Median Graph: A New Exact Algorithm Using a Distance Based on the Maximum Common Subgraph

M. Ferrer[a,*], E. Valveny[a], F. Serratosa[b]

[a] *Centre de Visió per Computador, Departament de Ciències de la Computació. Universitat Autònoma de Barcelona, 08193 Bellaterra, Spain*

[b] *Departament d'Enginyeria Informàtica i Matemàtiques, Universitat Rovira i Virgili, 43007 Tarragona, Spain*

Abstract

Median graphs have been presented as a useful tool for capturing the essential information of a set of graphs. Nevertheless, computation of optimal solutions is a very hard problem. In this work we present a new and more efficient optimal algorithm for the median graph computation. With the use of a particular cost function that permits the definition of the graph edit distance in terms of the maximum common subgraph, and a prediction function in the backtracking algorithm, we reduce the size of the search space, avoiding the evaluation of a great amount of states and still obtaining the exact median. We present a set of experiments comparing our new algorithm against the previous existing exact algorithm using synthetic data. In addition, we present the first application of the exact median graph computation to real data and we compare the results against an approximate algorithm based on genetic search. These experimental results show that our algorithm outperforms the previous existing exact algorithm and in addition show the potential applicability of the exact solutions to real problems.

Key words: Median Graph, Maximum Common Subgraph, Minimum Common Supergraph, Graph Matching

* Corresponding author: Tel.: +34 93 581 23 01 / Fax.: +34 93 581 16 70
Email address: `mferrer@cvc.uab.cat` (M. Ferrer).

1 Introduction

In structural pattern recognition, graphs have been shown as a useful data structure to represent complex objects. The nodes of the graph typically represent parts of the object and the edges represent the relations between them. This high representational power of graphs has been shown very useful in many areas of computer vision and pattern recognition such as character recognition [18], shape analysis [22] and 3D object recognition [27]. Nevertheless, graphs have the main drawback of its combinatorial nature. The comparison between two objects that is quite simple when they are represented by feature vectors for instance, becomes highly complex when they are represented using graphs. The time required by any of the exact algorithms to compare two graphs may, in the worst case, become exponential in the size of graphs. Approximate algorithms in contrast, have polynomial time complexity, but do not guarantee to give the optimal solution.

In this paper we are interested in inferring the representative of a set of objects given a set of noisy samples of such object. In the graph domain it is not easy to define the representative of a set. In this context, the generic concept of *median*, which has been widely used in areas such as statistics and probability theory, turns out to be very useful. In the structural domain the *median graph* [16] has been defined in order to represent this concept. Given a set of graphs, the median graph is defined as the graph that has the smallest sum of distances (SOD) to all graphs in the set. However, the computation of the median graph is exponential both in the number of input graphs and their size [9]. A number of algorithms have been presented in the past to compute the generalized median graph. The only exact algorithm proposed up to now [20] is based on an A^* algorithm handled by a data structure called multimatch. As the computational cost of this algorithm is very high, a set of approximate algorithms have also been developed in the past based on different approaches such as genetic search [16,20], greedy algorithms [15] and spectral graph theory [12,26].

The main contributions of this paper are twofold. Firstly, from a theoretical point of view, we will show that, under a particular cost function and a graph edit distance based on the maximum common subgraph, both introduced in [3], the search space for the median graph can be drastically reduced. In addition, using the same conditions, we have also defined an heuristic strategy in order to avoid the exploration of some states of this new search space. Secondly, based on this theoretical result, we will present a new exact and more efficient algorithm for the computation of the generalized median graph.

To validate the theoretic part, we have carried out a set of experiments comparing our new algorithm with the previous existing exact algorithm using synthetic data. We will show how the new algorithm clearly outperforms the multimatch algorithm. In addition, we have applied the new algorithm to a set of real data using a database of graphs representing molecules. In this case we have compared our approach with an approximate algorithm based on genetic search. We will show how the new exact algorithm compares well in terms of computation time with the approximate algorithm while it is clearly better in terms of accuracy of the median graph. This result suggests that the exact computation of the median graph can be extended for the first time (in a limited way) to real applications. The previous existing algorithms could only be applied to synthetic datasets containing only a small number of small graphs. Moreover, and in spite of the complexity, to be able to obtain exact solutions for the median graph can be useful to better understand the nature of such concept and may help to improve the existing approximate algorithms or to introduce new approximate solutions.

The rest of the paper will be as follows. In section 2 we define some basic concepts and introduce our notation. Section 3 is devoted to explain the concept of median graph and the different algorithms to compute it. The new algorithm, including the reduction of the search space and the heuristic prediction function is described in section 4. Experimental results are shown in section 5 and, finally the main conclusions are stated in section 6.

2 Definitions and notation

2.1 Basic definitions

Let L_V and L_E denote the set of node and edge labels, respectively. A **graph** is a triple $g = (V, \alpha, \beta)$ where, V is the finite set of nodes, α is the node labeling function ($\alpha : V \longrightarrow L_V$), and β is the edge labeling function ($\beta : V \times V \longrightarrow L_E$). We assume that our graphs are fully connected. Consequently, the set of *edges* is implicitly given (i.e. $E = V \times V$). Such assumption is only for notational convenience, and it doesn't impose any restriction in the generality of our results. In the case where no edge exists between two given nodes, we can include the special null label ε in the set of labels L_E to model such situation. If $V = \phi$, then g is called the *empty graph*. Finally, the number of nodes of a graph g is denoted by $|g|$.

Given two graphs, $g_1 = (V_1, \alpha_1, \beta_1)$ and $g_2 = (V_2, \alpha_2, \beta_2)$, g_2 is a **subgraph** of g_1, denoted by $g_2 \subseteqq g_1$ if,

- $V_2 \subseteqq V_1$
- $\alpha_2(v) = \alpha_1(v)$ for all $v \in V_2$

- $\beta_2((u,v)) = \beta_1((u,v))$ for all $(u,v) \in V_2 \times V_2$

From this definition, it follows that, given a graph $g_1 = (V_1, \alpha_1, \beta_1)$, a subset $V_2 \subsetneqq V_1$ of its vertices uniquely defines a subgraph. Such subgraph is called the subgraph *induced* by V_2.

Finally, it is important to be able to check whether two graphs are identical or not. The isomorphism between two graphs permits to check such a condition. Given two graphs $g_1 = (V_1, \alpha_1, \beta_1)$, and $g_2 = (V_2, \alpha_2, \beta_2)$, a **graph isomorphism** between g_1 and g_2 is a bijective mapping $f : V_1 \longrightarrow V_2$ such that,

- $\alpha_1(x) = \alpha_2(f(x))$ for all $x \in V_1$
- $\beta_1((x,y)) = \beta_2((f(x), f(y)))$ for all $(x,y) \in V_1 \times V_1$

If there exists a graph isomorphism between two given graphs g_1 and g_2, we say that g_1 and g_2 are isomorphic.

2.2 Minimum Common Supergraph

Let g_1, g_2 and g_3 be three graphs. If both g_1 and g_2 are subgraphs of g_3 then g_3 is called a **common supergraph** of g_1 and g_2 [7]. Generalizing the same idea for a set $S = \{g_1, g_2, ..., g_n\}$, a graph g_{m_S} is called a **minimum common supergraph of S** (also denoted by mcs(S)) if $\{g_1, g_2, \cdots, g_n\}$ are subgraphs of g_{m_S} and there is no other common supergraph of $\{g_1, g_2, \cdots, g_n\}$ having less nodes than g_{m_S}. A set of three graphs and a possible minimum common supergraph of them is shown in Figures 1(a) and 1(b) respectively.

