J Comb Optim (2009) 17: 21–44
DOI 10.1007/s10878-008-9184-7

Generalized median graphs and applications

Lopamudra Mukherjee · Vikas Singh · Jiming Peng · Jinhui Xu · Michael J. Zeitz · Ronald Berezney

Published online: 13 September 2008

Abstract We study the so-called Generalized Median graph problem where the task is to construct a prototype (i.e., a 'model') from an input set of graphs. While our primary motivation comes from an important biological imaging application, the prob-

This work was supported by NSF grants CCF-0546509, IIS-0713489, and NIH grant GM 072131-23. The second author was also supported in part by the Department of Biostatistics and Medical Informatics, UW-Madison and UW ICTR, funded through an NIH Clinical and Translational Science Award (CTSA), grant number 1 UL1 RR025011.

L. Mukherjee (✉)
Department of Mathematical and Computer Sciences, University of Wisconsin–Whitewater, Whitewater, USA
e-mail: mukherjl@uww.edu

V. Singh
Department of Biostatistics and Medical Informatics, University of Wisconsin–Madison, Madison, USA
e-mail: vsingh@biostat.wisc.edu

J. Peng
Industrial and Enterprise Systems Engineering, University of Illinois at Urbana-Champaign, Urbana, USA
e-mail: pengj@uiuc.edu

J. Xu
Department of Computer Sci. and Eng., The State University of New York at Buffalo, Buffalo, NY, USA
e-mail: jinhui@cse.buffalo.edu

M.J. Zeitz · R. Berezney
Department of Biological Sciences, The State University of New York at Buffalo, Buffalo, NY, USA

M.J. Zeitz
e-mail: berezney@buffalo.edu

R. Berezney
e-mail: mjzeitz@buffalo.edu

lem effectively captures many vision (e.g., object recognition) and learning problems, where graphs are increasingly being adopted as a powerful representation tool. Existing techniques for his problem are evolutionary search based; in this paper, we propose a polynomial time algorithm based on a linear programming formulation. We propose an additional algorithm based on a bi-level method to obtain solutions arbitrarily close to the optimal in (worst case) non-polynomial time. Within this new framework, one can optimize edit distance functions that capture similarity by considering vertex labels as well as he graph structure simultaneously. We first discuss experimental evaluations in context of molecular image analysis problems—he methods will provide the basis for building a topological map of all 23 pairs of the human chromosome. Later, we include (a) applications to other biomedical problems and (b) evaluations on a public pattern recognition graph database.

Keywords Median graph · Graph edit distance · Graph matching · Prototype building · Chromosome organization · Cell-nucleus imaging

1 Introduction

Graphs are ubiquitous in many application areas where information regarding object-to-object relationships needs to be encoded. They serve as an invariant structure descriptor in object recognition and for low level image representations in vision. Complicated patterns of inter-related transactions and relationships between seemingly unrelated entities is extracted using these representations in data analysis. Recent literature in many research areas such as vision, data mining, and artificial intelligence includes several examples of graph theoretic algorithms for some classical problems like segmentation (Shi and Malik 2000), image denoising (Boykov et al. 2001), spatial data-mining (Ng and Han 1994), clustering (Brandes et al. 2003), shape matching (Siddiqi et al. 1998) and stereo (Horaud and Skordas 1989) using ideas such as spectral methods, graph cuts, bipartite matching and minimum spanning trees. In many applications, results from classical graph theory can be directly adapted to derive efficient solutions; an appropriate graph construction that encodes available information presents the main challenge here. In other cases, the construction strategy is somewhat simpler. However, issues such as distortion and noise affect the graph representations. For example, we may have multiple representations of the same object (or scene) based on the number of observations (or readings taken). We are then faced with the task of building *a single* composite model describing the scene in the best possible way. Problems of this form are broadly known as *Prototype Learning*, and require learning a model of a class given several of its members. The Generalized Median Graph problem asks following question: given a set of graphs, what is a *median* graph (not necessarily from the input set) that is a good representative (or prototype) for the set?

We are particularly interested in this problem in the context of applications in biological image analysis—specifically, in building a topological map of chromosome organization in the human cell nucleus. A proper understanding of the chromosomal organization will lead to insights into developmental changes and gene regulation and interaction; an all important focus of ongoing research is on its relationship to the organization and mutations in the genome (Cremer and Cremer 2001;

Parada et al. 2002; Nagele et al. 1998b). Given sets of nuclear images exhibiting chromosomal organization, the task is to derive a "model" that is a good representation of the organizational relationship among chromosomes. The objective essentially reduces to finding a median graph (i.e., an exemplar) for a set of attributed graphs from individual nuclear images.

The paper is organized as follows. We start by formulating the problem in Sect. 1.1 and then review some of the previous works for this problem as well as a closely related problem of graph isomorphism (in Sect. 1.2). Next, we describe how the graph isomorphism cost model can be generalized for attributed graphs (in Sect. 2.1). Later, we describe how this cost model can be incorporated into our optimization model (in Sects. 2.2, 2.3) that yields the generalized median graph (in Sect. 4). Finally, in Sect. 5 we show extensive experimental results and application of median graphs to two interesting biomedical problems. An earlier version of this paper appeared in Mukherjee et al. (2007).

1.1 Problem description

The median graph problem is formally defined as follows. Let L_V and L_E denote the set of node and edge labels of the graph, respectively. A labeled undirected graph G is then a four tuple, $G = (V, E, f_v, f_e)$, where

- V (and E) denote the vertex (and edge) sets,
- $f_v : V \to L_V$ is a function assigning labels or weights to the nodes, and
- $f_e : E \to L_E$ is a function assigning labels or weights to the edges.

Let $\mathbb{G} = (G_1, G_2, \dots, G_n)$ be a collection of graphs (in arbitrary orientation) with the following properties:

- $G_i = (V_i, E_i, f_v, f_e)$, $V_i \subset V$ and $E_i \subset E$ $\forall i$.
- No restriction on the uniqueness of the vertex labels of V_i, i.e., $f_v(u_i) = f_v(v_i)$; $u_i, v_i \subset V_i$ is permissible.
- No restriction on the cardinality of the graphs in $\mathbb{G}$, i.e., $|G_i| \neq |G_j|$; $G_i, G_j \in G$ is permissible.

The median graph for $\mathbb{G} = \{G_1, \dots, G_n\}$ must minimize the sum of distances as follows.

$$\bar{G} = \arg\min_{\hat{G}} \sum_{i=1}^{n} d(\hat{G}, G_i),$$

where $d(\cdot,\cdot)$ is an appropriately defined 'distance' function. In the simplest case, $d(\cdot,\cdot)$ can be the cost of the fewest edit operations required to 'convert' one graph to the other. Alternative definitions of distance may reflect a *similarity measure* between a pair of graphs, as we will see shortly. When $\bar{G} \in \mathbb{G}$, the median graph is the set median; if we waive this requirement, we get the generalized median graph problem—the topic of this paper.

Notation Throughout this paper, bold upper case letters ($\mathbf{A}$) and bold upper case letters with a single subscript ($\mathbf{A_0}$) will denote matrices; upper case letters with two subscripts or two superscripts will refer to individual matrix entries (A_{ij}, A_0^{ij}, and $A(i,j)$).

1.2 Previous works

The idea of median graphs was recently introduced by Jiang et al. (2001). They proposed an approach for generating the median graph from a set of graphs by simultaneously finding (1) the generalized median graph and (2) the optimal mapping between the to-be-computed median and each graph in the input set. The actual optimization, however, was genetic search based. A subsequent paper by Hlaoui and Wang (2006) focused on building the median graph through a two step process. First, they found a set of likely nodes for the median. This was done using a clustering algorithm on the node sets of all input graphs. The input graphs were matched (by node edits alone) to the node set of the median at step i, and this process was iterated to determine a good choice for the median. In the second stage, they assigned labels to edges based on the matching results from the first stage. Their method relied on certain hypotheses where local choices that improved the objective function were picked greedily. To summarize, both existing approaches are soft-computing based and no combinatorial solution methods are available. We note that for the special case when the graphs are certain types of trees, some nice results are known (Phillips and Warnow 1996).

Observe that to be able to build a median for a set of graphs, it is necessary to be able to quantitatively evaluate the similarities between a subset of graphs. When we consider only two graphs, we have the graph matching or graph isomorphism problem.[1] Unlike generalized median graphs which is still relatively new, the graph isomorphism problem, has been extensively studied by the optimization, mathematics, and computer science communities. The earliest papers on this topic are due to Corneil and Gotlieb (1970) and Ullmann (1976). The status of the problem is interesting—no polynomial time algorithms are known, at the same time, it is also not known to be **NP**-hard. However, a number of approaches exist for solving various special cases of this problem, for example see Boblaender (1990). We will avoid an exhaustive discussion (we refer to Toda 1999 for details) but discuss only on a subset of algorithms that have used mathematical programming frameworks to do graph matching.

