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. The problem finds applications in many vision (e.g., object recognition) and learning problems where graphs are increasingly being adopted as a representation tool. Existing techniques for this problem are evolutionary search based; in this paper, we propose a polynomial time algorithm based on a linear programming formulation. We present an additional bi-level method to obtain solutions arbitrarily close to the optimal in non-polynomial time (in worst case). Within this new framework, one can optimize edit distance functions that capture similarity by considering vertex labels as well as the graph structure simultaneously. In context of our motivating application, we discuss experiments on molecular image analysis problems - the methods will provide the basis for building a topological map of all pairs of the human chromosome.

1. Introduction

Graphs are a natural choice in applications 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. Recent literature in vision includes several examples of graph theoretic algorithms for some classical problems like segmentation, image denoising and stereo using ideas such as spectral methods, graph cuts 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 propagate into 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 chromosone 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 [5, 14]. 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 effectively reduces to finding a median graph for a set of attributed graphs from individual nuclear images. In addition, the general-
ized median graph problem is also a natural formalization for many problems arising in computer vision and structural pattern recognition as well as specific graph learning problems arising in drug design [6].

1.1. Problem Description

Let a labeled undirected graph $G$ be given as $G = (V, E, f_v, f_e)$, where

- $V$ is the vertex set, $E$ is the edge set
- $L_V$ is the set of node labels, $L_E$ is the set of edge labels,
- $f_v : V \rightarrow L_V$ is a mapping from nodes to labels or weights, and
- $f_e : E \rightarrow L_E$ is a mapping from edges to labels or weights.

Let $G = (G_1, G_2, \ldots, G_n)$ be a collection of graphs in arbitrary orientation with these properties

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

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

$$\bar{G} = \arg \min \sum_{i=1}^{n} d(\bar{G}, G_i) \quad G_i \in G,$$

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 $\bar{G} \in G$, the median graph is the set median; if we waive this requirement, we get the generalized median graph problem.

1.2. Previous works

Generalized Median graphs were recently introduced by Jiang, M"unger, and Bunke [10]. Their solution involved a genetic search based algorithm; however, the work was significant because it provided a link between problems of model construction (given a set) and median graphs. A subsequent paper by Hlaoui and Wang [8] relied on certain application-specific hypotheses where local choices that improved the objective function were iteratively picked. The reader may notice that despite the fact that the median graph problem allows a precise combinatorial graph theoretic definition, both existing techniques are soft computing based or heuristic approaches; no combinatorial solution paradigms are known (except the special case of certain types of trees [15]).

While the problem of generalized median graphs is still somewhat young, a strongly related problem, called the graph isomorphism problem, has been extensively studied by the computer vision and theoretical computer science communities. The earliest papers on this topic are due to Corneil and Gotlieb [4] and Ullmann [17]. The status of the problem is interesting – no polynomial time algorithms are known, at the same time, a formal proof of NP-hardness is also unknown. However, a number of approaches exist for solving various special cases of this problem. We will avoid an exhaustive discussion (see [16] for details) but will focus only on a subset of algorithms that are relevant for putting the remainder of this paper in context.

In [18], Umebayama 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. The algorithm by Almohamad and Duffua in [2] also employs adjacency matrices for weighted graph matching optimization. It uses a Linear Program (LP, for short) formulation that nicely exploits the relationship between permutation matrices and graph isomorphism as follows.

$$\min \|A_{G_0} - P A_{G_1} P^T\|,$$

where $A_{G_0}$ and $A_{G_1}$ denote the adjacency matrices of the two edge weighted graphs and $P$ denotes a permutation matrix applied to one of the graphs. Hence, the graph matching problem reduces to finding a $P$ that minimizes the difference of one graph (say, $A_{G_0}$) with the permuted version of the other (say, $P(A_{G_1})$).

The algorithm in [2] cannot be directly applied to the isomorphism problem on general graphs with unequal number of vertices or vertex labeled graphs. 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 insertion and deletion of nodes ($f_v(u)$), edges ($f_e(e)$) and substitution of nodes ($d_v(f_v(u_1), f_v(u_2))$) and edges ($d_e(f_e(e_1), f_e(e_2))$). 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 (Hamming distance) and edge replacement costs are in $L_1$ or $L_2$ space, the matching will favor mapping $\Delta abc$
in $G_0$ to $\triangle def$ in $G_1$ instead of $\triangle abc$. Clearly, the matching is a trade-off between competing influences.