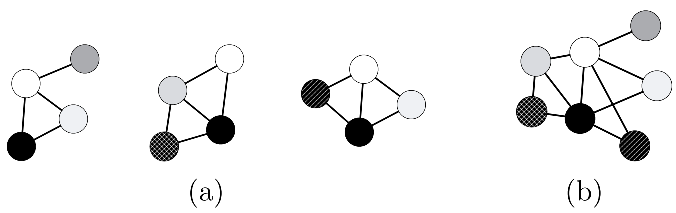


Fig. 1. A set S of three graphs (a), and a possible mcs(S) of them (b).

2.3 Maximum Common Subgraph

Let g_1 and g_2 be two graphs, and $g'_1 \subsetneqq g_1$, $g'_2 \subsetneqq g_2$. If there exists a graph isomorphism between g'_1 and g'_2 then, both g'_1 and g'_2 are called a **common subgraph** of g_1 and g_2. A graph g_M is called a **maximum common subgraph** (usually denoted by MCS(g_1,g_2)) of g_1 and g_2 if g_M is a common subgraph of

g_1 and g_2 and there is no other common subgraph of both g_1 and g_2 having more nodes than g_M.

An example of two graphs g_1 and g_2 and a possible $MCS(g_1, g_2)$ between them is shown in Figure (2). Notice that according to the definition of *graph* given in section 2.1 these graphs are fully connected. The "null" or unexisting edges are represented by dashed lines in the figure. In addition, notice that according to the definition of *subgraph* given in section 2.1, g_2 is not a subgraph of g_1 because g_2 has no edge between the white and black nodes and therefore, given the same set of nodes in g_1 and g_2, g_2 have a different set of edges. Thus, a possible $MCS(g_1, g_2)$ is the graph g_M shown in Figure 2(c). In this case, g_M consists also of a subset of nodes of both g_1 and g_2, but in this case the edges connecting such a subset is the same in g_1 and in g_2. Thus, since g_M is an induced subgraph of both g_1 and g_2 and there is no other subgraph of these two graphs having more nodes, g_M is a possible $MCS(g_1, g_2)$. This fact will be an important point to be taken into account in Section 4.

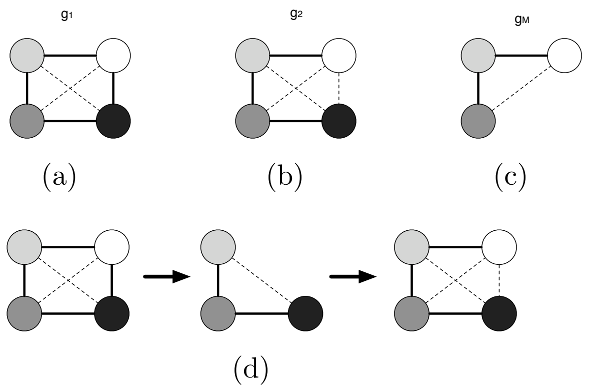


Fig. 2. Two graphs g_1 (a) and g_2 (b), a possible $MCS(g_1, g_2)$ (c) and an edit path between g_1 and g_2 (d).

In the last years some papers have been presented related to the MCS computation based on different approaches and algorithms [2,11,19,25]. An explanation of such methods and a comparison between them can be found in [5] and [10].

2.4 Graph Edit Distance

The graph edit distance [4,24], has been shown as one of the most widely used methods to compute the dissimilarity between two graphs.

The basic idea behind the graph edit distance is to define the dissimilarity of two graphs as the minimum amount of distortion required to transform one graph into the other. To this end, a number of distortion or edit operations e, consisting of the insertion, deletion and substitution of both nodes and edges are defined. Given these edit operations, for every pair of graphs, g_1 and g_2, there exists a sequence of edit operations, or edit path $p(g_1, g_2) = (e_1, \ldots, e_k)$ (where each e_i denotes an edit operation) that transforms g_1 into g_2. In general, several edit paths exist between two given graphs. This set of edit paths is denoted by $\wp(g_1, g_2)$. In order to quantitatively evaluate which edit path is the best, edit costs are introduced. The basic idea is to assign a penalty cost c to each edit operation according to the amount of distortion it introduces in the transformation. The edit distance d between two graphs g_1 and g_2, denoted by $d(g_1, g_2)$, is given by the minimum cost edit path that transforms one graph into the other.

2.4.1 Graph edit distance and the Maximum Common Subgraph

Several exact and approximate algorithms have been presented in the past related to the computation of the distance between two graphs [17,21,23]. In this work, we will use a particular cost function [3], that allows to define the edit distance of two graphs g_1 and g_2 in terms of their MCS.

Table 1 summarizes this cost function.

Table 1
Detail of the cost function

Operations on nodes	**Cost**
Node deletion $c_{nd}(u)$	always equal to 1
Node insertion $c_{ni}(u)$	always equal to 1
Node substitution $c_{ns}(u, v)$	0 if $\alpha_1(u) = \alpha_2(v)$; ∞ otherwise

Operations on edges	**Cost**
Edge deletion $c_{ed}(e)$	always equal to 0
Edge insertion $c_{ei}(e)$	always equal to 0
Edge substitution $c_{es}(e_1, e_2)$	0 if $\beta_1(e_1) = \beta_2(e_2)$; ∞ otherwise

Where α_i and β_i are the node and edge labeling functions of a graph g_i.

In such particular cost function, deletions and insertions of nodes have always a cost of 1, deletions and insertions of edges have always a cost of 0, and node and edge substitutions take the values of 0 or ∞ depending on whether the substitution is identical or not, respectively. Note that, from the definition of graph given in Section 2.1 stating that graphs are fully connected, edge deletions only occur when a node is deleted and then all its incident edges are deleted too. Other situations as in the case of the graphs of Figure 2(a) and 2(b) are treated as a substitution of the edge by a null-labeled edge, not as an edge deletion. The practical consequences of this observation will be explained in more detail in Section 4.1.

Under such cost function, the edit distance $d(g_1, g_2)$ between two given graphs g_1 and g_2 is related to their MCS in the following way:

$$d(g_1, g_2) = |g_1| + |g_2| - 2\,|MCS(g_1, g_2)| \tag{1}$$

This result demonstrates the intuitive idea that the more two graphs have in common, the lower their distance. Beyond the theoretical implications of this result there are also important practical implications. For instance, an immediate consequence of this result is that any algorithm that computes the graph edit distance may be used for maximum common subgraph computation if it is run under this cost function.

More recently, the same cost function has been used in [8], to establish the relation between the MCS and the mean of two graphs. Concretely, it is shown that, under this cost function, the MCS of two graphs is also the mean, that is the graph that minimizes the sum of distances to these two graphs. Finally, in [7] it is shown the relation between the MCS and the minimum common supergraph (mcs) of two graphs. In that work, it is demonstrated that under an extensive family of cost functions, including this particular cost function, the computation of the mcs can be done from the computation of the MCS.

It is therefore clear that this cost function has important potential applications in different graph matching problems. Thus, in the rest of the paper we will assume, that the distance between two graphs is computed according to the Equation (1).

3 Generalized Median Graph

Given a set of graphs, the concept of median graph has been presented as a useful tool to compute a representative of such a set. Let U be the set of graphs that can be constructed using labels from L. Given $S = \{g_1, g_2, ..., g_n\} \subseteq U$, the **generalized median graph** of S is defined as the graph $\bar{g} \in U$ such that its sum of distances (SOD) to all the graphs in S is minimum:

$$\bar{g} = arg \underset{g \in U}{min} \sum_{g_i \in S} d(g, g_i) \tag{2}$$

Notice that $\bar{g}$ is not usually a member of S, and in general more than one generalized median graph can be found for a given set S.