In Umeyama (1988), Umeyama proposed an algorithm for graph matching based on eigen decomposition of the adjacency matrix of a graph. The technique is efficient in practice but is only applicable for adjacency matrices with no repeat eigen values (graphs with low connectivity will have multiple zero eigen values Justice and Hero 2006). The linear programming (LP, for short) formulation by Almohamad and Duffuaa (1993) uses adjacency matrices for weighted graph matching. This approach nicely exploits the relationship between permutation matrices[2] and graph isomor-

[1]A graph isomorphism from a graph G to a graph G' is a bijective mapping from the nodes of G to the nodes of G' that preserves all labels and the structure of the edges. A subgraph isomorphism applies from subgraphs of G to subgraphs of G'.

[2]A permutation matrix is obtained by permuting the rows of an $n \times n$ identity matrix according to a permutation of the numbers 1 to n. Each row/column has precisely single 1 and every permutation maps to a unique permutation matrix.

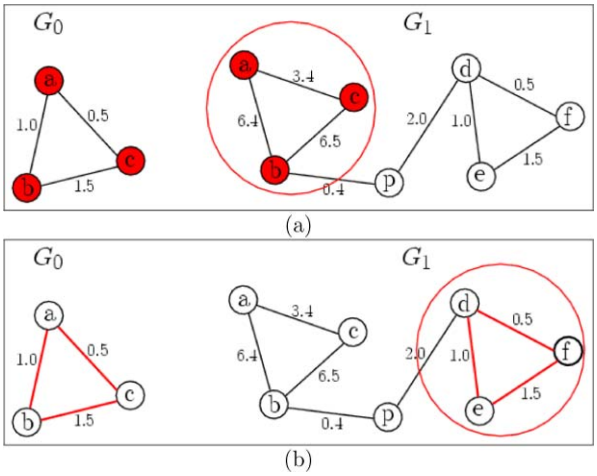


Fig. 1 Matching with labels and weighted edges

phism as follows.

$$\min \|\mathbf{A_0} - \mathbf{PA_1P^T}\|, \tag{1}$$

where $\mathbf{A_0}$ and $\mathbf{A_1}$ denote the adjacency matrices of the two edge weighted graphs and $\mathbf{P}$ denotes a permutation matrix. Hence, the problem reduces to finding one to one correspondences between the vertex sets of G_1 and G_2 such the graphs become 'close' to each other. As (1) shows, this is equivalent to finding a permutation matrix, P that minimizes the difference of one graph with the permuted version of the other. One can immediately see that this problem is closely related to the Quadratic Assignment Problem (QAP). To obtain a solution, the authors solve the linearized model corresponding to (1) and then apply the Hungarian method to obtain 0–1 solutions.

The algorithm in Almohamad and Duffuaa (1993), while being the first of its kind, has a few limitations. First, this algorithm cannot be directly applied to the isomorphism problem on more general pairs of graphs where the number of vertices in two graphs are different. Also, it assumes that the graphs are not vertex labeled. Hence, while the algorithm is suitable to handle a special case of matching graphs with the same number of unlabeled vertices, extending the algorithm to the more general case of *vertex labeled* graphs with *weighted* edges adds a new realm of complexity to the problem. To see this, let us consider the factors contributing to the edit cost in general graphs. Normally, 'edits' are performed on nodes and edges and can be broadly classified as

1. Node insertion: $f_v(u)$, Node deletion: $f_v(u)$
2. Node substitution: $d_v(f_v(u_1), f_v(u_2))$
3. Edge insertion: $f_e(e)$, Edge deletion: $f_e(e)$.

The problem of generalized graph isomorphism becomes rather ill-posed for the case where the costs (for nodes *and* edges) are not defined in the same metric space. To motivate this argument, let us consider an illustrative example. In Fig. 1, we seek to match graphs G_0 and G_1 in a minimal edit cost sense. Assuming vertex substitution has unit cost (e.g., Hamming distance) and edge replacement costs are in L_1 or L_2

space (where L_p refers to the p-norm distance), the matching will favor mapping Δabc in G_0 to Δdef in G_1 instead of Δabc. Clearly, the matching is a trade-off between competing influences.

For the special case of graphs with labeled vertices and unweighted edges, Justice and Hero (2006) very recently proposed a modification of the algorithm in Almohamad and Duffuaa (1993) to define the Graph Edit Distance in terms of an Integer Linear Programming formulation. This technique allows G_0 and G_1 to have unequal number of labeled vertices as follows. An edit grid is constructed, given as a complete graph, $G_\Omega = (V_\Omega, E_\Omega)$ where $V_\Omega = V_1 \cup V_2$ and $\max(|V_1|, |V_2|) \leq |V_\Omega| = N \leq |V_1| + |V_2|$, $N \neq 0$. The vertex labels belong to an alphabet, Σ. First, the nodes (and edges) on the edit grid are initialized to ϕ (and 0). Then, the initial graph G_0 is placed on this edit grid. The ϕ-labeled vertices of the edit grid take the labels of the vertices of G_0 that are aligned with them. All edges in G_Ω (labeled 0) are converted to 1 if they denote a real edge in G_0. An edit operation on this *standard placement* of G_0 is then defined as either changing the label of a vertex from a character in Σ to ϕ denoting deletion or from ϕ to a character in Σ denoting insertion. Edit operations on edges are denoted likewise using $0 \to 1$ or $1 \to 0$ transitions. This amounts to permuting the standard placement of G_0 to minimize the following objective.

$$\min \sum_{i=1}^{N} \sum_{j=1}^{N} d(f_v(A_0^i), f_v(A_1^j))P^{ij} + \frac{1}{2}\|\mathbf{A_0} - \mathbf{P}\mathbf{A_1}\mathbf{P^T}\|, \tag{2}$$

where $\mathbf{A_0}$ and $\mathbf{A_1}$ are the adjacency matrices of the standard placements of G_0 and G_1 on the edit grid and A_0^i (and A_1^j) is the ith (and jth) vertex of $\mathbf{A_0}$ (and $\mathbf{A_1}$) and $d \in \{0, 1\}$ is the cost function for the vertices.

While vertex and edge edit costs are in the same space (binary), (2) does not adequately capture the notion of *similarity distance*. To see this, consider two to-be-matched input graphs with vertices having large degree (≥ 3), as shown in Fig. 2 (nodes c and x). If vertices with mismatched labels from the two graphs are aligned, one naturally pays a cost of 1 for a single vertex pair mismatch, see $x \to c$ in Fig. 2 (second column). This is clearly small compared to the cost incurred when vertices having a large difference in degree are matched to each other, see $c \to c$ in Fig. 2 (first column). A natural tendency of the algorithm, therefore, would be to match vertices with the *same degree*, instead of vertices with the *same label*. In fact, in pathological cases labels may be completely marginalized in favor of same-degree-vertex alignments, especially if the vertices have high degree. While this is in accordance with the edit cost definitions, but in most applications, the semantic meaning of a vertex in a graph is as important as its structural relationship with other nodes. For example, consider matching a 'C' (carbon) to a 'H' (hydrogen) simply because they share bonds with the same number (degree) of atoms.

The above discussion suggests that using edit distance as a cost function in many applications yields a biased weighted matching where a degree mismatch has a higher penalty. A generalized distance function may be more suitable that considers the cost of replacing vertices to reflect the vertex labels and the associated edge information (structure) concurrently. We will discuss these issues in the next section.

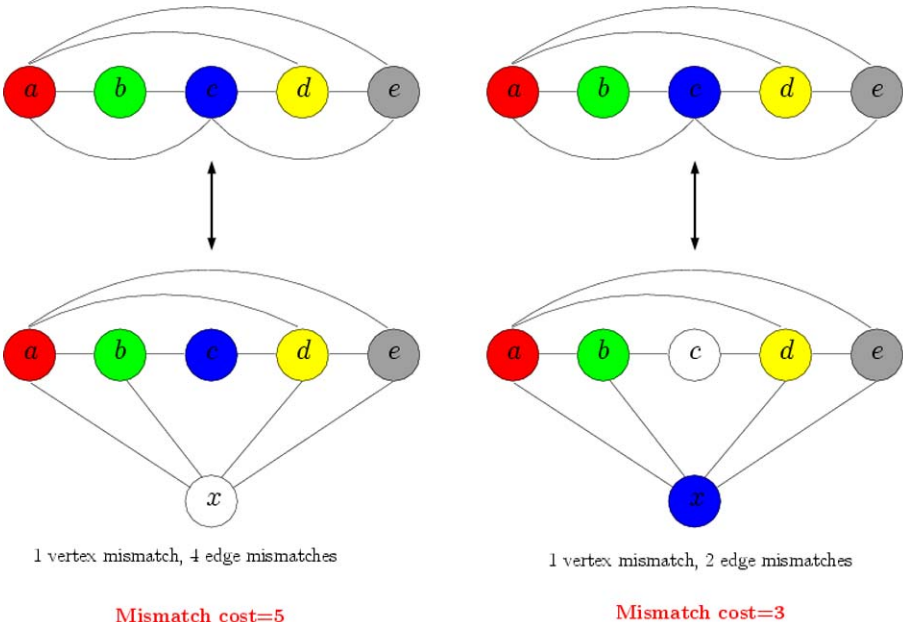

a
b
c
d
e
x
1 vertex mismatch, 4 edge mismatches
Mismatch cost=5
1 vertex mismatch, 2 edge mismatches
Mismatch cost=3