![Figure 1: Matching with labels and weighted edges.](image)

For the special case of graphs with labeled vertices and unweighted edges, Justice and Hero [11] very recently proposed a modification of the algorithm in [2] 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, $G_{Ω} = (V_{Ω}, E_{Ω})$ where $|V_{Ω}| = N ≤ |V_1| + |V_2|$. 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. This amounts to permuting the standard placement of $G_0$ to minimize the following.

$$
\min \sum_{i=1}^{N} \sum_{j=1}^{N} d(f_v(A_0^i), f_v(A_1^j))P^{ij} + \frac{1}{2}||A_0 - PA_1P^T||, \tag{3}
$$

where $A_0$ and $A_1$ 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 $i$th (and $j$th) vertex of $A_0$ (and $A_1$) and $d \in \{0, 1\}$ is the cost function for the vertices.

While vertex and edge edit costs are in the same space (binary), (3) does not adequately capture the notion of similarity distance. To see this, consider two to-be-matched input graphs with vertices having large degree ($≥ 3$). If vertices with mismatched labels from the two graphs are aligned, one naturally pays a cost of 1 for a single vertex pair mismatch. This is clearly small compared to the cost incurred when vertices having a large difference in degree are matched to each other (see (3)). 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, one may construct examples where such an approach may ignore labels almost completely in favor of same-degree-vertex alignments, especially if the vertices have high degree. While it could be argued that 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, in matching graphs that represent chemical compounds (e.g., Lewis structure), should we match a ‘C’ (carbon) to a ‘H’ (hydrogen) simply because they share bonds with the same number (degree) of atoms?

The analysis above indicates that using edit distance as a cost function in many applications yields a biased weighted matching where a degree mismatch has a higher penalty. Therefore, we must somehow re-weight the cost of replacing vertices to reflect the vertex labels as well as the associated edge information (structure) concurrently. We will discuss these issues next.

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. 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 §1.2 while preserving the intuitions discussed above, we consider a generalization of edit distances for matching graphs. We will focus on matching edges alone; fortunately, since edges are identified by their end vertices, it is possible to align vertices implicitly while aligning edges explicitly as we illustrate now. Assume we are given two to-be-matched vertex labeled graphs, $G_0 = (V_0, E_0, f_v, f_e)$ and $G_1 = (V_1, E_1, f_v, f_e)$. We are also given a matrix, $Φ$, indicating similarity between vertex labels. The entry, $Φ(i, j)$ indicates the extent of dissimilarity between labels (colors) $i$ and $j$.

$$Φ(i, j) = 0 \text{ if } i = j. \text{ The presence or absence of a link between vertices is indicated by their edge weights, } f_e \text{ where } f_e \in \{0, 1\}. \text{ An explicit match between edge } e_i = (a, b) \in G_0 \text{ and } e_j = (c, d) \in G_1 \text{ aligns the two vertices of the edges i.e., } a \rightarrow c, b \rightarrow d. \text{ Therefore, while}$$
we do not focus on matching vertices, we can indirectly ‘charge’ vertices with misaligned labels. This idea allows us to naturally derive the cost that must be paid when an edge \(e_i = (a, b)\) (in source) is matched with another edge \(e_j = (c, d)\) (in target) given as

\[
\text{cost}(e_i, e_j) = \mathcal{J}(a, c) + \mathcal{J}(b, d) + ||f_\varepsilon(e_i) - f_\varepsilon(e_j)||,
\]

where the \(\text{cost}(e_i, e_j) = 0\) if both end vertices of the edge pair match perfectly. Otherwise, it reflects sum of the costs of (i) misaligned vertices (based on the dissimilarity of aligned vertex labels) and (ii) difference of weights of the edges (which is 0 unless it is matched to a null edge).

While (4) 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 \(\mathcal{J}(u_1, u_2) > 0\). Each edge incident on \(u_1\) pays a cost of \(\mathcal{J}(u_1, u_2)\) due to a misaligned \(u_1\). Thus, the total cost is \(\text{deg}(u_1) \cdot \mathcal{J}(u_1, u_2)\) where \(\text{deg}(\cdot)\) denotes the degree of a vertex. Similarly, the cost for all incident edges of \(u_2\) is \(\text{deg}(u_2) \cdot \mathcal{J}(u_1, u_2)\). The total cost of one vertex misalignment pair \((u_1 \leftrightarrow u_2)\) is then

\[
D(u_1, u_2) = [\text{deg}(u_1) + \text{deg}(u_2)] \cdot \mathcal{J}(u_1, u_2).
\]

Notice that all edges associated with \(u_1\) and \(u_2\) have paid only part of their cost (due to vertex misalignment on one side and reflecting the first term in (4)). However, the second vertex of edge \(e_i\) (see second term in (4)) is automatically considered at the time of evaluation of the other pair of aligned vertices. The cost function that models this is

\[
\min \sum_{i=1}^{N} \sum_{j=1}^{N} D(i, j) P^{ij} + \frac{1}{2} \|A_0 - PA_1 P^T\|. \tag{6}
\]

The second term in (6) evaluates the cost of matching a ‘real’ edge to a ‘non-existent’ or null edge (see third term in (4)). But, (6) as a measure of graph similarity is still not entirely accurate. It is perfect when an edge matches with a null edge; however, when an edge matches to another real edge, the cost due to \(\mathcal{J}(u_1, u_2)\) 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.