As shown in equation (2) a graph distance between every candidate median g and each graph $g_i \in S$ must be computed. Since the computation of the graph edit distance is a well-known problem which is NP-complete in terms of complexity, the computation of the generalized median graph can only be done in exponential time, both in the number of graphs in S and their size. As a consequence, in real applications we are forced to use suboptimal methods in order to obtain approximate solutions for the generalized median graph in a reasonable time. Such approximate methods generally apply some kind of heuristics in order to reduce the complexity of the computation of the graph edit distance and the size of the search space respectively.

Both exact [20] and approximate [12,15,16] algorihtms have been presented in the past for the median graph computation. In this work, we are interested in the exact median computation.

The only optimal algorithm presented up to now [20], is based on a combinatorial search exploiting the fact that the number of nodes of the candidate median must be in between 0 and the sum of nodes of all graphs in the set S [16]. The algorithm implements an A^*-based search procedure handled by a structure called multimatch. In such structure, a simultaneous transformation (edit operations) from a candidate median graph to all the graphs in the set is encoded. The operations over the edges are implicitly given by such structure. The labels for each node/edge are selected from the set of labels in such a way that the SOD of the encoded candidate median to all the graphs is minimized. Unfortunately, this approach suffers from a high complexity. The results presented in [9] show that for a set of two graphs having 6 nodes each, the time needed to compute the median graph grows up to 10^4 seconds (all the experiments were run on an IBM Thinnode Power 2 computer).

4 New Exact Algorithm for the Generalized Median Graph

In this section we will present a new and more efficient algorithm for the exact median graph computation. To do this, we will tackle the problem from two different angles. Firstly, in section 4.1 we will show how, using the particular cost function presented in section 2.4.1 the search space where the median has to be searched for can be drastically reduced. After that, in section 4.2, we will present a heuristic prediction function that will permit to avoid the evaluation of some states in the new search space. These two improvements will lead us to present a new algorithm for the median graph computation in section 4.3.

4.1 Reduction of the Search Space

In the Multimatch approach the search space for the median graph computation is determined as follows: suppose that $S = \{g_1, g_2, \ldots, g_n\}$ and let $k = \sum_{i=1}^{n} |g_i|$ be the sum of the sizes of all the graphs in S. Then, the algorithm explores all possible graphs having size in between 0 and k as possible candidate medians. For each number of nodes, all possible combinations of labels for these nodes and all possible combinations of edges (including the *null* edge) linking these nodes have to be tested. Thus, a huge number of combinations must be taken into account, and it makes this approach unfeasible already for a small number of graphs.

Two observations regarding this approach can be done that will permit to reduce the size of the search space. The former is that in [20] the union graph g_u, the graph containing all the nodes and all the edges of the graphs in S, is used as the starting point of the search space. This definition allows to explore all the possible combinations between nodes and edges but it also includes the repetition of some states in the search space, corresponding to combinations that can be found in more than one graph of the set. In order to avoid the generation of these repeated states we propose the use of the minimum common supergraph of S, g_{m_S}, instead of the union graph g_u. The use of g_{m_S} allows us to explore all the possible combinations (all the possible graphs generated from the union graph can be generated using the minimum common supergraph), avoiding the repeated combinations shared by different graphs of the set.

The latter is that given a set of nodes of g_u, all the edges in the graphs of S (including the null edge) are taken into account as possible candidates to construct the intermediate median graph. However, this number of combinations can be reduced taking into account an important property of the cost function given in [7]. It states that there always exists an optimal edit path between two given graphs that implies neither non-identical node substitutions nor non-identical edge substitutions.

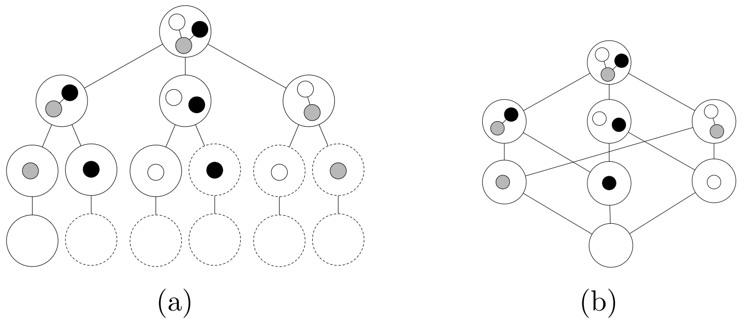
This result can be seen in the next example. Given the graphs g_1 and g_2 of Figure (2) one could suppose that the cheapest way to convert g_1 into g_2 is by deleting the edge linking the white and black nodes in g_1. Notice that because of the definition of graph given in Section 2.1 deleting an edge is equivalent to substitute such edge by an edge labelled as "null". But this operation is not allowed because it implies a non-identical edge operation. Then, a possible sequence of edit operations (Figure 2(d)) consists in deleting the white node from g_1 (including the deletion of its adjacent edges) and inserting the same node in g_2 together with their incident edges. This sequence of edit operations has a cost equal to 2, one node deletion and one node insertion. This result is consistent with the result obtained applying Equation (1), since $|g_1| = 4$, $|g_2| = 4$ and $|MCS(g_1, g_2)| = 3$.

Considering this property of the cost function, given a subset of nodes in g_u, only the labels of the edges connecting these nodes in g_u have to be considered when searching for the median graph. Any other label would lead to non-identical edge substitutions in some of the graphs of the set and therefore, to a non-optimal edit path that would never be applied. Thus, only the induced subgraphs of the g_u are valid options in the search space.

Conclusion: Merging both observations the new search space is composed only by the induced subgraphs of the minimum common supergraph of S, g_{m_S}.

4.1.1 Generation of the New Search Space

In this new scenario, the next approach can be followed. We take the minimum common supergraph of S, $g_{m_S} = mcs(S)$ with $|g_{m_S}| = p$ as the first candidate median graph. From this initial candidate median graph, new candidates can be generated by simply removing nodes and their adjacent edges from this first combination. A naive approach to explore all these possible candidate median graphs with a size between 0 and p may be carried out by a simple backtracking based algorithm. The root node corresponds to g_{m_S}. The tree is expanded in a depth-first approach exploring all the combinations produced by removing nodes from the root. The edges are implicitly deleted when a node is removed from a graph. Unfortunately, this straightforward approach still generates duplicate combinations of candidate medians, because different branches may lead to the same combinations of nodes, as can be seen in Figure 3(a), where the g_{m_S} is located in the root node. Notice that some of the combinations with 1 and 0 nodes are repeated in the tree. They are shown in the figure by means of dashed circles.



(a) (b)

Fig. 3. Search space including repeated combinations (a) and without these repeated combinations (b).

Then, a different approach can be followed where only non-repeated combinations are generated (Figure 3(b)). The number of different possible candidate medians can be easily computed. For a given i, with $0 \leq i \leq p$, the number of possible candidate median graphs with size i is $C_i^p = \frac{p!}{i!(p-i)!}$. That is, all the possible combinations of i nodes from a number of p nodes. As the search space is composed of all the induced subgraphs of g_m between the sizes 0 and p, the total number of possible different candidates corresponds to the sum $\sum_{i=1}^{n} C_i^p$, which is exactly 2^p. These 2^p different combinations can be generated using a breadth-first approach. Then, the search space may be thought as a rhombus (Figure 4). At the top of this rhombus (level 0) there is $g_{m_S} = mcs(S)$ with $|g_{m_S}| = p$. Below, there are all the induced subgraphs of g_{m_S} having *p-1* nodes (level 1). Concretely there are $C_{p-1}^p = p$ graphs. At the next level, each of these induced subgraphs will generate new induced subgraphs of g_{m_S} having *p-2* nodes. The number of combinations increase until the central row of the rhombus is reached (the maximum number of combinations). From this point to the bottom the number of combinations decreases until the last row (level p) is reached. At this point there is $C_0^p = 1$ combinations, corresponding to the empty graph g_e. This will be the new search space used to find the median graph.

