

# Improved Orientations of Physical Networks\*

Iftah Gamzu<sup>†</sup>

Danny Segev<sup>‡</sup>

Roded Sharan<sup>§</sup>

## Abstract

The orientation of physical networks is a prime task in deciphering the signaling-regulatory circuitry of the cell. One manifestation of this computational task is as a maximum graph orientation problem, where given an undirected graph with  $n$  vertices and a collection of vertex pairs, the goal is to orient the edges of the graph so that a maximum number of pairs are connected by a directed path. We develop an approximation algorithm for this problem with a performance guarantee of  $O(\log n / \log \log n)$ . This result improves on a previous logarithmic approximation, due to Medvedovsky et al. (WABI '08). In addition, motivated by interactions whose direction is pre-set such as protein-DNA interactions, we extend our algorithm to handle mixed graphs, a major open problem posed by earlier work. In this setting, we demonstrate that a polylogarithmic approximation ratio is achievable under biologically-motivated assumptions on the sought paths.

---

\*An extended abstract of this paper appeared in *Proceedings of the 10th International Workshop on Algorithms in Bioinformatics*, pages 215–225, 2010.

<sup>†</sup>Blavatnik School of Computer Science, Tel-Aviv University, Tel-Aviv 69978, Israel. Email: [iftgam@tau.ac.il](mailto:iftgam@tau.ac.il). Supported by the Israel Science Foundation, by the European Commission under the Integrated Project QAP funded by the IST directorate as Contract Number 015848, by a European Research Council (ERC) Starting Grant, and by the Wolfson Family Charitable Trust.

<sup>‡</sup>Department of Statistics, University of Haifa, Haifa 31905, Israel. Email: [segevd@stat.haifa.ac.il](mailto:segevd@stat.haifa.ac.il).

<sup>§</sup>Blavatnik School of Computer Science, Tel-Aviv University, Tel-Aviv 69978, Israel. Email: [roded@tau.ac.il](mailto:roded@tau.ac.il). Supported by a research grant from the Israel Science Foundation (grant no. 385/06).

## 1 Introduction

A fundamental problem in the study of biological networks concerns with inference of causal relations that are often not covered by current experimental techniques. One prime example for such deficiency is protein-protein interaction (PPI) networks. While PPIs have been measured at large scale across tens of organisms for over a decade now, current technologies do not provide information on the direction in which signal flows. Such information may be indirectly obtained from perturbation experiments in which a gene is perturbed and as a result other genes change their expression levels. Employing the assumption that the expression changes imply directed pathways from the perturbed, or causal, gene to the affected genes, several researchers have successfully inferred interaction directions.

The inference of interaction directions that best fit the perturbation experiments can be cast as a *maximum graph orientation* problem. An instance of this problem consists of an undirected graph  $G = (V, E)$  with  $n$  vertices, representing the PPI network. An additional ingredient of the input is a collection  $C$  of requests, each of which is characterized by an ordered pair of source-target vertices, representing a causal gene and an affected gene. The goal is to orient the graph in a way that maximizes the number of satisfied requests. An orientation of a graph is an assignment of a single direction to each edge. A request  $i$  with a source-target pair  $(s_i, t_i)$  is said to be satisfied if there is a directed path from  $s_i$  to  $t_i$  in the oriented graph. We note that any instance of the maximum graph orientation problem can be reduced to one where the underlying graph is a tree. Indeed, if the input graph contains cycles, one can sequentially contract them, one after the other. In each step, the edges of an arbitrary cycle are all oriented in the same direction (either clockwise or counter-clockwise). As a result, every pair of vertices on this cycle admit a directed path between them, and thus, the cycle can be contracted into a single vertex.

