could always extend it with at least one green edge by keeping at least y independent yellow edges □.

Theorem 4 deals with the case when $n = 2 \cdot t$ and $y+g < t$. If $y+g = t$, Theorem 2 does not hold, see the following example.

Example 1. *Let $n = 2t$ and $y+g = t$. Then there exists a complete graph K_n edge colored with two colors, with the following properties. K_n contains a yellow matching of size $y = t-1$ and a green matching of size $g = 1$, but there is no $(y,g) = (t-1,1)$ matching. An example is presented on Figure 3. for $n = 6$, $y = 2$, $g = 1$.*

E. Conclusions of our theorems

Our theorems state that if a yellow matching of size y and a green matching of size g appears in a complete or a complete bipartite graph somewhere, and $y+g < n/2$, then there is a (y,g) colored matching. We have also presented

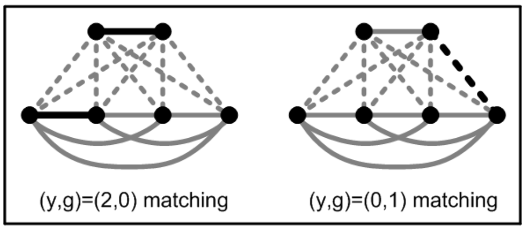


Figure 3: An example graph with 6 vertices, where a yellow 2-matching and a green 1-matching exist, but there is no $(y,g) = (2,1)$ matching.

methods, to find a colored matching with the given property.

Suppose, there are edges in the graph with no information of their colors, and denote this set with T. Our theorems also mean that, if we have found a yellow and a green matching in this graph of the given size, no matter how we choose the colors of the edges in T, the gained colored complete graph will have an (y,g) matching.

V. Algorithm for finding colored matchings in l-edge-colored graphs

In subsection V-C we give an algorithm for finding $(c_1, c_2, .., c_l)$ colored matchings in an l-edge-colored graph, but the first two subsections contain some remarks on colored matchings in case of restrictions on the number of colors and on the graph structure.

A. Perfect colored matchings in 2-edge-colored graphs K_n

Note that perfect matchings can occur only in graphs with even number of vertices. Hence in this subsection we will assume that $n = 2t$. As explained in the previous sections, in case of 2-edge-colored complete graphs, Theorem 1 holds only if $y+g < n/2$ (see Example 1). In this subsection we present an algorithm to decide if there exists a perfect (y,g) colored matching in K_n, that is $y+g = n/2$. The basic idea of the algorithm is the following. Instead of analyzing the K_n graph, we select the edges corresponding to one of the colors, and process the graph induced by these edges.

Assume that the yellow edges were selected. Let $G_y = (V_y, E_y)$ denote the graph induced by the yellow edges. In this graph each matching of size y should be checked if it can be augmented by a green matching of size g.

B. Perfect colored matchings in l-edge-colored graphs K_n

Our conjecture for 3 (or more) colors is that it is NP-hard to decide if a graph has a $(r,y,g,...)$ matching of red, yellow and green, etc. colors even if we have found matchings of these colors of the given size separately.

A simple example is presented on Fig. 4 , with a complete graph colored with 3 colors. There exists a red and a green matching of size one in the graph separately, but there is no $(r,y,g) = (1,0,1)$ colored matching. Note that $r+y+g = 2 < n/2 = 3$, so in case of more than two colors, the existence of a $(r,y,g,...)$ colored matching cannot be guaranteed even if its size is less than $n/2$.

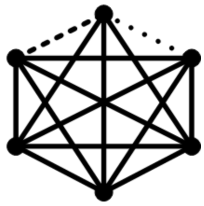


Figure 4: An example graph with 6 vertices and three different colors on the edges. There is a red matching (dotted line) of size one, and a green matching (dashed line) of size one as well, but there is not (r,y,g)=(1,0,1) colored matching in the graph.

However, matchings corresponding to each color are useful in case of inexact graph matching, even if the colors are handled separately. In case of colored matchings, the effectiveness of the comparison depends on the size of the matchings.