Fig. 2 Two possible matchings where characters in the nodes are the *labels* and the node colors denote the alignment determined by the algorithm. The cost for vertex *and* edge mismatch is 1. In the *left figure*, the matching seems reasonable noticing that the vertices with the same labels are well aligned. The mismatch cost is 5; the right matching ignores the vertex labels and matches vertices with the same degree to get a lower mismatch cost

2 Main ideas

2.1 A suitable cost function

Consider an image registration problem where the images have colored regions or certain segmented objects. The image's graph representation would naturally encode the regions as vertices with the respective colors as their labels. Additionally, edges between the vertices may denote some 'link' between their corresponding regions or relationships between objects as observed in the image. The registration problem then requires one to match the images (or its corresponding graph link structures) in a manner that matches similar colors (regions) while maximally preserving the graph topologies. By this interpretation, there is a cost associated with the extent of similarity of two colors.[3] For example, matching a *blue* vertex with a *green* vertex has a higher cost compared to matching a *blue* vertex with a *cyan* vertex. This cost is different from edge weights; while label-to-label cost relates vertices in source and target graphs, edge weights usually denote the strength of a 'relation' between vertices in the same graph.

To address the ill-posedness problem outlined in the previous section, we adopt the following strategy—we will match the edges explicitly, however, since edges

[3]This can be chosen based on application, the L_1 distance between the two label values is one possibility.

are identified by their end vertices, this process will also match vertices implicitly. Assume two to-be-matched vertex labeled graphs, $G_0 = (V_0, E_0, f_v, f_e)$ and $G_1 = (V_1, E_1, f_v, f_e)$. The dissimilarity between the vertex labels of these two graphs is encoded in a matrix $\mathcal{S}$, $\mathcal{S}(i, j) = 0$ if $i = j$ and $\mathcal{S}(i, j) = 1$ otherwise. The presence or absence of a link between vertices is indicated by their edge weights, f_e where $f_e \in \{0, 1\}$. Therefore every edge $e = (u, v)$ has three components, the two end vertices and the link f_e. Consider a match between edge $e_i = (a, b) \in G_0$ and $e_j = (c, d) \in G_1$, this aligns the two end vertices pair of the edges, i.e., $a \to c, b \to d$. Therefore, while we do not focus on matching vertices, we can indirectly 'charge' vertices with misaligned labels. This allows us to derive the cost that must be paid for such an 'edge' matching as

$$\mathrm{cost}(e_i, e_j) = \mathcal{S}(a, c) + \mathcal{S}(b, d) + \| f_e(e_i) - f_e(e_j) \|, \tag{3}$$

where the $\mathrm{cost}(e_i, e_j) = 0$ if both end vertices of the edge pair match perfectly. Otherwise, it reflects the sum of the costs of

- misaligned vertices (based on the dissimilarity of aligned vertex labels) and
- difference of weights of the edges, which is 0 unless it is matched to a null (i.e., absent) edge.

While (3) allows evaluating the cost due to vertex mismatches given two aligned edges as parameters, we must represent the sum of such costs for a pair of graphs in an algebraically computable manner—to define *what* must be optimized. Suppose a vertex $u_1 \in G_0$ is aligned with $u_2 \in G_1$ such that $\mathcal{S}(u_1, u_2) > 0$. Each edge incident on u_1 pays a cost of $\mathcal{S}(u_1, u_2)$ due to a misaligned u_1 (we assume that $\mathcal{S}(\cdot, \cdot)$ is a metric cost). The total cost of one vertex misalignment pair ($u_1 \leftrightarrow u_2$) is then

$$D(u_1, u_2) = [\delta(u_1) + \delta(u_2)] \cdot \mathcal{S}(u_1, u_2), \tag{4}$$

where $\delta(\cdot)$ refers to the degree of the node. Notice that all edges associated with u_1 and u_2 have paid only part of their cost. The vertex misalignment cost, if any, for the other end vertices of the incident edges of u_1 and u_2, is considered at the time of evaluation of the other pairs of aligned vertices. Considering all edges, the cost function that models this is

$$\min \sum_{i=1}^{N} \sum_{j=1}^{N} D(i, j) P^{ij} + \frac{1}{2} \| \mathbf{A_0} - \mathbf{P A_1 P^T} \|. \tag{5}$$

The second term in (5) evaluates the cost of matching a 'real' edge to a null edge (see third term in (3)). However, (5) as a measure of graph similarity distance is still not entirely accurate. The calculation is perfect when an edge matches with a null edge; however, when an edge matches to another real edge, the dissimilarity, $\mathcal{S}(u_1, u_2)$, is counted twice (for $u_1 \in G_0$ and for $u_2 \in G_1$). This overestimation must be deducted to reflect the actual cost. We will discuss this shortly.

Before we proceed any further, we will first generalize the notion of graph similarity distance (in (5)) in the context of median graphs and then address the overestimation issues in the generalized setup for convenience of presentation. If there are only two graphs, the cost calculated is the similarity of one graph to the *permuted version* of the other. In our formulation, all input graphs are embedded (placed) in an edit grid, G_Ω. A permutation applied on an input graph is simply a change of placement from one position to the other. If all the input graphs are isomorphic, then it should be possible to find the same placement for all graphs, which is also the generalized median of the input set. The distance of the input graphs to the median in this case is zero. Extending this logic to the general case where input graphs are not necessarily isomorphic, our objective is to find a set of permutations—one for each graph such that the resultant set of placements are the same (or as close as possible). It can be verified that the *mean* of all such close placements will yield the generalized median for the set of input graphs. Because which permutation must be applied to a certain graph is not known in advance, we must simultaneously permute all pairs of graphs, and calculate the cost incurred. In the next section, we will introduce the integer program that models this intuition. We will then address the overestimation in (5).

2.2 Model I: Cost when both graphs are permuted

The edit grid, G_Ω, provides a common 'reference' frame in which the sum of variations (cost) between the permuted graphs can be defined and computed precisely. Let us first consider the cost definition when two graphs are simultaneously permuted onto G_Ω and then extend the definition for multiple graphs. If graphs, G_0 and G_1, are permuted by P_0 and P_1 respectively, (5) is represented as follows.

$$\sum_{i=1}^{N}\sum_{k=1}^{N}\sum_{j=1}^{N} D(i,j)P_0^{ik}P_1^{kj} + \frac{1}{2}\|\mathbf{P_0A_0P_0^T} - \mathbf{P_1^TA_1P_1}\|, \tag{6}$$

where $P_0^{ik} = 1$ indicates that $v_i \in G_0$ is permuted to position k on the edit grid. A triplet, (i, j, k), such that $P_0^{ik}P_1^{kj} = 1$ implies that $v_i \in G_0$ is permuted to the position k, and position k on the edit grid maps to $v_j \in G_1$ and so we must incur a cost, $D(i, j)$. The main difficulty in (6) involves the product term of two variable matrices, $\mathbf{P_0}$ and $\mathbf{P_1}$. Our linearization involves an additional variable $\mathbf{X} \in \Re^{N\times N\times N}$, X_{ijk} is 1 if $P_0^{ik}P_1^{kj} = 1$ and 0 otherwise. This naturally yields

$$P_0^{ik} + P_1^{kj} = 2 \quad\Longrightarrow\quad X_{ijk} = 1, \tag{7}$$

where "$\Longrightarrow$" denotes *implies that*. Equation (7) can be converted to regular linear constraints as

$$P_0^{ik} + P_1^{kj} \geq 2X_{ijk}, \qquad P_0^{ik} + P_1^{kj} - 1 \leq X_{ijk}, \quad X_{ijk} \in \{0, 1\}. \tag{8}$$

Observe that when $P_0^{ik} + P_1^{kj} = 2$, X_{ijk} must be 1 to simultaneously satisfy both constraints in (8); if $P_0^{ik} + P_1^{jk} \leq 1$, X_{ijk} must be 0. Equations (7) and (8) are hence equivalent. Incorporating these into (6) gives

$$\sum_{i=1}^{N}\sum_{j=1}^{N} D(i,j)\left(\sum_{k=1}^{N} X_{ijk}\right) + \frac{1}{2}\|\mathbf{P_0A_0P_0^T} - \mathbf{P_1^TA_1P_1}\|. \tag{9}$$

We now revisit the overestimation issues from Sect. 2.1. To calculate the overestimation in (9), we first need to determine the edge-to-edge mappings between G_0 and G_1. This is given by the "1" entries in the dot product of $\mathbf{P_0A_0P_0^T}$ and $\mathbf{P_1^TA_1P_1}$. The overestimation, say $C \in \Re^{N\times N}$, is associated with (i, j) positions with 1s and must be deducted. For presentation purposes, let us define the following notations

$$\gamma = (\mathbf{P_0A_0P_0^T} + \mathbf{P_1^TA_1P_1}), \tag{10}$$

$$\bar{\gamma} = (\mathbf{P_0A_0P_0^T} - \mathbf{P_1^TA_1P_1}), \tag{11}$$

$$\phi(\mathcal{S}, \mathbf{X})_i = \sum_{i_1=1}^{N}\sum_{j_1=1}^{N} \mathcal{S}(i_1, j_1)X_{i_1 j_1 i}. \tag{12}$$