**Previous work.** Medvedovsky et al. [15] seem to have been the first to study the maximum graph orientation problem. They demonstrated that the seemingly simple setting when the underlying graph is a star is equivalent to the *maximum directed cut* problem. The latter problem admits a semidefinite programming based 0.874-approximation algorithm [4, 13], while approximating it within factors of  $11/12 \approx 0.916$  and  $\alpha_{\text{GW}} \approx 0.878$  is NP-hard [10] and Unique Games-hard [12], respectively. These hardness bounds clearly follow to our problem. On the positive side, Medvedovsky et al. [15] devised an  $O(\log n)$  approximation algorithm for arbitrary trees, and proposed an exact dynamic-programming algorithm for the special case of path graphs.

Further research along these lines focused on variants of the maximum graph orientation problem. For instance, Hakimi, Schmeichel, and Young [8] studied the special setting in which the set of requests contains all vertex pairs, and developed an exact polynomial time algorithm. Arkin and Hassin [2] established hardness results for the problem of deciding whether one can orient a *mixed* graph, i.e., a graph in which the orientation of some edges is predetermined, to satisfy a given set of requests.

**Our results.** The contribution of this paper is two-fold: (i) We propose a deterministic algorithm for the maximum graph orientation problem whose approximation ratio is  $O(\log n / \log \log n)$ . This result improves on a previous logarithmic approximation, due to Medvedovsky et al. [15]. An interesting feature of our algorithm is that it optimizes with respect to some fixed optimal solution, whereas previous result uses the entire collection of requests as a reference point. Notably, any algorithm that optimizes with respect to the entire set of requests cannot achieve  $o(\log n)$ -approximation as there are certain instances in which any orientation cannot satisfy more than a logarithmic fraction of requests [14]. (ii) We devise an approximation algorithm for the generalized scenario of mixed graphs. This scenario is motivated by the need to include protein-DNA interactions (PDIs), which are both fundamental to signal transduction and key mediators in the observed expression changes, in the network. The performance guarantee of our algorithm depends on the number of PDI segments in the underlying pathways. For typical cases, in which

the pathways contain at most one segment, our algorithm has an approximation ratio of  $O(\log n)$ .

The rest of the paper is organized as follows: In Section 2, we describe and analyze the algorithm for the maximum graph orientation problem, while in Section 3, we study the mixed-graphs scenario.

## 2 Orienting Undirected Graphs

In this section, we devise a deterministic algorithm that achieves an approximation guarantee of  $O(\log n / \log \log n)$  for the maximum graph orientation problem. Our algorithm employs the classify-and-select paradigm. More specifically, it exploits various structural properties to partition the collection of requests into  $O(\log n / \log \log n)$  pairwise-disjoint classes. For each such class, given the additional structure imposed, we separately compute a graph orientation which satisfies a constant fraction of the optimal number of satisfiable requests from this class. In particular, each class is treated in a completely independent fashion, as if there are no other classes under consideration. Consequently, the above-mentioned approximation ratio follows by picking, out of the set of all the computed graph orientations, the one that satisfies a maximum number of requests.

### 2.1 The classification process

In what follows, we specify the process by which requests are partitioned into classes. To this end, we begin by presenting the notion of an almost-balanced decomposition, which can be viewed as a generalization of the well-known centroid decomposition [5]. Note that structural properties in this spirit have been explored and exploited in various settings (see, e.g., [3, 7, 9]).

**Definition 2.1.** Let  $T = (V, E)$  be a tree. An *almost balanced  $k$ -decomposition* of  $T$  is a partition of  $T$  into  $k$  edge-disjoint subtrees  $T_1, \dots, T_k$  such that each subtree contains between  $|E|/(3k)$  and  $3|E|/k$  edges.