For convenience of presentation, let us first generalize the notion of graph similarity distance (in (6)) in the context of median graphs. Addressing the overestimation issues discussed above in the generalized setup turns out to be significantly easier. Notice that when we have just two graphs, the cost calculated is the similarity of one graph to the permuted version of the other. In our formulation, all the input graphs are embedded (placed) in an edit grid, \(G_{\Omega}\), \(|G_{\Omega}| \geq N\). Here, a permutation is simply a change of placement from one position to the other. Hence, the generalized median is a graph on the edit grid that minimizes the distance of every graph in the input set (post-permutation) to itself. This distance is clearly zero when all input graphs are isomorphic. 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 (6). Throughout this paper, upper case letters (\(A\)) and upper case letters with a single subscript (\(A_0\)) will denote matrices; upper case letters with two subscripts or two superscripts will refer to individual matrix entries (\(A_{ij}, A_{ij}^T\), and \(A(i, j)\)).

### 2.2. Cost when both graphs are permuted

The edit grid, \(G_{\Omega}\), provides a common ‘reference’ frame in which the 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 \(G_{\Omega}\) and then extend the definition for multiple graphs. If graphs, \(G_0\) and \(G_1\), are permuted by \(P_0\) and \(P_1\) respectively, (6) is represented as follows.

\[
\sum_{i=1}^{N} \sum_{k=1}^{N} D(i, j) P_0^{ik} P_1^{kj} + \frac{1}{2} \|P_0 A_0 P_0^T - P_1^T A_1 P_1\|. \tag{7}
\]

where \(P_0^{ik} P_1^{kj} = 1\) 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 (7) involves the product term of two variable matrices, \(P_0\) and \(P_1\). Our linearization involves an additional variable \(X \in \mathbb{R}^{N \times N \times N}\), \(X_{ijk}\) is 1 if \(P_0^{ik} P_1^{kj} = 1\) and 0 otherwise. This naturally yields

\[
P_0^{ik} + P_1^{kj} = 2 \implies X_{ijk} = 1. \tag{8}
\]

(8) can be converted to regular linear constraints as

\[
P_0^{ik} + P_1^{kj} \geq 2X_{ijk},
\]

\[
P_0^{ik} + P_1^{kj} - 1 \leq X_{ijk}. \tag{9}
\]

Observe that when \(P_0^{ik} + P_1^{kj} = 2\), \(X_{ijk}\) must be 1 to simultaneously satisfy both constraints in (9); if
\( P^k_0 + P^i_j \leq 1 \), \( X_{ijk} \) must be 0. (8) and (9) are hence equivalent. Incorporating these into (7) gives

\[
\sum_{i=1}^{N} \sum_{j=1}^{N} D(i, j)\left( \sum_{k=1}^{N} X_{ijk} \right) + \frac{1}{2} \| P_0 A_0 P_0^T - P^T_1 A_1 P_1 \|. \tag{10}
\]

We now revisit the overestimation issues from §2.1. To calculate the overestimation in (10), we first need to determine the edge-to-edge mappings between \( G_0 \) and \( G_1 \). This is given by the \( "n" \) entries in the dot product of \( P_0 A_0 P_0^T \) and \( P^T_1 A_1 P_1 \). The overestimation, say \( C \in \mathbb{R}_n \times n \), is associated with \( (i, j) \) positions with 1s and must be deduced. For presentation purposes, let us define the following notations

\[
\gamma = (P_0 A_0 P_0^T + P^T_1 A_1 P_1), \quad \tilde{\gamma} = (P_0 A_0 P_0^T - P^T_1 A_1 P_1), \tag{11}
\]

\[
\phi(\mathcal{X}, X)_i = \sum_{i_1=1}^{N} \sum_{j_1=1}^{N} \mathcal{X}(i_1, j_1) X_{i_1 j_1 i}, \tag{12}
\]