C. Algorithm for finding colored matchings

The method presented in Algorithm 1 is based on the recursive function *ColMatch*. The graphs induced by the colors are handled in the different levels of the recursion. Note that ranking the colors can decrease the running time. Colors should be ranked based on the number of their occurrence in the graph. The smaller the number of edges, the faster the algorithm can rule out the existence of the colored matching (if there is no such matching).

Note that before running this algorithm it is worth checking for matchings of the required size in case of each color separately, since it can be carried out by Edmonds's algorithm in polynomial time.

Further simplification of the method in case of special graph classes is in progress.

VI. Test results

Our suggested method for speeding up graph query was tested on a dataset of 'AIDS Screened' chemical structural data available at

$http://dtp.nci.nih.gov/docs/aids/aids\_data.html$. The dataset contains the structure of 42390 chemical compounds. The description of this dataset (number of vertices of the graphs modeling the compounds and the corresponding maximum matchings) is presented on Fig. 5. For a fixed number of vertices the size of the maximum matchings might be different. The small histograms show the distribution of the size of the maximum matchings in case of 30,50,75 and 100 vertices. As the number of vertices raises the deviation of the size of the maximum matchings also increases.

Tests were carried out on this dataset in order to evaluate the efficiency of using maximum matching as a descriptor of graphs. Each graph in the dataset was used as query to search

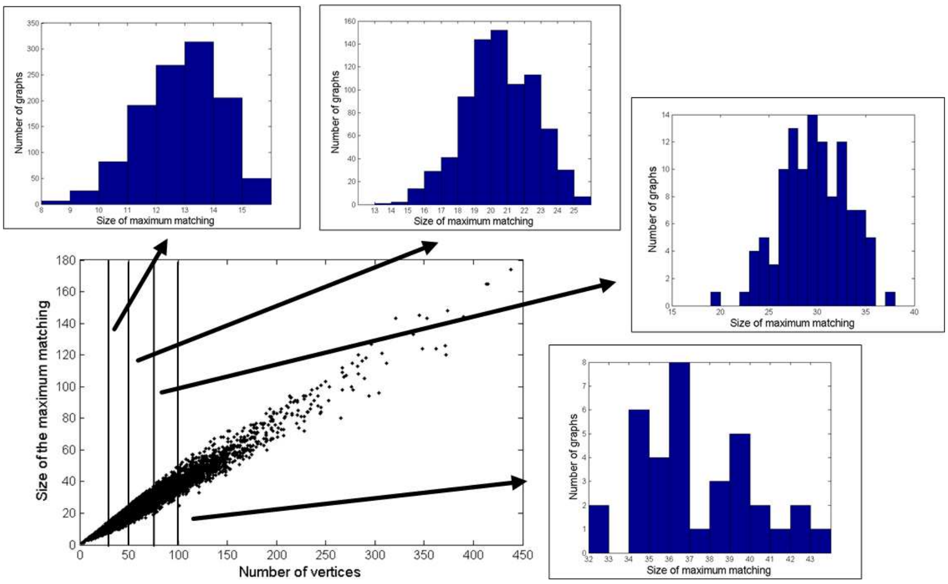


Figure 5: Description of the test dataset. For 42390 chemical compounds the size of the graphs and the size of the corresponding maximum matchings are visualized. Detailed description for graphs with 30,50,75,100 vertices is also presented. Each histogram shows the distribution of the size of the maximum matchings for graphs with 30,50,75,100 vertices.