Then,

$$C_{ij} = \begin{cases} \phi(\mathcal{S}, \mathbf{X})_i + \phi(\mathcal{S}, \mathbf{X})_j & \text{if } \gamma_{ij} = 2; \\ 0 & \text{if } \gamma_{ij} \le 1. \end{cases} \tag{13}$$

Notice that for an edge, e_{ij}, when the sum in (13) is 2, i.e., when the dot product of $\mathbf{P_0A_0P_0^T}$ and $\mathbf{P_1^TA_1P_1}$ at position (i, j) is 1, C_{ij} is non-zero (i.e., equal to the overcharged cost of matching the end vertices of e_{ij}). Here, i and j denote the indices of vertices of the individual graphs *after* permutation, these can be mapped back to the original indices of those vertices in G_0 and G_1 using X. Let M be the maximum mismatch cost of an edge (computed offline, see (3)). The conditional constraints in (13) can then be easily expressed as

$$\begin{aligned} &\phi(\mathcal{S}, \mathbf{X})_i + \phi(\mathcal{S}, \mathbf{X})_j - C_{ij} \le (2 - \gamma_{ij})M, \\ &C \le M\mathbf{P_0A_0P_0^T}, \qquad C \le M\mathbf{P_1^TA_1P_1}. \end{aligned} \tag{14}$$

Here, the constraints apply for all matrix elements. Therefore, the final objective function that takes care of the overestimation is

$$\min \quad \sum_{i=1}^{N}\sum_{j=1}^{N} D(i, j)\left(\sum_{k=1}^{N} X_{ijk}\right) + \frac{1}{2}\|\bar{\gamma}\| - \underbrace{\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} C_{ij}}_{\text{overestimation}}, \tag{15}$$

$$\begin{aligned} &\mathbf{P_0^T}e = e; \quad \mathbf{P_1}e = e; \quad \mathbf{P_1^T}e = e, \\ &P_0^{ik} + P_1^{kj} \ge 2X_{ijk}, \\ &P_0^{ik} + P_1^{kj} - 1 \le X_{ijk}, \\ &\phi(\mathcal{S}, \mathbf{X})_i + \phi(\mathcal{S}, \mathbf{X})_j - C_{ij} \le (2 - \gamma_{ij})M, \\ &C \le M\mathbf{P_0A_0P_0^T}, \\ &C \le M\mathbf{P_1^TA_1P_1}, \\ &\mathbf{P_0}, \mathbf{P_1}, \mathbf{X} \in \{0, 1\}; \mathbf{C} \in [0, M]. \end{aligned} \tag{16}$$

When only one graph is permuted, we have a simpler model with only $O(n^2)$ variables and constraints. We discuss this in the next section.

2.3 Model II: Cost function when only one graph is permuted

In this section, we consider the simplified case of Sect. 2.2, i.e., when only two graphs are given. Then, we can keep the target graph fixed (which serves as an edit grid) and only the source graph is permuted; we no longer need to linearize a product term. This is also the same as finding the isomorphic mapping of one graph with respect to the other and can be used for matching two graphs. The number of variables and constraints required are only $O(n^2)$. For convenience, let

$$\psi_i(\mathbf{P}) = \sum_{l=1}^{N} \mathcal{S}(l,i)P^{il} \tag{17}$$

we thus have the following simplified Integer Program,

$$\min \quad \sum_{i=1}^{N}\sum_{j=1}^{N} D(i,j)P^{ij} + \frac{1}{2}\underbrace{(\mathbf{S}+\mathbf{T})}_{\text{slack variables}} - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} C_{ij} \tag{18}$$

$$\begin{aligned} \text{s.t.} \quad & \mathbf{P}e = e; \quad \mathbf{P^T}e = e, \\ & \mathbf{A_1P} - \mathbf{PA_0} + \mathbf{S} - \mathbf{T} = 0, \\ & \psi_i(\mathbf{P}) + \psi_j(\mathbf{P}) - C_{ij} \le (2 - (\mathbf{PA_0} + \mathbf{A_1P})_{ij})M, \\ & C \le M\mathbf{A_1P}; \quad C \le M\mathbf{PA_0}, \\ & \mathbf{S}, \mathbf{T}, \mathbf{P}, \mathbf{X} \in \{0,1\}; \mathbf{C} \in [0, M]. \end{aligned} \tag{19}$$

The relaxation of this Integer Program can be solved optimally in polynomial time. We can then round the solution to obtain integral solution values. We discuss the rounding in Sect. 4.1.1.

3 Linearization and relaxation

In this section, we describe a relaxation technique to derive a polynomial time solvable version of the problem in Sect. 2.2. We illustrate the ideas using the familiar objective function in (1) in Sect. 1.2. The strategy applies readily to (15)–(16). We observe that the objective, $\sum_{ij}(\mathbf{A_1} - \mathbf{PA_2P^T})_{ij}$, is equivalent to

$$\sum_{l}(\text{vec}(\mathbf{A_1}) - \text{vec}(\mathbf{A_2})(\mathbf{P}\otimes\mathbf{P}))_l, \tag{20}$$

where $\otimes$ indicates the Kronecker product and vec$(\cdot)$ indicates "vectorizing" a matrix. Consider a matrix variable $\mathbf{Q}$ where $\mathbf{Q} = \mathbf{P}\otimes\mathbf{P}$, the nonlinearity in the second term can be addressed[4] as follows.

$$\sum_l (\mathrm{vec}(\mathbf{A_1}) - \mathrm{vec}(\mathbf{A_2})\mathbf{Q})_l, \tag{21}$$

where $\mathbf{Q} \in \Pi_{n^2}$ and Π_d denotes permutation matrices of size $d \times d$. $\mathbf{Q}$ has some additional interesting properties; specifically, the (i, j)th *block* of $\mathbf{Q}$, denoted as $Q_{[ij]}$ corresponds to element P^{ij}. Below, we use $Q(i, j)$ to denote the (i, j)th *entry* of $\mathbf{Q}$.

$$\min \quad \sum(\mathbf{S}+\mathbf{T}) \tag{22}$$

$$\begin{aligned}\text{s.t.} \quad & \mathrm{vec}(\mathbf{A_1}) - \mathrm{vec}(\mathbf{A_2})\mathbf{Q} + \mathbf{S} - \mathbf{T} = 0 \\ & \sum_{l=1}^{n} Q(k+(i-1)n, (j-1)n+l) = P^{ij}, \quad \forall i, j, k \in \{1, \dots, n\}, \\ & \sum_{i=1}^{n} Q_{[ij]} = P, \qquad \sum_{j=1}^{n} Q_{[ij]} = P, \quad \forall i, j \in \{1, \dots, n\}, \\ & \mathbf{P}e = e; \quad e^T\mathbf{P} = e^T; \quad \mathbf{P}, \mathbf{Q}, \mathbf{S}, \mathbf{T} \ge 0.\end{aligned} \tag{23}$$

Equations (23)–(23) is polynomial time solvable.

4 Generalized median graphs

A natural extension of (15)–(16) yields a model for computing the generalized median graph for a set. Let the 'distance' between two graphs as defined in (15) be given as $d_p(G_0, G_1, \mathbf{P_0}, \mathbf{P_1})$. Notice that while $d_p(\cdot)$ simply computes the cost given four parameters (no unknowns), (15) tries to optimize the distance where the permutations are unknown. Using the notation above, our objective is to permute all graphs onto the edit grid and can be expressed as

$$\min \sum_{x=1}^{K} \sum_{y=1}^{K} d_p(G_x, G_y, \mathbf{P_x}, \mathbf{P_y}), \tag{24}$$

where K is the number of graphs in the set. The 0–1 solution can be obtained by rounding the fractional entries of the K permutation matrices (discussed in detail in Sect. 4.1.2) and obtaining the median graph as a mean of the permuted input graphs. This yields a polynomial time algorithm for computing the generalized median graph, no bounded running time algorithms were known before.

Our experimental results using this model are discussed in Sect. 5. In general, this approach can be used for reasonably sized graphs (no assumptions on the structure) and yields good results in practice. Since the number of variables in the model are $O(N^4K)$, for large graphs some application specific heuristics may need

[4]The relaxation in Almohamad and Duffuaa (1993) works perfectly when only one graph is permuted, we use (1) only as an example.

to be employed. In the following sections, employing the ideas above and additional observations we discuss a hybrid approach that yields a factor two approximation ratio. Finally, we propose another bi-level algorithm that finds a solution arbitrarily close to the optimal (though running time is non-polynomial in worst case).