Then,

\[
C_{ij} = \begin{cases} \phi(\mathcal{X}, X)_i + \phi(\mathcal{X}, X)_j - C_{ij} \leq (2 - \gamma_{ij}) M, \\ C \leq MP_0 A_0 P_0^T, \\ C \leq MP^T_1 A_1 P_1. \end{cases} \tag{13}
\]

Notice that for an edge, \( e_{ij} \), when the sum in (13) is 2, i.e., when the dot product of \( P_0 A_0 P_0^T \) and \( P^T_1 A_1 P_1 \) 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 (4)). The conditional constraints in (13) can then be easily expressed as

\[
\phi(\mathcal{X}, X)_i + \phi(\mathcal{X}, X)_j - C_{ij} \leq (2 - \gamma_{ij}) M,
\]

\[
C \leq MP_0 A_0 P_0^T, \quad C \leq MP^T_1 A_1 P_1. \tag{14}
\]

Here, the constraints (denoted without subscripts) apply for all matrix elements. Therefore, the final objective function that takes care of the overestimation is

\[
\min \sum_{i=1}^{N} \sum_{j=1}^{N} D(i, j)\left( \sum_{k=1}^{N} X_{ijk} \right) + \frac{1}{2} ||\tilde{\gamma}|| - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} C_{ij}, \tag{15}
\]

where \( \tilde{\gamma} \) is as defined in (11) and the constraints are

\[
P_0 e = e; \quad P^T_0 e = e; \quad P_1 e = e; \quad P^T_1 e = e, \tag{16}
\]

\[
P^k_0 + P^i_j \geq 2X_{ijk}, \quad P^k_0 + P^i_j - 1 \leq X_{ijk}, \tag{17}
\]

\[
\phi(\mathcal{X}, X)_i + \phi(\mathcal{X}, X)_j - C_{ij} \leq (2 - \gamma_{ij}) M, \tag{18}
\]

\[
C \leq MP_0 A_0 P_0^T, \quad C \leq MP^T_1 A_1 P_1, \tag{19}
\]

\[
P_0, P_1, X \in \{0, 1\}; \quad C \in [0, M]. \tag{20}
\]

**Remark:** When only one graph is permuted, the simpler model only has \( O(n^2) \) variables and constraints.

### 3. Linearization and Relaxation

In this section, we describe a relaxation technique to derive a polynomial time solvable version of the problem in §2.2. We illustrate the ideas using the familiar objective function in (2) in §1.2. The strategy applies readily to (15)-(20). We observe that the objective, \( \sum (\gamma - P A G_j P^T)_{ij} \), is equivalent to

\[
\sum_i (\vec{\text{vec}}(\gamma_1) - \vec{\text{vec}}(\gamma_2)(P \otimes P))_i, \tag{21}
\]

where \( \otimes \) indicates the Kronecker product and \( \text{vec}(\cdot) \) vectorizes a matrix by stacking its columns. Consider the LP

\[
\begin{align*}
\min & \quad \sum_{i=1}^{N} \left( S + T \right) \\
\text{s.t.} & \quad \vec{\text{vec}}(\gamma_1) - \vec{\text{vec}}(\gamma_2)Q + S - T = 0 \\
& \quad \sum_{i=1}^{N} Q(k + (i - 1)n, (j - 1)n + l) = P^{ij}, \forall i, j, k \\
& \quad \sum_{i=1}^{N} Q_{ij} = P \sum_{j=1}^{N} Q_{ij} = P, \forall i, j \in [1, n], \\
& \quad Pe = e; e^T P = e^T; P, Q, S, T \geq 0. \tag{24}
\end{align*}
\]

The relaxation of (23)-(24) is polynomial time solvable.

### 4. Generalized Median Graphs

A natural extension of (15)-(20) 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 \( d_p(G_0, G_1, P_0, P_1) \). 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

\[
\min \sum_{x=1}^{K} \sum_{y=1}^{K} d_p(G_x, G_y, P_x, P_y), \tag{25}
\]

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 and obtaining the median graph as a mean of the permuted input variables.
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 §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 is $O(N^4 K)$, for large graphs some application specific heuristics may need to be employed. In the following sections, employing the ideas above we discuss a hybrid approach that yields a factor two approximation ratio if graph distances are known correctly. 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 [9] as a first step. Let the distance (discussed above) between two graphs be given as $d(G_0, G_1) = d_{pq}(G_0, G_1, I, P_1)$. We first introduce the following result.

**Lemma 1** For any three graphs $G_p$, $G_q$ and $G_r$, $d(G_p, G_q) \leq d(G_p, G_r) + d(G_q, G_r)$ i.e., $d(\cdot, \cdot)$ is metric.