**Lemma 2.2.** ([6]) *Let  $T = (V, E)$  be a tree with  $|E| \geq k$ . An almost balanced  $k$ -decomposition of  $T$  exists and can be found in polynomial time. In addition, the number of vertices that are shared by at least two subtrees is less than  $k$ .*

The classification process corresponds to a recursive decomposition of the input tree  $T$ . Let  $\mathcal{T}_1 = \{T_1, \dots, T_k\}$  be an almost balanced  $k$ -decomposition of  $T$  into  $k$  edge-disjoint subtrees. The first class of requests,  $C_1$ , consists of all requests separated by  $\mathcal{T}_1$ , namely, requests  $i$  whose endpoints  $s_i, t_i$  reside in different subtrees of the decomposition  $\mathcal{T}_1$ . Figure 1 provides a concrete example of requests separated by an almost balanced decomposition. Now, to classify the remaining set of requests,  $C \setminus C_1$ , we recursively apply the previously-described procedure with respect to the collection of subtrees in  $\mathcal{T}_1$ . Specifically, in the second level of the recursion, an almost balanced  $k$ -decomposition is computed in each of the subtrees  $T_1, \dots, T_k$ , to obtain a set  $\mathcal{T}_2$ , comprising of  $k^2$  subtrees. The second class of requests,  $C_2$ , consists of all yet-unclassified requests separated by  $\mathcal{T}_2$ . In other words, the endpoints of each request  $i \in C_2$  reside in different subtrees of  $\mathcal{T}_2$ , but in the same subtree of  $\mathcal{T}_1$ . The remaining classes  $C_3, C_4, \dots$  are defined in a similar manner. It is important to note that the recursive process ends as soon as we arrive at a subtree with strictly less than  $k$  edges. In this case, we make use of the trivial decomposition, where the given subtree is broken into its individual edges.

It is quite obvious that, up until this point in time,  $k$  was treated as a parameter whose value has not been determined yet. For our purposes, we employ the above-mentioned classification process with  $k = \lceil \log n \rceil$ . One can easily verify that the overall number of levels in the recursion, or equivalently, the number of request classes is  $O(\log_k n) = O(\log n / \log \log n)$ . This claim is immediately implied by observing that the maximum size of a subtree in level  $\ell$  of the recursion

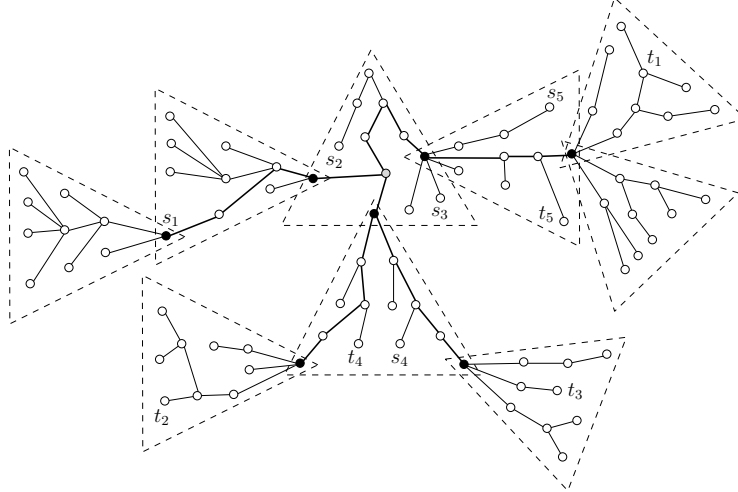


Figure 1: An almost balanced 9-decomposition. Here, requests 1, 2, and 3 are separated, whereas requests 4 and 5 are not.

is at most  $(3/k)^\ell \cdot |E|$ . As a side note, we remark that the balanced decomposition property, ensuring that all subtrees are of size roughly  $|E|/k$ , does not play any role from this point on, as its sole purpose was to restrict the number of classes to  $O(\log n / \log \log n)$ .