We employ the concept of generalized median for a set of objects in metric spaces (Jiang and Bunke 2002) as a first step. Let the distance (discussed above) between two graphs be given as $d(G_0, G_1) = d_p(G_0, G_1, \mathbf{I}, \mathbf{P_1})$. The best choice of $\mathbf{P_{uv}}$ corresponding to G_u and G_v gives a $d(\cdot, \cdot)$ that has the following property.

Lemma 1 (Metric) *For any three graphs G_p, G_q and G_r, $d(\cdot, \cdot)$ is a metric.*

Proof Identity. For isomorphic graphs, with the correct permutation (aligned identical vertices and edges), (15) evaluates to 0. For the 'only if' part, consider that a permutation that aligns a vertex incorrectly will allow the first term of the objective to be non-zero. For an incorrectly aligned edge, a similar argument holds considering either the first term or the second and third terms together.
Symmetricity. The permutations $\mathbf{P}$ and $\mathbf{P^T}$ in the forward and reverse directions are vertex and edge mapping preserving and correspond to the same cost w.r.t. (15).
Non-negativity. The third term of (15) is negative. However, it is upper bounded by the first term i.e., $\sum_{i=1}^N \sum_{k=1}^N \sum_{j=1}^N D(i, j) P_0^{ik} P_1^{kj}$. While the first terms account for the cost for all edges coming out of vertices that are matched, the third is the sum of costs when an edge is matched to another edge (a subset of). Therefore, for a permuation matrix we have

$$\sum_{i=1}^{N} \sum_{j=1}^{N} C_{ij} \leq \sum_{i=1}^{N} \sum_{k=1}^{N} \sum_{j=1}^{N} D(i, j) P_0^{ik} P_1^{kj}.$$

Triangle inequality. Suppose triangle inequality does not hold. Then, the sum of distances corresponding to $\mathbf{P_{pq}}$ and $\mathbf{P_{qr}}$ is less than the value of (15) for $\mathbf{P_{pr}}$. We may construct a new matrix of mappings $\bar{\mathbf{P}}_{\mathbf{pr}} = \mathbf{P_{pq}} \mathbf{P_{qr}}$ which yields a smaller value of (15) compared to $\mathbf{P_{pr}}$. But $\mathbf{P_{pr}}$ minimizes (15) and we have the desired result. □

Let $\mathbb{G} = (G_0, G_1, \ldots, G_K)$ be a collection of graphs with $|V_i| = |V_j|, \forall i, j$. Dummy nodes can be introduced if $|V_i| \neq |V_j|$, so $N = \max(|V_1|, \ldots, |V_n|)$. Let G^* be the optimal median of this set. By definition, G^* satisfies

$$G^* = \arg\min_{\bar{G}} \sum_{i=1}^{K} d(\bar{G}, G_i).$$

We will avoid computing G^* directly. Instead, we employ the lower bounding ideas in Jiang and Bunke (2002) to model the distance of every graph, G_i to the generalized median by an unknown variable, x_i. Therefore, $x_i = d(G^*, G_i)$. Then, the problem can be modeled as follows.

$$\min \quad \sum_{i=1}^{K} x_k \tag{25}$$

$$\begin{aligned} \text{s.t.} \quad & x_k + d(G_k, G_l) \geq x_l, \quad \forall k, l, \\ & x_l + d(G_k, G_l) \geq x_k, \quad \forall k, l, \\ & x_k + x_l \geq d(G_k, G_l), \quad \forall k, l, \\ & x_i \geq 0, \quad \forall i \in \{1, \dots, K\}. \end{aligned} \tag{26}$$

Equations (25)–(26) can be solved optimally in polynomial time.

Property 1 (From Jiang and Bunke 2002) *The sum of the entries of* $\mathbf{x} = (x_1, \dots, x_K)^T$ *denotes a lower bound on the distance of all graphs to the optimal generalized median.*

Lemma 2 *There must be a graph* $G_a \in \mathbb{G}$ *that satisfies*

$$\sum_{i=1}^{K} d(G_k, G_a) \leq 2 \sum_{i=1}^{K} d(G_k, G^*),$$

where G^* *is the optimal median graph for* G.

Proof Let $\mathbf{x}^* = (x_1^*, \dots, x_K^*)^T$ be an optimal solution for (25)–(26). Consider a graph $G_a \in \mathbb{G}$ s.t. $a = \{j : x_j^* \leq x_i^*, \forall i \in \{1, \dots, K\}\}$. From (26), we know $\forall G_i, d(G_i, G_a) \leq x_i^* + x_a^*$. Since $x_a \leq x_i^*$, $d(G_i, G_a) \leq 2x_i^*$. Therefore $\sum_{i=1}^{K} d(G_i, G_a) \leq 2 \sum_{i=1}^{K} x_i^*$. Applying Property 1, the lemma follows. □

Theorem 1 *The graph* $G_a \in \mathbb{G}$, $a = \{j : x_j^* \leq x_i^*, \forall i \in \{1, \dots, K\}\}$, *is a* 2*-approximation for the generalized median graph problem if* $d(\cdot, \cdot)$ *is known.*

If $d(\cdot, \cdot)$ is known correctly, the factor two approximation (for the Hybrid algorithm) holds by comparing the solutions of (25)–(26) and (15)–(16) and outputting the better solution.

4.1 Rounding

4.1.1 Rounding the solution of Model 2 in Sect. 2.3

We would like to convert the fractional solution of **P** into a 0–1 permutation matrix which is ‘close’ to **P**. This is done as follows. We construct a bipartite graph, $G = (V, E)$ with subgraphs G_1 and G_2, where the indices of rows in P denote vertices of G_1 and indices of columns in **P** are given by the vertices of G_2. Since **P** is of size $n \times n$, $|V| = 2n$. The edge set, E, has exactly $\frac{n^2}{2}$ members because we join every

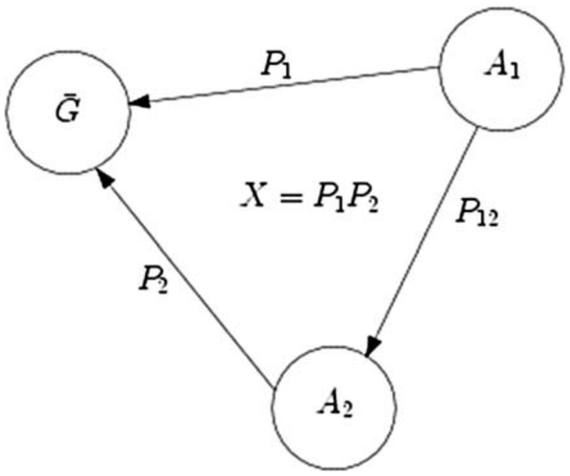


Fig. 3 Permutation relations for two graphs in the input set and $\bar{G}$

vertex in G_1 to every vertex in G_2. The weight of an edge, $w(e)$, between $v_i \in G_1$ and $v_j \in G_2$ is equal to the entry $P(i, j)$. The optimal solution to the bipartite matching problem can be obtained efficiently. It is easy to prove that this yields the closest permutation matrix to **P**.

4.1.2 Rounding for generalized median graph model

The above strategy can also be directly adopted for the generalized median graph case with multiple graphs. However, we noticed that better empirical results can be obtained by adopting the following alternate strategy. Consider the schematic diagram in Fig. 3 where G_1 and G_2 are any two graphs in the input set and $\mathbf{P_1}$ and $\mathbf{P_2}$ are their unknown permutations to the median, $\bar{G}$. Let P_{12} be the permutation matrix for a 'matching' between G_1 and G_2. Recall that the solution to the LP in Sect. 4 yields not only $\mathbf{P_1}$ and $\mathbf{P_2}$ (both fractional) but also $\mathbf{X_{12}} = \mathbf{P_1}\mathbf{P_2}$. Thus, a *rounded* X_{12} must also be a permutation matrix. Now, if G_1, G_2 and $\bar{G}$ were isomorphic, we will have $\mathbf{P_1}(\mathbf{P_2})^{\mathbf{T}} = \mathbf{P_{12}}$. Even where they are not perfectly isomorphic, we typically have the following relationship,

$$\mathbf{P_1}(\mathbf{P_2})^{\mathbf{T}} \approx \mathbf{P_{12}}. \tag{27}$$

It is known that for permutation matrices we have the property, $\mathbf{P}^{-\mathbf{1}} = \mathbf{P}^{\mathbf{T}}$; so we have $\mathbf{P_2} = \mathbf{P_1}^{\mathbf{T}}\mathbf{X_{12}}$. Substituting this into (27), we get

$$(\mathbf{P_1})^{\mathbf{2}}\mathbf{X}_{\mathbf{12}}^{\mathbf{T}} \approx \mathbf{P_{12}}, \tag{28}$$

$$(\mathbf{P_1})^{\mathbf{2}} \approx \mathbf{P_{12}}\mathbf{X_{12}}. \tag{29}$$

Using this idea, we can first obtain an estimated value of $(\mathbf{P_1})^2$ by taking the product of $\mathbf{X_{12}}$ and $\mathbf{P_{12}}$. This can then be factorized using Singular Value Decomposition and rounded using bipartite matching techniques as in Sect. 4.1.1. We notice that this

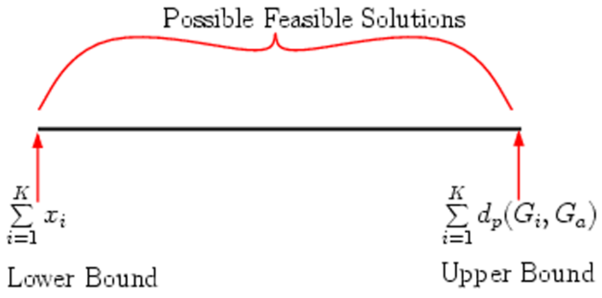


Fig. 4 Feasibility search

alternate process yields superior results for multiple graphs compared to a straightforward extension of bipartite matching based rounding discussed earlier.