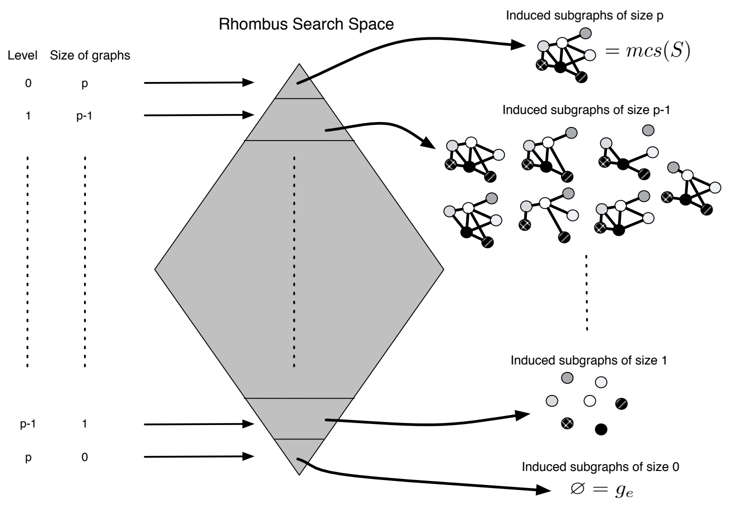

Rhombus Search Space
Level
Size of graphs
0
p
1
p-1
p-1
1
p
0
Induced subgraphs of size p
$= mcs(S)$
Induced subgraphs of size p-1
Induced subgraphs of size 1
Induced subgraphs of size 0
$\varnothing = g_e$

Fig. 4. Detail of the rhombus search space

4.2 Prediction of the cost function

This search space may still have a huge number of states. It is therefore desirable to avoid the evaluation of as many states as possible. To this end, we will show in this section that, from a given candidate median in the search tree, we can introduce a prediction of the cost associated to the candidate medians generated from the current node at the next level of the search space. Thus, evaluating this prediction function we will be able to decide whether it is worth to explore these candidate medians or not.

Let us start by defining the evaluation function, f^*, of a candidate median, $\bar{g}^*$. This function is just the sum of distances of the candidate median to all the graphs in the set, g_i. Then, taking the definition of the graph edit distance (equation (1)), we can express $f^*(\bar{g}^*)$, in the following way:

$$\begin{aligned} f^*(\bar{g}^*) &= SOD\,(\bar{g}^*) = \sum_{i=1}^{n} d(g_i, \bar{g}^*) = \sum_{i=1}^{n} (|g_i| + |\bar{g}^*| - 2|MCS(g_i, \bar{g}^*)|) \\ &= n|\bar{g}^*| + \sum_{i=1}^{n} |g_i| - 2\sum_{i=1}^{n} |MCS(g_i, \bar{g}^*)| \end{aligned}$$

The complexity of this expression depends on the cost of computing the MCS, which, in the general case, is exponential in the size of the involved graphs. It is therefore desirable to avoid as many evaluations of this function as possible. To this end, we can try to infer the value of this function as we traverse the search space, without having to compute the MCS between the candidate median and all the graphs. Let us take two candidate median graphs, $\bar{g}_1$ and $\bar{g}_2$. As we will be traversing the search space by generating all the induced subgraphs of g_{m_S}, let us suppose too, without loss of generality, that $\bar{g}_2$ is an induced subgraph of $\bar{g}_1$. Now, using this assumption we will try to find some relation between $f^*(\bar{g}_1)$ and $f^*(\bar{g}_2)$. Let us denote by $h^*(\bar{g}_1, \bar{g}_2)$ the function that computes the difference between both evaluation functions,

$$\begin{aligned} h^*(\bar{g}_1, \bar{g}_2) &= f^*(\bar{g}_2) - f^*(\bar{g}_1) = \\ &= n\,(|\bar{g}_2| - |\bar{g}_1|) - 2\sum_{i=1}^{n} |MCS(g_i, \bar{g}_2)| + 2\sum_{i=1}^{n} |MCS(g_i, \bar{g}_1)| \end{aligned} \quad (3)$$

We will call h^* the *prediction* function as we can express the evaluation function of any graph $\bar{g}_2$ in the search space in terms of the evaluation function of any other previous graph $\bar{g}_1$ and the function $h^*(\bar{g}_1, \bar{g}_2)$.

Now, let us analyze this prediction function. If we are traversing the search space from node $\bar{g}_1$ to node $\bar{g}_2$ and we have already evaluated $f^*(\bar{g}_1)$, we know the value of all the terms in $h^*(\bar{g}_1, \bar{g}_2)$ except for $|MCS(g_i, \bar{g}_2)|$. However, we can infer an upper limit for this term and therefore, we can define an estimation of this prediction function. We will denote this estimation by $h(\bar{g}_1, \bar{g}_2)$.

Let us observe that the MCS between two graphs will never have more nodes than any of them. In addition, we have assumed that we are traversing the search space from $\bar{g}_1$ to $\bar{g}_2$ and, therefore, $\bar{g}_2$ is a subgraph of $\bar{g}_1$. Thus, we can easily conclude that $|MCS(g_i, \bar{g}_2)| \le |MCS(g_i, \bar{g}_1)|$. Therefore, the following condition holds:

$$|MCS(g_i, \bar{g}_2)| \leq \min(|g_i|, |\bar{g}_2|, |MCS(g_i, \bar{g}_1)|) \tag{4}$$

Then, combining equations (3) and (4) we can obtain the following estimation of the prediction function:

$$\begin{aligned} h^*(\bar{g}_1, \bar{g}_2) \geq h(\bar{g}_1, \bar{g}_2) = \\ = n(|\bar{g}_2| - |\bar{g}_1|) + 2\sum_{i=1}^{n} |MCS(g_i, \bar{g}_1)| - 2\sum_{i=1}^{n} \min(|g_i|, |\bar{g}_2|, |MCS(g_i, \bar{g}_1)|) \end{aligned} \tag{5}$$

Finally, $h(\bar{g}_1, \bar{g}_2)$ can be used to obtain an estimation of the evaluation function of node $\bar{g}_2$, $f(\bar{g}_2)$:

$$f(\bar{g}_2) = f^*(\bar{g}_1) + h(\bar{g}_1, \bar{g}_2) \tag{6}$$

Using this estimation we can reduce the computation time of the algorithm described in section 4.3 by reducing the number of candidate graphs whose evaluation function needs to be explicitly computed. Given candidate a median graph, $\bar{g}^*$, we can compute this estimation for all the graphs in the search space that can be reached from this initial graph. All nodes $\bar{g}_j$ whose estimation $f(\bar{g}_j)$ is greater than the actual evaluation function of the current median graph, $\bar{g}^*$ can automatically be discarded and do not have to be evaluated.

One difficulty to apply this strategy is that, given a graph in the search space, we cannot guarantee that the prediction function is either positive or negative for all its induced subgraphs. Therefore, as we are generating the search space level by level, discarding one state of the search space using the prediction function does not permit to discard all its remaining induced subgraphs too. Then, in the algorithm that we will present in the next section, we will use an strategy consisting of using the prediction function to discard states only at the next level of the search space.