## 2.2 An orientation algorithm for a single decomposition

Notice that a class of requests, say  $C_\ell$ , generally consists of several subsets of requests, each created when different subtrees in  $\mathcal{T}_{\ell-1}$  are partitioned by the decomposition  $\mathcal{T}_\ell$ . More specifically, assuming that the subtrees in  $\mathcal{T}_{\ell-1}$  are  $T_1, T_2, \dots$ , the class  $C_\ell$  can be written as the disjoint union of  $C_\ell^1, C_\ell^2, \dots$ , where  $C_\ell^j$  is the set of requests that are first separated when  $T_j$  is partitioned. Recall that the path of any request separated by some subtree decomposition must be contained in that subtree (otherwise, this request would have been separated in previous recursion steps). This observation implies that it is sufficient to compute an orientation for a single subtree decomposition and its induced set of separated requests. Given a polynomial-time algorithm that computes such an orientation, one can *sequentially* apply it to each of the subtree decompositions in the same recursion level. The resulting orientations (in edge-disjoint subtrees) can then be “glued” to form a single orientation, defined for the entire edge set, satisfying at least as many requests as the overall number of requests satisfied in all individual subtrees.

In what follows, we focus our attention on a single decomposition, and devise a randomized algorithm that computes an orientation which satisfies, in expectation, a constant fraction of the optimal number of satisfiable requests for this decomposition. Later on, we will argue that this algorithm can be easily derandomized. Formally, an instance of the problem in question consists of a tree  $T = (V, E)$ , and a partition  $\mathcal{T} = \{T_1, \dots, T_k\}$  of this tree into  $k$  edge-disjoint subtrees, where  $k \leq \lceil \log n \rceil$ , and the number of vertices shared by at least two subtrees is less than  $k$ . In addition, we are given a collection  $\mathcal{C}$  of requests, where each request path is separated by  $\mathcal{T}$ , meaning that  $s_i$  and  $t_i$  reside in different subtrees of the decomposition  $\mathcal{T}$ .

**Notation and terminology.** Prior to describing the finer details of our algorithm, we introduce some notation and terminology. To better understand the suggested notation, we refer the reader to a concrete example in Figure 2.

- Let  $\text{OPT}$  denote the number of satisfied requests in some fixed optimal orientation of  $T$ .
- Let  $V_B \subseteq V$  be the set of *border vertices* of  $\mathcal{T}$ , that is, the set of vertices that are shared by

at least two subtrees in  $\mathcal{T}$ . Moreover, let  $S \subseteq T$  be the *skeleton* of  $\mathcal{T}$ , namely, the minimal subtree spanned by all border vertices. Note that this subtree consists of the union of paths connecting any two vertices in  $V_B$ . Finally, let  $V_J \subseteq V$  the set of *junction vertices*, defined as non-border skeleton vertices with degree at least 3 (counting only skeleton edges).

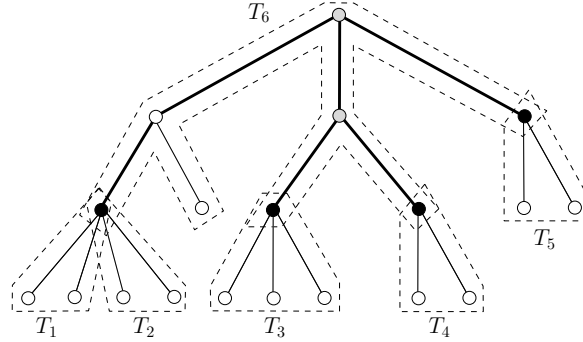


Figure 2: An almost balanced 6-decomposition. Note that black vertices are border vertices, gray vertices are junction vertices, and bold edges make up the skeleton of the decomposition.

**The algorithm.** We are now ready to present the specifics of our orientation algorithm. Our algorithm consists of two phases: *segment guessing*, where the optimal direction state of disjoint subpaths of the skeleton is attained, followed by *randomized assignment*, in which individual edges are assigned a direction.