4.2 Alternate bi-level algorithm (a sketch)

In this section, we briefly outline our alternate bi-level algorithm to obtain even better solutions.

Let $U = \sum_{i=1}^{K} d(G_k, G_a)$. We know that the optimal solution lies in $[\sum_i x_i^*, U]$ and any feasible solution in this range is a factor two approximation. We adopt a binary search in $[\sum_i x_i^*, U]$ starting at a value $c = \sum_i x_i^*$ and solving a feasibility problem (FP) at each step. The objective function of the FP is max 0 subject to the constraints that the sum of distances of $G_i \in G$ to an unknown graph $\hat{G}$ is upper bounded by c. Here the adjacency matrices of $\hat{G}$ and P_i are variables. The constraints are defined by (15)–(16). If this returns a feasible solution, we are done (i.e., $\hat{G} = G^*$). Otherwise, we continue the binary search (see Fig. 4). The FP can be solved using branch-and-bound type methods (available in commercial solvers) but may have non-polynomial running time in the worst case.

5 Experimental results

We present our experimental evaluations in three parts. In the first subsection, we evaluate the generalized cost model on a publicly available graph database for isomorphism testings. We follow this with evaluations of median graph determination given a collection of graphs. In the next subsection, we discuss experiments in context of our primary motivating application in biological image analysis. Finally, we discuss an application of median graphs to drug design. The numerical results are based on an implementation in C++ using CPLEX as the linear program solver.

5.1 Database evaluations

From the *Graph Database* (Foggia et al. 2001) (see http://amalfi.dis.unina.it/graph/), we selected about 900 labeled (both vertices and edges have labels) graphs (files with the prefix *r005*). All combinations of the parameters, *size* $= \{20, 25, 40\}$, and *isomorphism percentage(mcs)* $= \{90\%, 70\%, 50\%\}$, were considered. For every unique

Table 1 Table of isomorphism evaluations on graph database

Size	Persentage of isomorphism		
	*mcs*90	*mcs*70	*mcs*50
20	93.62%	87.18%	87.26%
25	92.32%	82.96%	84.67%
40	86.48%	81.87%	81.25%

combination, all isomorphism pairs were evaluated and the mean of the errors analyzed. We considered the vertex attributes alone which were normalized to a smaller integer numeric scale. The dissimilarity matrix $\mathcal{S}$ of vertex labels was created as follows. Let G_0 and G_1 be two input graphs and let d_{G_l,L_i} denote the degree of vertex v_i with label L_i in graph G_l. Then the dissimilarity between vertices $v_i \in G_0$ and $v_j \in G_1$ is given as

$$\mathcal{S}_{ij} = \begin{cases} 0 & \text{if } L_i = L_j, \\ \max(\|L_i - L_j\|, d_{G_0,L_j} + d_{G_1,L_i}) & \text{if } L_i \neq L_j. \end{cases}$$

The second case above appropriately penalizes matching of vertices with different labels (in general, $\mathcal{S}$ depends on the application).

The average percentage of correct mappings for each of nine combinations is shown in Table 1. When the graphs are close to isomorphic (e.g., $mcs90$) albeit with arbitrary orientations and embedding, our algorithm does quite well ($\sim 90\%$) in finding the permutations that matches similar labels together while maximally preserving the graph structure at the same time. In the extreme case ($mcs50$), when only 50% of the graphs are isomorphic, our algorithm still performs well and achieves about 83% accuracy on an average.

We now discuss the experimental evaluations of our generalized median graph algorithm. The idea is to generate a base graph and then create a new set of graphs from the base by randomly adding noise. The median determined can then be evaluated against the base graph. The base graph was generated as in Foggia et al. (2001) and served as the truth (known median). For the set of input graphs, error was added (vertices/edges added and deleted) in the base graph. The probability of error in an edge (and error magnitude) connecting two vertices of a graph was considered independent of the vertices. The computed generalized median graph was compared to the base graph, the mean of the errors (w.r.t. similarity distance) normalized by the size of the base graph was then analyzed. Here, $N = 6$, $K \in [4, 10]$ and 30% error was introduced (in Figs. 5(a), 5(c)) in the vertices and edges. In Figs. 5(b) and 5(d), the percentage of error introduced was varied. In Fig. 5(a), we evaluate the performance as a function of the number of graphs in the input set. The y-axis in the plot shows the normalized similarity distances of every graph to the base graph (under best permutation) calculated as above. Observe that the average characterizes the amount of 'noise' in the input set when the best possible permutations are applied on the graphs. We can see that the algorithm performs well and consistently finds a generalized median that is close to the base graph. The deterioration in performance with a larger set of graphs (around 10) is only marginal. We can see the same trend when the errors introduced in the set are increased to up to 50% in Fig. 5(b). In Figs. 5(c) and 5(d),

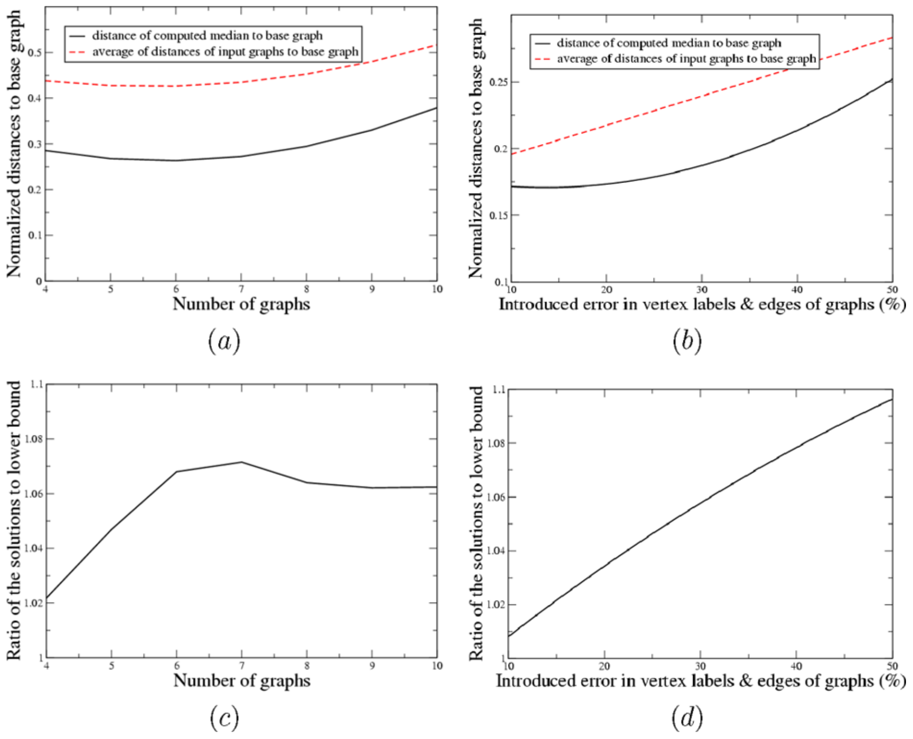


Fig. 5 Average of the distances of input set graphs and the distance of the computed median to the base graph w.r.t. the number of graphs (with 30% error) in (**a**) and introduced error in (**b**). Ratios of the median to the lower bound in (26)–(27) w.r.t. the number of graphs in (**c**) and introduced error in (**d**)

we repeat the same evaluations as above; however, we compare the performance of our algorithm with the lower bound. This gives us a reference to evaluate the similarity distance of the computed median to the best (not necessarily existent) solution. In both plots, we can see that the solutions are very close to the optimal.