4.3 New Exact Algorithm

Keeping in mind the reductions of the search space proposed so far (including the heuristic function), and assuming that the minimum common supergraph of S, g_{m_S} is computed, we are able to present a new and more efficient exact algorithm to compute the generalized median graph. For the sake of completeness the algorithm is presented as **Algorithm 1**.

The algorithm receives a set S of graphs as input and returns a complete list of true median graphs, all of them with the same SOD. The first task is to compute the g_{m_S}. This is carried out using a similar approach to that presented in [6]. Once g_{m_S} is computed, the algorithm computes the term $SOD(g_{m_S}) = \sum d(g_i, g_{m_S})$ using the function $ComputeSOD(g_{m_S}, S)$. The distance between two graphs $d(g_1, g_2)$ is computed using the Equation (1). The pair $(g_{m_S}, SOD(g_{m_S}))$ is stored in the candidate medians list (line 3). After that, all the possible induced subgraphs of g_{m_S} with size $|g_{m_S}| - 1$ are generated using the function $Expand(g_m)$ (line 5), and stored in a table. Given a graph g, the function $Expand(g)$ generates all its induced subgraphs with size $|g| - 1$, by taking all the possible different combinations of its nodes and for each combination building the induced subgraphs of g with these nodes. Once all the induced subgraphs of g_{m_S} are generated, they are marked as valid or not by using the heuristic function explained in section 4.2. Only for valid graphs, its SOD is computed and compared with the actual best SOD. If it is better, this graph becomes the new current median graph. However, for all the graphs at the current level (either valid or not), all its induced subgraphs are generated using again the function *Expand* in order to make them available in the next iteration. This is due to the fact explained at the of the previous section that we can only discard states at the next level of the search space. In addition, since different graphs may generate the same induced subgraph (the search space is a rhombus, not a tree), we keep a hash table at each level of the search space that permits to know if an induced subgraph has already been generated and if it is valid or not. The process is repeated until all the levels of the rhombus search space have been visited.

5 Experimental Setup

In this section we will provide the results of an experimental evaluation of the proposed method for the generalized median graph computation in order to validate the improvements we have introduced with our new algorithm. We have separated such experimental setup in two different parts. Firstly, in section 5.1 we will compare our new algorithm with the multimatch approach using a dataset of synthetical data. After that, in section 5.2 we will extend the applicability of our algorithm to a real database and we will compare it against the genetic algorithm presented in [16]. It is important to notice that all these experiments were run on an Apple iMac computer equipped with a 2.16GHz Intel Core 2 Duo processor and 2Gb of main memory.

Algorithm 1 Exact-median Algorithm

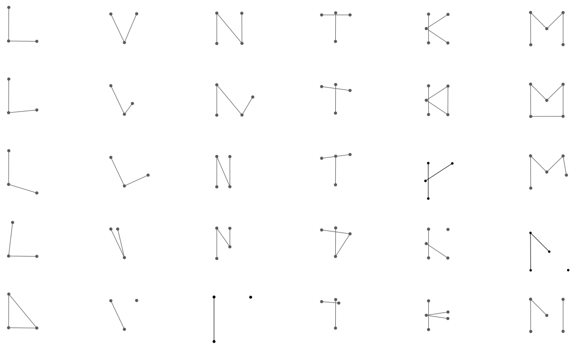
```
Require: A set of graphs $S = \{g_1, g_2, \ldots, g_n\}$
Ensure: A list L of true median graphs
 1: $g_{m_S} = Compute\_mcs(S)$
 2: $ComputeSOD(g_{m_S}, S)$
 3: Insert pair $(g_{m_S}, SOD(g_{m_S}))$ in $L$
 4: Let $SOD(g_{m_S})$ be the minimum SOD
 5: Expand($g_{m_S}$)
 6: for size = $|g_{m_S}| - 1$ to 0 do
 7:   while $RemainInducedSubGraph(size)$ do
 8:     $g = GetNextInducedSubgraph()$
 9:     Expand(g)
10:     if IsValid(g) then
11:       $ComputeSOD(g, S)$
12:       if $SOD(g)$ is less than the minimum SOD then
13:         Let $SOD(g)$ be the minimum SOD
14:         Delete L
15:         Insert pair $(g, SOD(g))$ in $L$
16:       else
17:         if $SOD(g)$ is equal to the minimum SOD then
18:           Insert pair $(g, SOD(g))$ in $L$
19:         end if
20:       end if
21:     end if
22:   end while
23: end for
24: Return L
```

5.1 Application to Synthetic Data

In this experiments, the graph dataset was composed of graphs that represent capital letters. It was originally designed manually at the University of Bern. From the original set composed by 15 letters, we only used a subset of 6 letters L, V, N, T, K and M. Then, from each original model, we manually generated 4 distorted instances. Therefore, our dataset is composed of 30 elements and 6 classes. Each class represents a letter and contains 5 different instances of each letter. Table 2 shows the original letters in the first row and the four distorted letters in the rest of the rows. Such distortions include moving or deleting nodes and edge deletions. Thus, in the same class graphs with different cardinality may appear. The letters are represented by graphs as follows. The straight lines are represented by edges and the terminal points of the lines by the nodes. Nodes are labelled by a two-dimensional attribute that represents the position (x,y) of the terminal point. Edges have a one-dimensional and binary attribute that represents existence or non-existence.

The experiments consisted of the computation of the generalized median graphs using different number of graphs in the training set. For each median, the elapsed time and the number of SOD computations needed to compute it were recorded.

Table 2
Entire database used in the experiments.



5.1.1 Experimental Results

To compare the computation time and the number of SOD computations of our new algorithm with respect to the multimatch algorithm, we defined 24 sets of graphs. For every class, we formed 4 different sets composed of 2, 3, 4 and 5 graphs, respectively. Table 3 shows, for each letter (first column), the number of nodes of the graph that represents it (in brackets) and, for each of the 4 sets, the maximum sum of nodes of the set.

Table 3
Sum of nodes in the set S.

	N°graphs in S			
Letters ($Nodes/graph$)	2	3	4	5
L,V (3)	6 ●	9 ●	12 ●	15
N,T (4)	8 ●	12 ●	16	19
K,M (5)	10 ●	15	20	24

● *Combinations used in the Multimatch algorithm*

Notice that, because of computation time, not all the possible combinations could be used to compute the generalized median graph using the multimatch algorithm. The used combinations are marked with the symbol ● in table 3. In contrast, all the possible combinations could be applied to our new exact algorithm.

In figure 5 we can see some of the median graphs obtained with our algorithm. The figure shows the median graphs obtained using 5 graphs for every letter.

A first important remark is that in four out of the six letters (figure 5(a)-5(d)) the median graph is also the maximum common subgraph of all the graphs in the set. Although it is only an empirical result, this fact permits to think of a possible relationship between the maximum common subgraph and the median graph, as it has already been established in [8] for two graphs, but extended to a set of an arbitrary number of graphs.