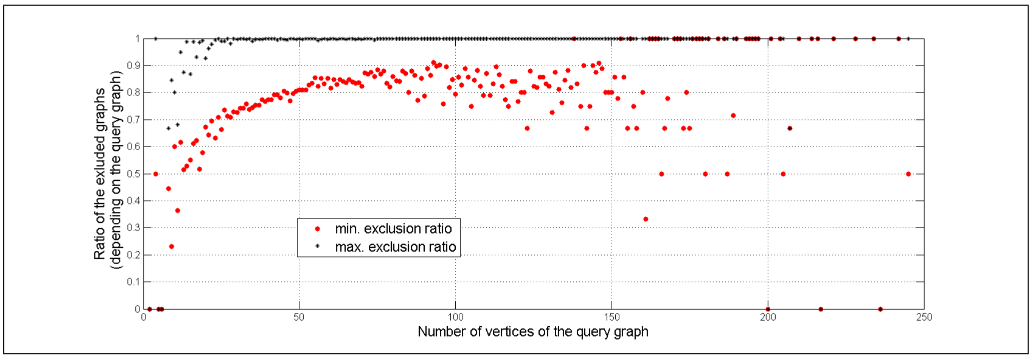


Figure 6: Test results on the dataset described on Fig. 5. Suppose that the query graph has n vertices. This figure shows the ratio of the graphs with n vertices that can be excluded based on their maximum matching. Tests were carried out with each graph selected as query. The black stars and the red dots show the best and the worst exclusion ratios among the graphs with a given number of vertices, respectively.

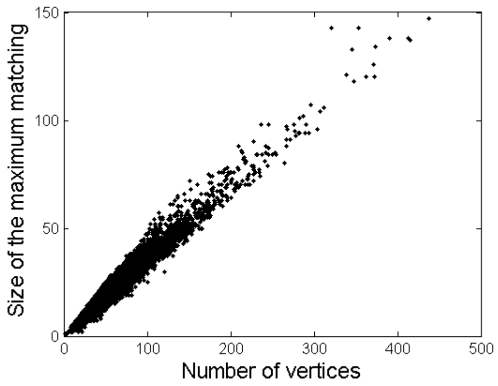
the dataset. Since the number of vertices is a property that is easy to be checked, we only ran the query within graphs of the same order.

Test results on the exclusion ratio, i.e. the ratio of the graphs excluded by the query within graphs of the same order are presented on Fig. 6. The exclusion ratio (ER) was computed the following way: $ER(G) = 1 - \frac{N_M - 1}{N_V - 1}$. N_V is the number of graphs of the database with the same order as graph G. N_M is the number of graphs with the same order as G in what the corresponding matching has the same size as in case of G.

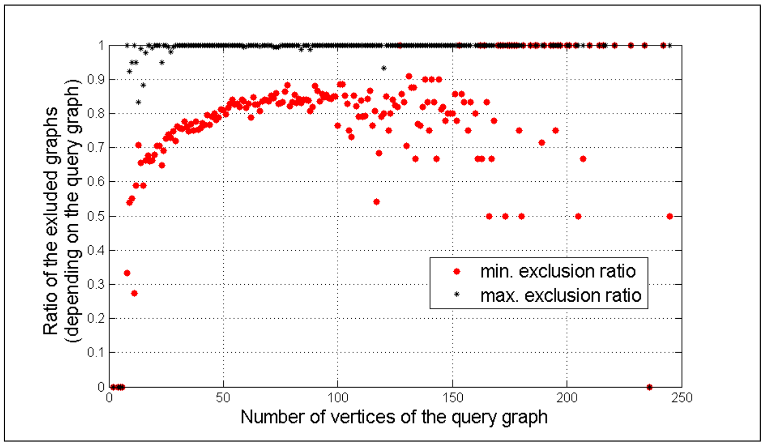
A query was run with each graph and for all different graph orders, the best and the worst result is shown on the figure marked with black and red, respectively. A query is considered to be better than another, if the corresponding exclusion ratio is higher, i.e. the larger number of graphs could be excluded.

With a few exceptions, even the worst excluding ratios (red marks) reach 0.5, that is, at least half of the graphs of a given order can be excluded regardless of the selected query graph.