5.2 Applications in biological image analysis

5.2.1 Brief biological background

Chromosomes within the human cell nucleus are known to occupy discrete territories and evidence from recent literature suggests that these might be functionally relevant because the position of these territories in nuclear space is 'organized' (Cremer and Cremer 2001). For example, chromosomes can be ordered radially from the center to the periphery of the nucleus. This organization shows a statistical correlation with gene density and the size of chromosomes (Tanabe et al. 2002). In humans, larger chromosomes (numbers 1–10) are arranged on the periphery. Secondly, chromosomes may have non-random neighbors (Nagele et al. 1998b), a finding that might account for preferential translocations and interactions between specific chromosomes. Biologists suspect that such patterns are related to genomic evolution and

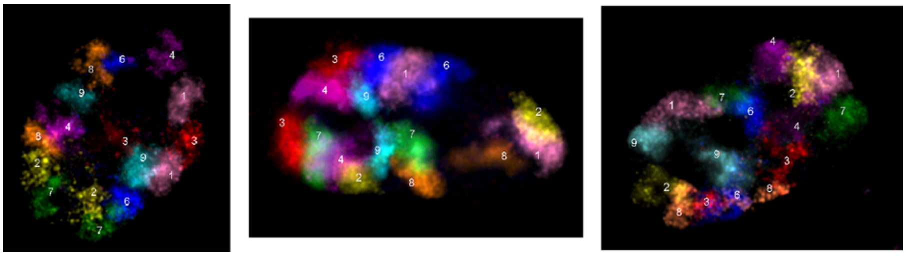


Fig. 6 Three nuclear images (out of a set of thirty-seven) with eight chromosome paints

cell-type-specific variability. In this regard, several researchers (Boyle et al. 2001; Nagele et al. 1998a) have stressed on the need of mapping large-scale organization and distribution of chromatin in various cell lines. The standard approach relies on a sequence of experiments on a large number of cells. Chromosome to chromosome relationships are manually investigated using the acquired images (see Fig. 6, one for every cell) sequentially. A reliable topological 'model' can provide the necessary foundation for studying the effect of higher-order chromatin distribution on nuclear functions. Unfortunately, an arrangement map for all 23 chromosome pairs in a diploid human cell nucleus is lacking so far. Such models are needed for different cell types at various stages of cell cycle and terminal differentiation (Boyle et al. 2001).

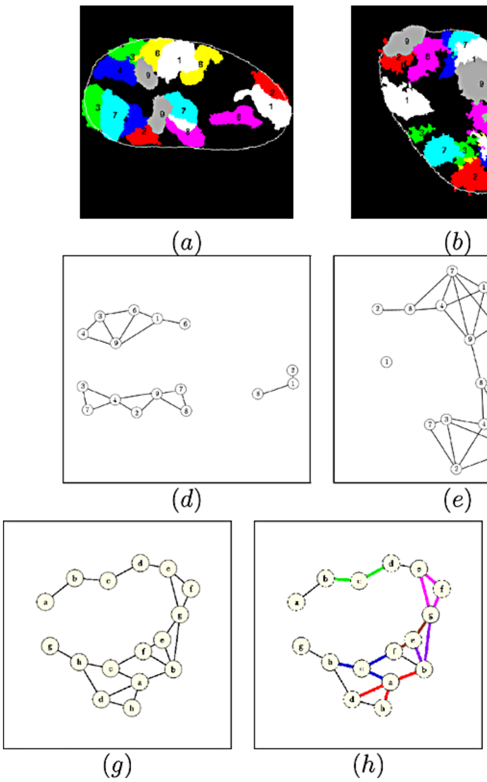
5.2.2 Chromosomal organization maps

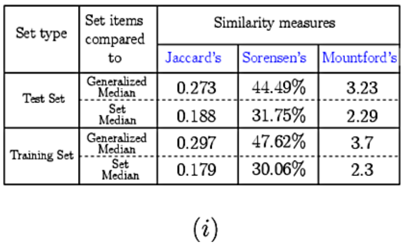
Topological map derivation of chromosomes transforms naturally to the generalized median graph problem. Each individual chromosome (denoted as numbers in Fig. 6) is represented as a labeled vertex in a graph. Due to microscopic acquisition limitations, at most 8 pairs of chromosomes can be 'labeled' per cell; hence, our graph has 16 vertices with at most 2 vertices with the same vertex label (i.e., chromosome numbering). Spatial proximity (adjacency) between chromosomes is expressed as an edge between vertices. We focus on 8 pairs of the larger chromosomes (numbers $1, 2, 3, 4, 6, 7, 8, 9$) from the *human lung fibroblast cell line*. The acquisition process was repeated for 37 cells overall. The raw images were segmented to create masks for individual chromosome pairs and individual graph representations derived (see Fig. 7).

To evaluate the goodness of our final 'prototype model' for this cell line we divide the cells into a training set (19 cells) and a test set (18 cells). We report on the following observations.

Optimality Figure 7(g) shows the median graph obtained from the training set. The objective function value was 1.43 times the lowest possible cost for computing the median for this set of graphs (lower bound by (25)–(26)). This ratio for the {training, test, entire} set medians were {1.58,1.6,1.63} indicating that the computed generalized median is a better representative for the set than any of the component graphs.

Substructure frequency We generated groups of chromosomes which occur frequently as a chain or a sub-group (e.g., numbers (b, c, d)) by graph mining and other





Set type	Set items compared to	Similarity measures		
		Jaccard's	Sorensen's	Mountford's
Test Set	Generalized Median	0.273	44.49%	3.23
	Set Median	0.188	31.75%	2.29
Training Set	Generalized Median	0.297	47.62%	3.7
	Set Median	0.179	30.06%	2.3

Fig. 7 2D sections of chromosome organization in 3 cell-images in (**a**)–(**c**) and their graphical representation in (**d**)–(**f**); a generalized median graph for a set of 19 cells in (**g**); all patterns (except one) with frequency of $\geq$70% across 19 cells (by graph-mining) were present in the generalized median graph and shown as colored in (**h**); average of similarity measures of items in the test/training sets w.r.t. to the generalized median and set median of those sets in (**i**)

available software (ibm 2004; Mukherjee et al. 2006). We then compared these to the results obtained by the median graph model for this cell line. Figure 7(h) shows groups (colored edges) with an occurrence frequency of $\geq$ 70%. Figure 7(h) shows that almost all such sub-graphs align *perfectly* with the computed median graph indicating that the model incorporates the essential information from the constituent graphs.

Structure prediction The basic question we wanted to answer was—given a new cell, can we predict the non-random part of its chromosomal organization? If yes, then the model serves its purpose and can be used to interpret biologically relevant information. To do this we calculated Jaccard's, Sorensen's, and Mountford's similarity of the median (from training set) with each cell in the training/test set (considering attributed edges as set members). The results in Fig. 7(i) show that the similarity is only slightly worse for the test set w.r.t. the training set. Considering that Sorensen's is a percentage similarity measure, we can draw the following inference—given a

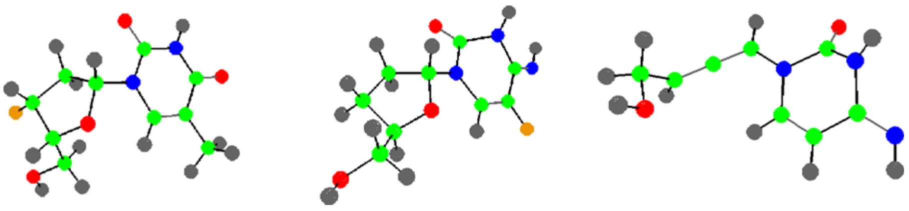


Fig. 8 Three among 12 chemical structures in the Pyrimidine Nucleosides input set. The structures (*left to right*) correspond to Thymidine, 3′-deoxy-3′-fluoro- (8CI 9CI), 2′,3′-Dideoxy-5-fluorocytidine, and N-[(Dimethylamino)methylene]-5-fluoro-2′,3′-dideoxycytidine. Different colors correspond to different atoms as follows: *gray* for H, *green* for C, *red* for O, *blue* for N, and *brownish-yellow* for F atoms

new cell, we can predict close to 50% of its chromosome organization (note that the cells are *not* isomorphic).

5.3 Application to drug design

Drug design involves finding drugs or small molecules that bind with a target molecule in the human body. The target molecule is biologically related to one or more disease conditions; the process of the drug molecule *binding* to the target inhibits the metabolic pathways specific to certain diseases.

Therefore, one important aspect of drug design is to identify the target molecule—this enables the design of *target specific* drugs. Unfortunately, in many scenarios, the chemical composition of the target molecule is not known accurately. To address this problem, researchers typically try a number of drug molecules with varying chemical structures to account for the variations in the target molecule. An additional problem stems from the fact that all molecules that bind to a specific target site (called active molecules) may not be suitable for use in drug design. For instance, they may have certain undesirable properties such as toxicity and instability. Therefore, given a number of molecules that bind to a target molecule, the task is to determine their structural similarities, and hence design a 'model structure' called *pharmacophore* such that one or more active molecules share structural similarity with the pharmacophore. In other words, a pharmacophore is a three-dimensional map of biological properties common to a large fraction of molecules that exhibit a particular activity. Hence, they serve as a *design template* in drug design.

Existing techniques for pharmacophore discovery rely on geometric point hashing (assuming that the structures are described as sets of points), inductive logic programming (Finn et al. 1998; Muggleton and De Raedt 1994) or graph mining (Mahé et al. 2006), also see Cook and Holder (2006). Frequent substructure mining in graphs considers the presence or absence of an edge alone (0 or 1) and such approaches cannot consider either the extent of dissimilarity of labels or the weight of the edges. Additionally, once substructures of small sizes have been mined, inferring the structure of the molecule by stitching together individual substructures relies on expert knowledge. Inductive Logic Programming, on the other hand is more general and learns logical rules that are valid in the given dataset; several widely used pharacophore discovery algorithms are based on ILP methods.