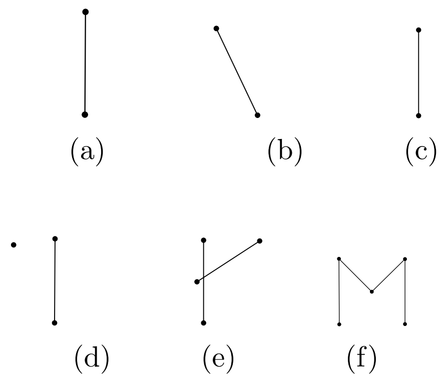


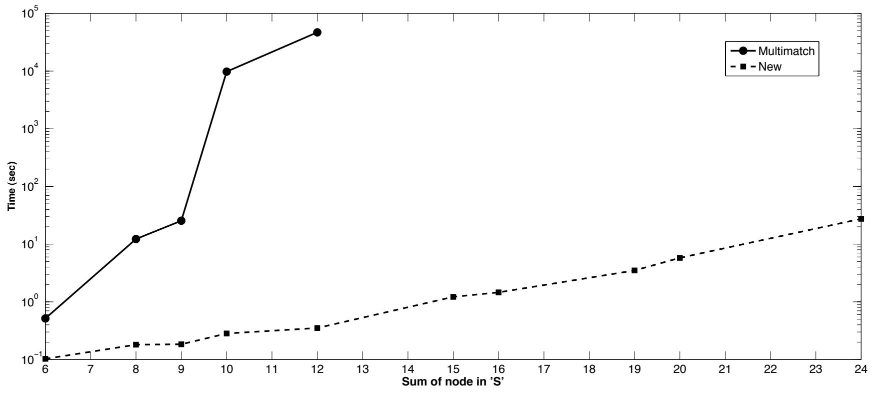
Fig. 5. Example of some computed medians.

5.1.2 Computation time and Number of SOD computations

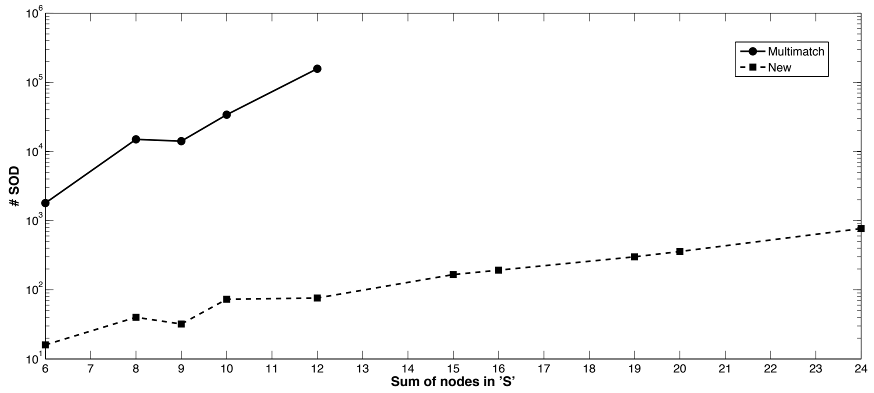
The results of the computation time (including the *mcs* computation) and the SOD computations required to calculate the median graph as a function of the total number of nodes in S are shown in Figures 6(a) and 6(b) respectively. Notice that the results are the mean values obtained for all sets S having the same number of nodes.

The first important result is that due to the combinatorial explosion of the multimatch algorithm it could be only applied to sets of graphs whose sum of nodes is up to 12. Beyond this limit, the time required for this algorithm is unfeasible. This result is consistent with the results presented in [16]. In contrast, our algorithm could be applied obtaining reasonable computation times to sets having up to 24 nodes. In addition, the computation time required by our algorithm is quite lower than the time required by the multimatch algorithm, even for small sum of nodes in S. Such difference in time is more evident when the sum of nodes in S becomes larger. All this facts can be appreciated in the results showed in Figure 6(a).

A similar behavior can be appreciated in the number of SOD computations needed for the two algorithms (Figure 6(b)). Again, a significant difference between the number of SOD computations required by the multimatch algorithm and our algorithm can be observed in the results. This reduction in the number of SOD computations can be associated both to the reduction of the search space and the heuristic function presented in section 4.2.



(a)



(b)

Fig. 6. Computation time (a) and number of SOD computations (b) as a function of the total number of nodes of the graphs in S.

5.1.3 Reduction in Time and SOD computations

In order to be able to quantify the gain achieved by our algorithm with respect to the multimatch algorithm, we compare in Table (4) the time and number of SOD computations required by both algorithms to compute the generalized median graphs. The values of the computation time of the multimacth algorithm, and our new exact algorithm are shown in columns 2 and 3 respectively. The next column (labelled as %Reduction) represents the reduction of time (in percentage) of our algorithm, taking the value of the multimatch algorithm as reference. The same reasoning with respect to the number of SOD computations can be done for the rest of the table.

We can observe that a significant reduction in the time needed for the median graph computation is achieved by our new algorithm with respect to the multimatch algorithm. This reduction is increased as the sum of nodes in the set also increases. Notice that the mean time percentage needed for our algorithm with respect to the multimatch algorithm to compute the median graph is

Table 4
Percentage of time and SOD computations required for the new exact algorihtms respect to the multimatch algorithm.

	Time (sec)			**SOD computations**		
$\sum \lvert g_i \rvert$	Multimatch	New	%Reduction	Multimatch	New	%Reduction
6	0.51	0.103	79.81	1794	16	99.11
8	12.25	0.181	98.53	2943	32	98.92
9	25.40	0.184	99.28	14092	40	99.72
10	9712	0.283	99.98	33961	72	99.79
12	46794.85	0.352	99.99	157176	76	99.95

about 4.5% (or a 95.5% of reduction). A similar reasoning can be done for the number of SOD computations. In this case the mean percentage of the required SOD computations is about 0.5% (99.5% of reduction).

5.2 Application to Real Data

In the previous section, it has been shown that the computation of the median graph by means of our new algorithm is not restricted to a very small number of graphs with a small number of nodes as in the case of the multimatch algorithm. In this section we will go a step beyond and we will extend the exact computation of the median graph to real data. In particular we will use our algorithm to obtain prototypes of certain molecules. We will compare the results with those obtained by our own implementation of the genetic algorithm presented in [16].

The molecule database consists of graphs representing molecular compounds. These graphs are extracted from the AIDS Antiviral Screen Database of Active Compounds [1]. Such database consists of two different classes of molecules: active and inactive, depending on whether they show activity against HIV or not respectively. The molecules are converted into graphs in a straightforward way, representing the atoms as nodes and the covalent bonds as edges. The nodes are labelled with the number of the corresponding chemical symbol. The edges are labelled with the valence of the linkage. Some examples of each class are shown in figure (7). In order to simplify the representation, in such examples, different chemical symbols are represented using a different gray tonality.

For each class we have 50 different instances or molecules. The experiments consisted in generating, for each class, 40 sets having a different number of graphs. The graphs in each set had different sizes and were chosen randomly

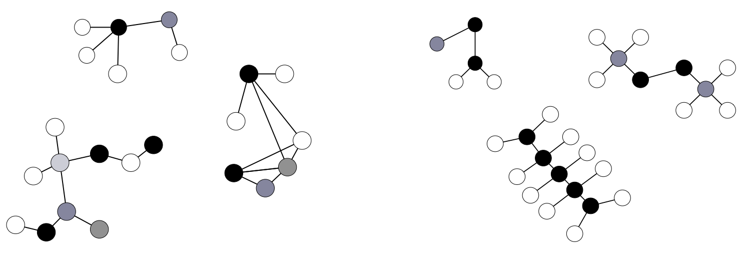

(a) (b)

Fig. 7. Active compounds (a) and Inactive compounds (b)

from the original set of 50 instances. In this case, for each set the time required and the SOD of the median graph obtained by each algorithm were recorded. It is important to notice that the sum of nodes in the sets ranged from 4 to 23 nodes. The results are shown in the next sections.