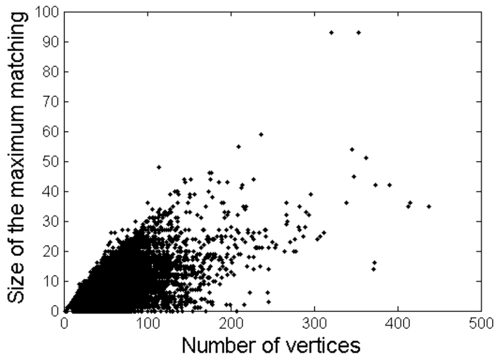
Two types of edges are marked in the database depending on the strength of the connection between the elements of the compounds. For further analysis, the types (labels) of the edges are also taken into consideration. For each 2-edge-labeled graph, two new graphs were generated keeping



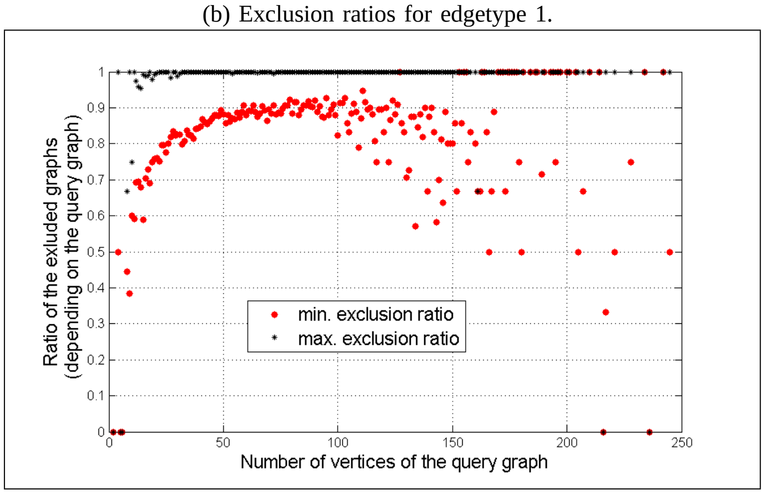
(a) Maximum matchings in the graphs of edgetype 1.



(b) Exclusion ratios for edgetype 1.



(c) Maximum matchings in the graphs of edgetype 2.



(d) Exclusion ratios for edgetype 2.

Figure 7: Distribution of the maximum matchings in the graphs of edge types 1 (a) and 2 (c). Corresponding exlusion ratios on (c) and (d) respectively.

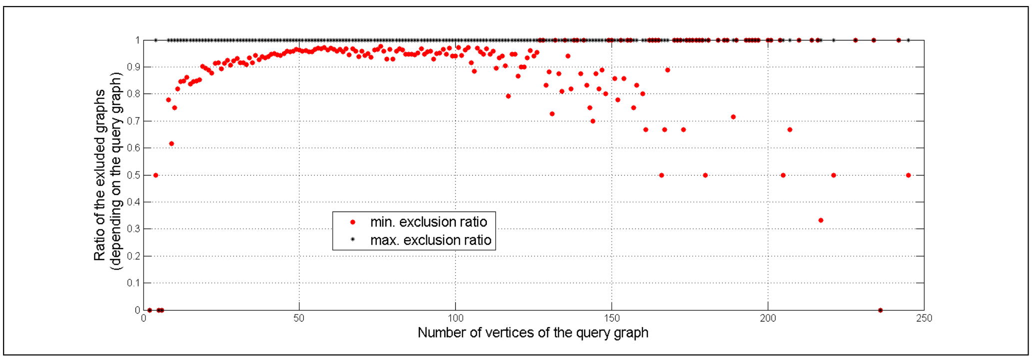


Figure 8: Best (red) and worst (black) exclusion ratios based on the colored matchings (output of Algorithm 1.)

Algorithm 1 Finds a $(c_1, c_2, c_3, ..., c_l)$ matching in l-edge-colored arbitrary graphs (if exists).

function ISINDEPENDENT(e_1, e_2)
  if $e_1 \cap e_2 = \emptyset$ **then return** true
  elsereturn false
  end if
end function

function COLMATCH($E_{rem}, M, Size, Color, level$)
  $M_{level} = \{e \in M | c(e) = Color(level)\}$;
  if $|M_{level}| = Size(level)$ **then**
    if $|Color| = level$ **then return** M
    else
      $l = level + 1$;
      Res =COLMATCH($E_{rem}, M, Size, Color, l$);
      return Res
    end if
  else
    $E_{level} = \{e \in E_{rem} | c(e) = Color(level)\}$;
    for $i = 1; i \leq |E_{level}|; i++;$ **do**
      if ISINDEPENDENT($M, E_{level}(i)$) **then**
        $R = E_{rem} \setminus E_{level}(i)$;
        $E' = \{e \in R | e \cap E_{level}(i) \neq \emptyset\}$;
        $R = R \setminus E'$;
        $m = M \cup E_{level}(i)$;
        Res =COLMATCH($R, m, Size, Color, level$);
        if $Res \neq \emptyset$ **then return** Res
        end if
      end if
    end for
    return $\emptyset$
  end if
end function