Let $\mathcal{G} = (G_0, G_1, \ldots, G_K)$ be a collection of graphs with $|V_i| = V_i, j$. 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

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

We will avoid computing $G^*$ directly. Instead, we employ the lower bounding ideas in [9] 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.

$$\min \sum_{i=1}^{K} x_i \quad \text{(26)}$$

s.t. $x_k + d(G_k, G_i) \geq x_i, \forall k, l$, $x_k + d(G_k, G_i) \geq x_i, \forall k, l$, $x_k + x_i \geq d(G_k, G_i), \forall k, l$, $x_i \geq 0, \forall i \in [1, K]. \quad \text{(27)}$

(26)-(27) can be solved optimally in polynomial time.

**Property 1** (from [9]) The sum of the entries of $X = (x_1, \ldots, x_K)^T$ denotes a lower bound on the distance of all graphs to the optimal generalized median.

**Lemma 2** There must be a graph $G_a \in \mathcal{G}$ 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 $x^* = (x^*_1, \ldots, x^*_K)^T$ be an optimal solution for (26)-(27). Consider a graph $G_a \in \mathcal{G}$ s.t.

$$a = \{ j : x^*_i \leq x^*_j, \forall i \in [1, K] \}.$$  

From (27), we know $\forall G_i, d(G_i, G_a) \leq x^*_i + x^*_a$. Since $x_0 \leq x^*_i, d(G_i, G_a) \leq 2x^*_i$. Therefore $\sum_{i=1}^{K} d(G_i, G_a) \leq 2 \sum_{i=1}^{K} x^*_i$. Applying Property 1, the lemma follows. \Box

**Theorem 1** The graph $G_a \in \mathcal{G}, a = \{ j : x^*_i \leq x^*_j, \forall i \in [1, K] \}$, 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 (26)-(27) and (15)-(20) and outputting the better solution.

### 4.1. Alternate bi-level algorithm (A sketch)

Let $U = \sum_{i=1}^{K} d(G_k, G_{i})$. We know that the optimal solution lies in $\left[ \sum_{i=1}^{K} x^*_i, U \right]$. We adopt a binary search in $\left[ \sum_{i=1}^{K} x^*_i, U \right]$ starting at a value $c = \sum_{i=1}^{K} x^*_i$ 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 $G$ is upper bounded by $c$. Here the adjacency matrices of $G$ and $P_i$ are variables. The constraints are defined by (15)-(20). If this returns a feasible solution, we are done (i.e., $G = G^*$). Otherwise, we continue the binary search. The FP can be solved using branch-and-bound type methods (available in most commercial solvers like CPLEX) but may have non-polynomial running time in the worst case.

### 5. Experimental Results

We discuss experiments related to our motivating biological image analysis application. Results on pattern recognition graph database and applications in drug-design will appear in the longer version of the paper.

#### 5.1. Applications in Biological Image Analysis

##### 5.1.1 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’ [5]. It shows a statistical correlation with gene density and the size of chromosomes [7]. In humans, larger chromosomes (numbers 1 – 10) are arranged on the periphery. Secondly, chromosomes may have non-random neighbors, a finding that might account for preferential translocations and interactions between specific chromosomes. Biologists suspect that such patterns are related to genomic evolution and cell-type-specific variability.
In this regard, several researchers [3, 13] 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. 2, 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 [3].

5.1.2 Chromosomal organization maps

Topological map derivation of chromosomes transforms naturally to the generalized median graph problem. Each individual chromosome (denoted as numbers in Fig. 2) 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. 3).

To evaluate the goodness of our final ‘model’, we divide the cells into a training set (19 cells) and a test set (18 cells). We report on the following observations. Optimality. Figure 3 (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 (26)-(27)). 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. Note: Exhaustive enumeration analysis suggests that a solution much closer to the lower bound is unlikely.

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 [1] and other software [12]. We then compared these to the results obtained by the median graph model for this cell line. Figure 3(h) shows groups (colored edges) with an occurrence frequency of ≥70%. Fig. 3(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. Notice that while the median can fit all subgraphs, all subgraphs cannot build the model.

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. 3(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 new cell, we can predict close to 50% of its chromosome organization. This means ∼50% of chromosomal relationships are preserved!

6. Conclusions

This paper applies mathematical programming to design an optimization framework for the generalized median graph problem in context of a real world vision problems in biological image analysis. In addition, the concept has applications to drug-design (structural pattern recognition) and other problems, we believe that the ideas will find general applicability in building exemplars from representations such as attributed skeletal graphs. The extension of the similarity metric is also of independent interest.

References

Figure 3: 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 ≥ 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).