5.2.1 Computation Time

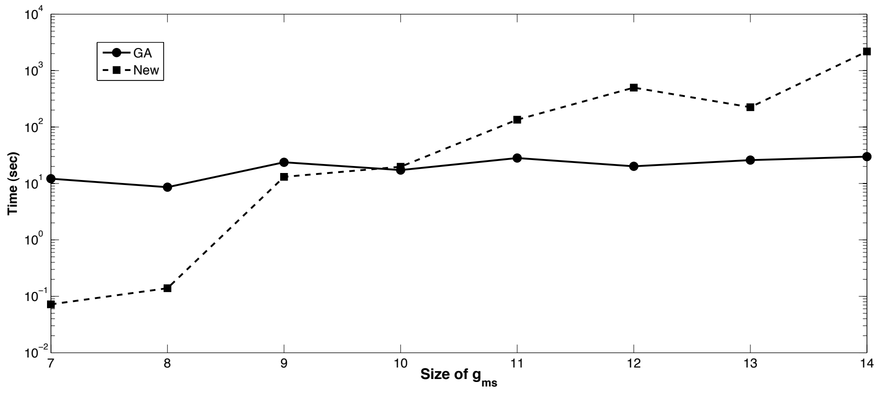
The required computation time for both active and inactive compounds are shown in figures 8(a) and 8(b) respectively. In both charts, the *x-axis* represent the size of the minimum common supergraph of the set, independently of the sum of nodes in the set.

First of all it is important to notice that for minimum common supergraphs of sizes up to 10-11, the time required by our algorithm is lower than the time required by the genetic algorithm. Such difference is specially relevant for small sizes of the minimum common supergraph (up to 8), and also for the case of inactive molecules, where the time required by our exact algorithm is two orders of magnitude lower than the genetic algorithm. It is important to mention that, for sizes of the minimum common supergrah up to 11 nodes, we can find sets whose sum of nodes was 20 nodes. This result suggests that for sets of graphs sharing large structures (and consequently the minimum common supergraph tends to the mean size of the graphs in the set), our algorithm may outperform the genetic algorithm.

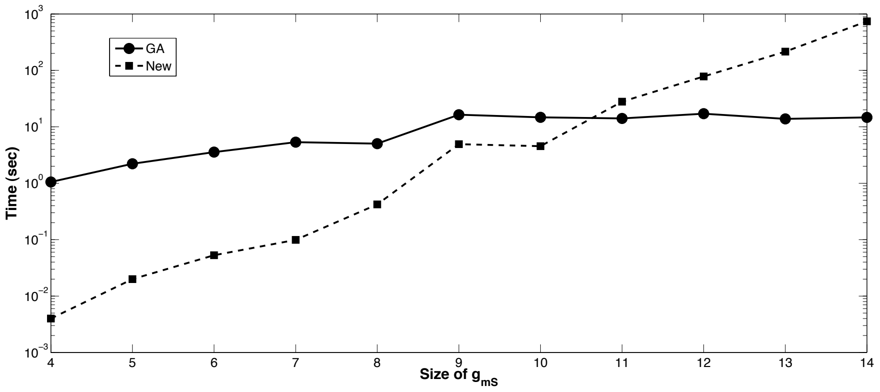
As our algorithm is an optimal one, the computation time for larger sizes of the minimum common supergraph increases rapidly. Nevertheless, the computation time needed in these cases is not unfeasible (the sum of nodes of the sets with minimum common supergraph of 14 nodes was 23 nodes).

5.2.2 SOD

The definition of the median graph implies the computation of the sum of distances of the candidate median to all the graphs in the set. In this sense, a measure of how good the median graph is, can be obtained by computing the



(a)



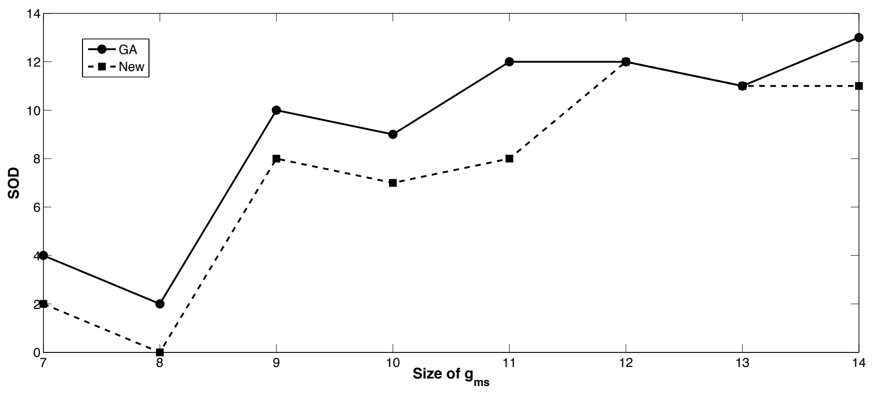
(b)

Fig. 8. Computation time for Active (a) and Inactive (b) compounds as a function of the size of mcs(S).

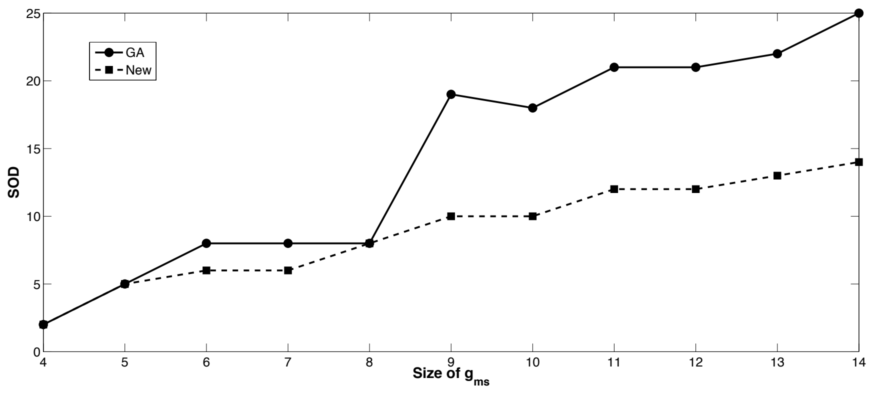
term SOD for the computed medians. A measure of the term SOD achieved by the medians computed by both algorithms is shown in figures 9(a) and 9(b), for the active and inactive compounds respectively. Again, the *x-axis* represents the size of the minimum common supergraph of the set, independently of the sum of nodes in the set.

As expected, the term SOD for the genetic algorithm is always greater than or equal to the term SOD achieved by the exact algorithm. This behavior is more clear in the case of inactive compounds, where for sizes of the minimum common supergraph greater than 9, the difference in the term SOD increases significantly (figure 9(b)). The difference is less evident in the case of active compounds.

Such difference in the term SOD suggests that the medians obtained by the genetic algorithm tend to diverge as the size of the minimum common supergraph increases (the graphs in the set are more dissimilar). This may lead to obtain median graphs that do not represent accurately the set of graphs. In



(a)



(b)

Fig. 9. SOD of computed median for Active (a) and Inactive (b) compounds as a function of the size of mcs(S).

contrast, the exact algorithm always finds an optimal median graph. Thus it always finds the best representative in the set (following the criteria of the definition of the median graph).

6 Conclusions

Median graphs have been presented as a good alternative to compute the representative of a set of graphs. Unfortunately, the exact computation of the median graph has an exponential complexity. Only one exact algorithm has been presented up. Although exact solutions are in most cases not feasible in real applications, it is still important to be able to obtain them as they will always be a better representation of a set. In addition, finding exact solutions may help us to understand the nature of this concept and can be used to obtain better approximate solutions too.