function MAIN($E, Size, Color$)
  $level = 1$; $E_{rem} = E$; $M = \emptyset$;
  Res =COLMATCH($E_{rem}, M, Size, Color, level$);
  if $Res \neq \emptyset$ **then** Output: Res
  elseOutput: No such matching.
  end if
end function

only the edges of type 1 and 2, respectively. The maximum matchings (Figs. 7a, 7c) and the exclusion ratios (Figs. 7b, 7d) were also computed for these new graphs as in the unlabeled case. The results clearly show that matchings of edges of type 2 tend to be more unique. Due to this, the corresponding exclusion ratios are tend to be higher than in case of edge type 1.

Another interesting conclusion of the tests are the results of the 2-edge-labeled case, where colored matchings were compared. Algorithm 1 was run to compute the colored matchings. Since the edges of type 2 performed better, this color was chosen at first. The exclusion ratios are presented on Fig. 8.

The worst exclusion ratios clearly outperform the ones corresponding to the unlabeled case. The tests confirm that colored matchings perform better than standard ones, however these are more complicated to compute.

VII. CONCLUSION

We have presented the first steps toward a graph matching method based on comparison of matchings. Our aim was to introduce a novel approach to compare graphs even if their edges are colored (or labeled). Our suggestion is to use matchings of graphs as a basis of distance measures, to overcome some of the complexity issues of graph comparison. We have shown interesting properties of colored matchings in case of two colors. We have analyzed the circumstances of the appearance of colored matchings using the well known method of finding matchings in graphs without edge colors. An algorithm was suggested to find colored matchings in l-edge-colored graphs. Test were run on a dataset of chemical compounds. We have shown that comparing matchings is a useful descriptor in graph comparison in this application field. Our goal in the future is the further analysis of the properties of edge colored graphs in case of more than two colors, concerning algorithmic complexity as well.

ACKNOWLEDGEMENTS

This work has been partially supported by Hungarian Scientific Research Fund grants 81493 and 80352.

REFERENCES

[1] (2005). *On Complexity and Approximability of the Labeled Maximum/Perfect Matching Problems*, volume 3827 of *LNCS*. Springer.

[2] Abdulkader, A. M. (1998). *Parallel Algorithms for Labelled Graph Matching*. PhD thesis, Colorado School of Mines.

[3] Bai, X. and Latecki, L. (2008). Path similarity skeleton graph matching. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 30(7):1282 –1292.

[4] Bunke, H. (1997). On a relation between graph edit distance and maximum common subgraph. *Pattern Recognition Letters*, 18(8):689 – 694.

[5] Carrabs, F., Cerulli, R., and Gentili, M. (2009). The labeled maximum matching problem. *Computers & OR*, 36(6):1859–1871.

[6] Conte, D., Foggia, P., Sansone, C., and Vento, M. (2004). Thirty years of graph matching in pattern recognition. *IJPRAI*, pages 265–298.

[7] Cormen, T. H., Stein, C., Rivest, R. L., and Leiserson, C. E. (2001). *Introduction to Algorithms*. McGraw-Hill Higher Education, 2nd edition.

[8] Dijkman, R., Dumas, M., and García-Bañuelos, L. (2009). Graph matching algorithms for business process model similarity search. In *Proc. 7th Int. Conf. on BPM'09*, pages 48–63, Berlin, Heidelberg. Springer-Verlag.

[9] Edmonds, J. (1965). Paths, trees, and flowers. *Canad. Journal of Mathematics*, 17:449–467.

[10] Fernandez, M. L. and Valiente, G. (2001). A graph distance metric combining maximum common subgraph and minimum common supergraph. *Pattern Recognition Letters*, 22(6):753–758.