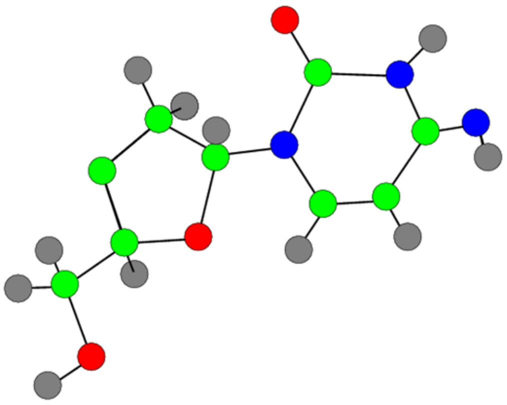


Fig. 9 The computed median graph for the input set of structures in Pyrimidine Nucleosides

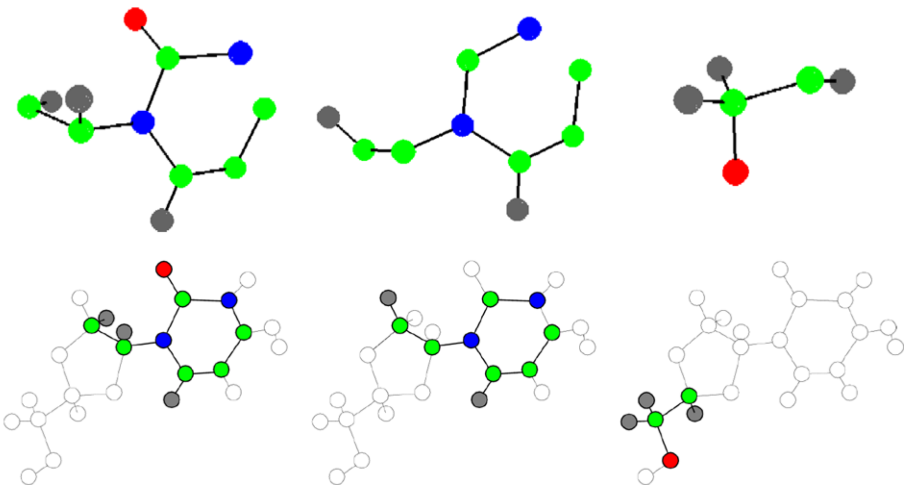


Fig. 10 Three frequently occurring substructures in the Pyrimidine Nucleosides set (*top row*) and their placement in the computed median (*bottom row*). All frequently occurring substructures were found in the generalized median graph. Note that because the model (*median graph*) had been computed, we could find the position of the substructures in the model

While sophisticated techniques and algorithms have been developed for this problem, we were interested in assessing the usefulness of the median graph formulation in this application. To evaluate this further, we used a set of 12 active compounds from the class of Pyrimidine Nucleosides selected from AIDS Antiviral Screen Database located at National Cancer Institute website at http://nci.cambridgesoft.com/ (a few are shown in Fig. 8). The size of molecules varied from 22 to 45. In these compounds, a single atom was present at most 22 times in a molecule. Given the large size of the compounds, we used a variation of the median graph algorithm proposed earlier (see Sect. 4) to speed up computation. Specifically, instead of assuming that the orientation of the median graph is unknown, we selected one of the input graphs as the base orientation. We then permuted all other input graphs to match with the first graph. The permuted orientation of all graphs was then averaged to generate the median graph. We repeated the process for each input graph—generating a new graph (and corresponding cost value) at each iteration. Ultimately, the best graph based on the objective function value (see (6)) was chosen as the median graph.

The generalized median graph is shown in Fig. 9. The objective value for the computed generalized median was slightly better ($\sim$2%) than the set median.

To evaluate how the frequently occurring substructures fit onto the median graph, we found all frequent subgraphs in the input set (frequency of $\geq$95%) by using standard graph mining software (Yan and Han 2002; ibm 2004). Some of these are shown in Fig. 10. Note that the graph mining software simply returns edge-groups that appear with high frequency; constructing a 'model' minimizing a distance function given frequently occurring substructures is not an easy task. Nonetheless, such frequent substructure information serves as a good verification tool; because the generalized median graph for the set had already been computed, we examined if the frequently occurring edge-groups were present in our model. We can see in Fig. 10, that all frequent substructures were also found in the computed median graph.

References

Almohamad HA, Duffuaa SO (1993) A linear programming approach for the weighted graph matching problem. IEEE Trans Pattern Anal Mach Intell 15(5):522–525

Boblaender HL (1990) Polynomial algorithms for graph isomorphism and chromatic index on partial k-trees. J Algorithms 11(4):631–643

Boykov Y, Veksler O, Zabih R (2001) Fast approximate energy minimization via graph cuts. IEEE Trans Pattern Anal Mach Intell 23(11):1222–1239

Boyle S, Gilchrist S, Bridger J, Mahy NL, Ellis JA, Bickmore WA (2001) The spatial organization of human chromosomes within the nuclei of normal and emerin-mutant cells. Hum Mol Genet 10:211–219

Brandes U, Gaertler M, Wagner D (2003) Experiments on graph clustering algorithms. In: Proc european symposium on algorithms, pp 568–579

Cook DJ, Holder LB (2006) Mining graph data. Wiley, New York, Chap 14

Corneil DG, Gotlieb CC (1970) An efficient algorithm for graph isomorphism. J Assoc Comput Mach 17(1):51–64

Cremer T, Cremer C (2001) Chromosome territories, nuclear architecture and gene regulation in mamalian cells. Nat Rev Genet 2:292–301

Finn PW, Muggleton S, Page D, Srinivasan A (1998) Pharmacophore discovery using the inductive logic programming system PROGOL. Mach Learn 30(2–3):241–270

Foggia P, Sansone C, Vento M (2001) A database of graphs for isomorphism and sub-graph isomorphism benchmarking. In: Proc IAPR workshop on graph-based repr

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

Horaud R, Skordas T (1989) Stereo correspondence through feature grouping and maximal cliques. IEEE Trans Pattern Anal Mach Intell 11(11):1168–1180

IBM frequent subgraph miner (2004)

Jiang X, Bunke H (2002) Optimal lower bound for generalized median problems in metric space. In: Proc of IAPR structural, syntactic, and statistical patt recog, pp 143–152

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

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

Mahé P, Ralaivola L, Stoven V, Vert J (2006) The pharmacophore kernel for virtual screening with support vector machines. J Chem Inf Model 46(5):2003–2014

Muggleton S, De Raedt L (1994) Inductive logic programming: Theory and methods. J Log Program 19(20):629–679

Mukherjee L, Singh V, Peng J, Xu J, Zeitz MJ, Berezney R (2007) Generalized median graphs: Theory and applications. In: Proc international conf on computer vision

Mukherjee L, Singh V, Xu J, Malyavantham KS, Berezney R (2006) On mobility analysis of functional sites from time lapse microscopic image sequences of living cell nucleus. In: Proc medical image computing and computer-assisted intervention, pp 577–585

Nagele R, Freeman T, McMorrow L, Lee H (1998a) Precise spatial positioning of chromosomes during prometaphase: Evidence for chromosomal order. Science 270:1830–1835

Nagele R, Freeman T, McMorrow L, Thomson Z, Kitson-Wind K, Lee H (1998b) Chromosomes exhibit preferential positioning in quiescent human cells. J Cell Biol 112:525–535

Ng RT, Han J (1994) Efficient and effective clustering methods for spatial data mining. In: Proc conf on very large data bases, pp 144–155

Parada L, McQueen PG, Munson PJ, Misteli T (2002) Conservation of relative chromosome positioning in normal and cancer cells. Curr Biol 12(19):1692–1697

Phillips CA, Warnow T (1996) The asymmetric median tree: a new model for building consensus trees. Discrete Appl Math 71(1–3):311–335

Shi J, Malik J (2000) Normalized cuts and image segmentation. IEEE Trans Pattern Anal Mach Intell 22(8):888–905

Siddiqi K, Shokoufandeh A, Dickinson SJ, Zucker SW (1998) Shock graphs and shape matching. In: Proc international conf on computer vision, pp 222–229

Tanabe H et al (2002) Evolutionary conservation of chromosome territory arrangements in cell nuclei from higher primates. Proc Natl Acad Sci 99(7):4424–4429

Toda S (1999) Graph isomorphism: its complexity and algorithms. In: Proc conf on found of soft tech and theoretical computer science, p 341

Ullmann JR (1976) An algorithm for subgraph isomorphism. J Assoc Comput Mach 23(1):31–42

Umeyama S (1988) An eigendecomposition approach to weighted graph matching problems. IEEE Trans Pattern Anal Mach Intell 10(5):695–703

Yan X, Han J (2002) gSpan:Graph-based substructure pattern mining. In: Proc IEEE international conference on data mining