The main contributions of this paper are twofold. Firstly and from a theoretical point of view, we have shown that using a particular cost function and a distance measure based on the maximum common subgraph, the size of the search space where the median graph has to be computed can be drastically reduced. In addition, an heuristic strategy has also been introduced in order to improve the basic version of this new algorithm avoiding the evaluation of some states in the search space. Secondly, and based on this theoretical result a new exact algorithm for the computation of the generalized median graph has been presented.

We have compared this new exact algorithm to the previous existing exact algorithm using synthetic data. The results show that our algorithm clearly outperforms the previous existing algorithm. Encouraged by these results, we have applied our algorithm to the computation of median graphs using real data and we have compared the results with those obtained by an approximate algorithm based on the genetic search. The results show that, although the application of the exact algorithm is still limited, it can be used in real problems for the first time.

There are still a number of issues to be investigated in the future. These results open a door towards the application of the new theoretical properties and the new algorithm to obtain more accurate approximate solutions of the median graph and also to increase the efficiency of the existing approximate algorithms. In this sense, we have already presented in [13,14] two theoretical results related to some bounds on the size and the maximum SOD of the median graph, that may help to obtain better approximate algorithms and also to extend the applicability of the median graph. In addition, the possible relation between the maximum common subgraph and the median graph suggested in section 5 can also be further investigated in order to see whether the algorithms for the computation of the MCS could be used to obtain exact or approximate solutions of the median graph.

Acknowledgements

This work has been supported by by the research Fellowship number 401-027 (UAB), the Cicyt project TIN2006-15694-C02-02 (Ministerio Ciencia y Tecnología) and the Spanish research programme Consolider Ingenio 2010: MIPRCV (CSD2007-00018). We would like to thank K. Riesen from the University of Bern for his help with the databases.

References

[1] Development Therapeutics Program DTP. AIDS Antiviral Screen. *http : //dtp.nci.nih.gov/docs/aids/aids_data.html* (2004).

[2] E. Balas, C. S. Yu, Finding a maximum clique in an arbitrary graph, SIAM J. Comput. 15 (4) (1986) 1054–1068.

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

[4] H. Bunke, G. Allerman, Inexact graph matching for structural pattern recognition, Pattern Recognition Letters 1 (4) (1983) 245–253.

[5] H. Bunke, P. Foggia, C. Guidobaldi, C. Sansone, M. Vento, A comparison of algorithms for maximum common subgraph on randomly connected graphs, in: Structural, Syntactic, and Statistical Pattern Recognition, Joint IAPR International Workshops SSPR 2002 and SPR 2002, Windsor, Ontario, Canada, August 6-9, 2002, Proceedings. Lecture Notes in Computer Science Vol. 2396, 2002.

[6] H. Bunke, P. Foggia, C. Guidobaldi, M. Vento, Graph clustering using the weighted minimum common supergraph, in: Graph Based Representations in Pattern Recognition, 4th IAPR International Workshop, GbRPR 2003, York, UK, June 30 - July 2, 2003, Proceedings. Lecture Notes in Computer Science Vol. 2726, 2003.

[7] H. Bunke, X. Jiang, A. Kandel, On the minimum common supergraph of two graphs, Computing 65 (1) (2000) 13–25.

[8] H. Bunke, A. Kandel, Mean and maximum common subgraph of two graphs, Pattern Recognition Letters 21 (2) (2000) 163–168.

[9] H. Bunke, A. Münger, X. Jiang, Combinatorial search versus genetic algorithms: A case study based on the generalized median graph problem, Pattern Recognition Letters 20 (11-13) (1999) 1271–1277.

[10] D. Conte, P. Foggia, M. Vento, Challenging complexity of maximum common subgraph detection algorithms: A performance analysis of three algorithms on a wide database of graphs, Journal of Graph Algorithms and Applications 11 (1) (2007) 99–143.

[11] P. J. Durand, R. Pasari, J. W. Baker, C. che Tsai, An efficient algorithm for similarity analysis of molecules, Internet Journal of Chemistry 2 (17).

[12] M. Ferrer, F. Serratosa, A. Sanfeliu, Synthesis of median spectral graph, in: J. S. Marques, N. P. de la Blanca, P. Pina (eds.), IbPRIA (2), vol. 3523 of Lecture Notes in Computer Science, Springer, 2005.

[13] M. Ferrer, F. Serratosa, E. Valveny, On the relation between the median and the maximum common subgraph of a set of graphs, in: F. Escolano, M. Vento

(eds.), Graph-Based Representations in Pattern Recognition, 6th IAPR-TC-15 International Workshop, GbRPR 2007, Alicante, Spain, June 11-13, 2007, Proceedings, vol. 4538 of Lecture Notes in Computer Science, Springer, 2007.

[14] M. Ferrer, E. Valveny, F. Serratosa, Bounding the size of the median graph, in: J. Martí, J.-M. Benedí, A. M. Mendonça, J. Serrat (eds.), IbPRIA (2), vol. 4478 of Lecture Notes in Computer Science, Springer, 2007.

[15] A. Hlaoui, S. Wang, Median graph computation for graph clustering, Soft Comput. 10 (1) (2006) 47–53.

[16] X. Jiang, A. Münger, H. Bunke, On median graphs: Properties, algorithms, and applications, IEEE Trans. Pattern Anal. Mach. Intell. 23 (10) (2001) 1144–1151.

[17] D. Justice, A. O. Hero, A binary linear programming formulation of the graph edit distance, IEEE Trans. Pattern Anal. Mach. Intell. 28 (8) (2006) 1200–1214.

[18] S. W. Lu, Y. Ren, C. Y. Suen, Hierarchical attributed graph representation and recognition of handwritten chinese characters, Pattern Recognition 24 (7) (1991) 617–632.

[19] J. J. McGregor, Backtrack search algorithms and the maximal common subgraph problem, Software - Practice and Experience 12 (1) (1982) 23–24.

[20] A. Münger, Synthesis of prototype graphs from sample graphs, in: Diploma Thesis, University of Bern (in German), 1998.

[21] M. Neuhaus, K. Riesen, H. Bunke, Fast suboptimal algorithms for the computation of graph edit distance, in: Structural, Syntactic, and Statistical Pattern Recognition, Joint IAPR International Workshops, SSPR 2006 and SPR 2006, Hong Kong, China, August 17-19, 2006, Proceedings. Lecture Notes in Computer Science 4109, 2006.

[22] A. R. Pearce, T. M. Caelli, W. F. Bischof, Rulegraphs for graph matching in pattern recognition, Pattern Recognition 27 (9) (1994) 1231–1247.

[23] A. Robles-Kelly, E. R. Hancock, Graph edit distance from spectral seriation, IEEE Trans. Pattern Anal. Mach. Intell. 27 (3) (2005) 365–378.

[24] A. Sanfeliu, K. Fu, A distance measure between attributed relational graphs for pattern recognition, IEEE Transactions on Systems, Man and Cybernetics 13 (3) (1983) 353–362.

[25] Y. Wang, C. Maple, A novel efficient algorithm for determining maximum common subgraphs, in: 9th International Conference on Information Visualisation, IV 2005, 6-8 July 2005, London, UK, IEEE Computer Society, 2005.

[26] D. White, R. C. Wilson, Mixing spectral representations of graphs, in: 18th International Conference on Pattern Recognition (ICPR 2006), 20-24 August 2006, Hong Kong, China, IEEE Computer Society, 2006.

[27] E. K. Wong, Model matching in robot vision by subgraph isomorphism, Pattern Recognition 25 (3) (1992) 287–303.