[11] Foggia, P., Sansone, C., and Vento, M. (2001). A performance comparison of five algorithms for graph isomorphism. In *Proc. 3rd IAPR TC-15 Workshop on Graph-based Representations in Pattern Recognition*, pages 188–199.

[12] Gao, X., Xiao, B., Tao, D., and Li, X. (2010). A survey of graph edit distance. *Pattern Analysis and Applications*, 13:113–129.

[13] Garey, M. R. and Johnson, D. S. (1990). *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA.

[14] Gori, M., Maggini, M., and Sarti, L. (2005). Exact and approximate graph matching using random walks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(7):1100–1111.

[15] Grygorash, O., Zhou, Y., and Jorgensen, Z. (2006). Minimum spanning tree based clustering algorithms. In *18th IEEE Int. Conf. on Tools with Artificial Intelligence, 2006. ICTAI '06.*, pages 73 –81.

[16] Haxhimusa, Y. and Kropatsch, W. (2004). Segmentation graph hierarchies. In *Structural, Syntactic, and Statistical Pattern Recognition*, volume 3138 of *LNCS*, pages 343–351. Springer Berlin, Heidelberg.

[17] Hopcroft, J. E. and Wong, J. K. (1974). Linear time algorithm for isomorphism of planar graphs. In *Proceedings of 6th STOC '74*, pages 172–184, New York, NY, USA. ACM.

[18] Ladicky, L., Russell, C., Kohli, P., and Torr, P. (2010). Graph cut based inference with co-occurrence statistics. In *Computer Vision ECCV 2010*, volume 6315 of *Lecture Notes in Computer Science*, pages 239–253. Springer Berlin,Heidelberg.

[19] Leordeanu, M. and Hebert, M. (2009). Unsupervised learning for graph matching. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 864 –871.

[20] LeSaulnier, T. D., Stocker, C., Wenger, P. S., and West, D. B. (2010). Rainbow matching in edge-colored graphs. *Electr. J. Comb.*, 17(1).

[21] Lipets, V., Vanetik, N., and Gudes, E. (2009). Subsea: an efficient heuristic algorithm for subgraph isomorphism. *Data Mining and Knowledge Discovery*, 19:320–350. 10.1007/s10618-009-0132-7.

[22] Liu, X., Veksler, O., and Samarabandu, J. (2010). Order-preserving moves for graph-cut-based optimization. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 32(7):1182 –1196.

[23] Macrini, D., Dickinson, S., Fleet, D., and Siddiqi, K. (2011). Object categorization using bone graphs. *Comput. Vis. Image Underst.*, 115:1187–1206.

[24] Monnot, J. (2005). The labeled perfect matching in bipartite graphs. *Inf. Process. Lett.*, 96(3):81–88.

[25] Neuhaus, M. and Bunke, H. (2005). A graph matching based approach to fingerprint classification using directional variance. In *In: Proc. 5th Int. Conf. on Audio-and Video-Based Biometric Person Authentication. LNCS 3546*, pages 191–200. Springer.

[26] Neuhaus, M. and Bunke, H. (2007). Automatic learning of cost functions for graph edit distance. *Information Sciences*, 177(1):239 – 247.

[27] Pal, D. and Rao, P. R. (2011). A tool for fast indexing and querying of graphs. In *Proc. 20th Int. Conf. Companion on World Wide Web*, WWW '11, pages 241–244.

[28] Raveaux, R., Burie, J.-C., and Ogier, J.-M. (2010). A graph matching method and a graph matching distance based on subgraph assignments. *Pattern Recognition Letters*, 31:394–406.

[29] Riesen, K. and Bunke, H. (2009). Approximate graph edit distance computation by means of bipartite graph matching. *Image and Vision Computing*, 27(7):950 – 959.

[30] Wang, G. and Li, H. (2008). Heterochromatic matchings in edge-colored graphs. In *The electronic journal of combinatorics 17.*

[31] Zhu, L., Ng, W. K., and Cheng, J. (2011). Structure and attribute index for approximate graph matching in large graphs. *Information Systems*, 36(6):958 – 972.