PHASE I: SEGMENT GUESSING. Let us name the vertex set  $V_B \cup V_J$  the *core* of the skeleton  $S$ . One can easily verify that  $|V_B \cup V_J| < 2k$  as  $|V_J| < |V_B| < k$ , by Lemma 2.2. We now partition the skeleton into a collection  $\Sigma(S)$  of edge-disjoint paths, which are referred to as *segments*. Each such segment is a subpath of  $S$  whose endpoints are core vertices, but its interior traverses only non-core vertices. Clearly,  $|\Sigma(S)| = |V_B \cup V_J| - 1 < 2k$ . We now argue that one could obtain in polynomial time the direction state that the optimal orientation induces on each segment  $\sigma \in \Sigma(S)$ , *simultaneously* for all segments. To this end, notice that any skeleton segment  $\sigma = \langle v_1, v_2, \dots, v_\ell \rangle$  may be in one of three possible direction states:

1. *Right direction*: all edges are consistently directed from  $v_1$  towards  $v_\ell$ , i.e.,  $v_1 \rightarrow v_2, v_2 \rightarrow v_3, \dots, v_{\ell-1} \rightarrow v_\ell$ .
2. *Left direction*: all edges are consistently directed from  $v_\ell$  towards  $v_1$ , namely,  $v_1 \leftarrow v_2, v_2 \leftarrow v_3, \dots, v_{\ell-1} \leftarrow v_\ell$ .
3. *Mixed direction*: the direction of segment edges is non-consistent.

These definitions imply that the total number of segment direction states to be examined is of polynomial size since  $3^{|\Sigma(S)|} = 3^{O(k)} = 3^{O(\log n)} = n^{O(1)}$ . As a consequence, we may assume without loss of generality that the set of direction states induced by the optimal orientation on all the segments of  $\Sigma(S)$  is known in advance. This assumption can be easily enforced by enumerating over all  $n^{O(1)}$  possible segment direction states.

PHASE II: RANDOMIZED ASSIGNMENT. The goal of this phase is to orient the graph while making sure that the edge directions respect the outcome of the segment guessing phase. For this purpose, we begin by considering skeleton segments that have a consistent direction, namely, segments in either right or left direction states, and assign all the edges in these segments their implied direction. The assignment procedure proceeds with two randomized assignment steps:

1. Each segment in a mixed direction state is assigned, independently and uniformly at random, a right or left direction. All segment edges are oriented according to the chosen direction.
2. Each of the decomposition subtrees  $T_1, \dots, T_k$  is assigned, independently and uniformly at random, the role of a *sender* or a *receiver*. All the edges of each sender subtree are oriented towards the skeleton (in its simplest form, when the subtree contains a single border vertex, all edges are oriented toward that vertex). In contrast, all the edges of each receiver subtree are oriented away from the skeleton. We refer the reader to an example in Figure 3(a).

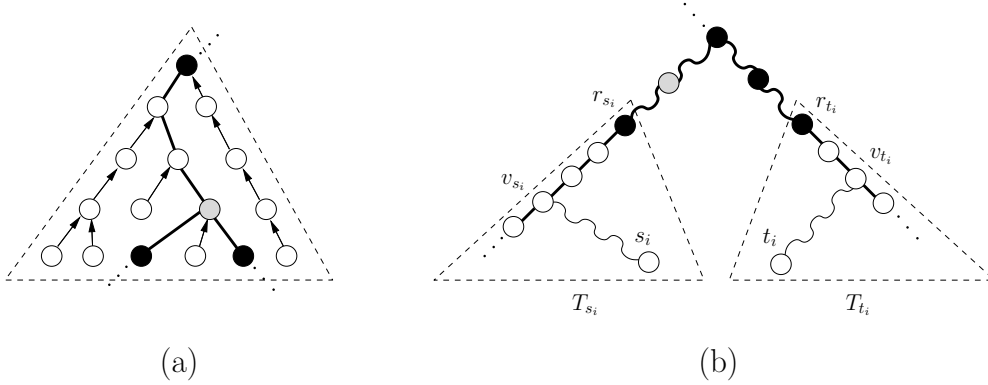


Figure 3: (a) An orientation of a sender subtree, where the bold edges are part of the skeleton. (b) A partition of a request path into five parts.

**Analysis.** We turn to prove that the expected number of satisfied requests is within a constant factor of optimal, as formally stated in the following theorem.

**Theorem 2.3.** *The resulting orientation satisfies at least  $\text{OPT}/16$  requests in expectation.*

**Proof.** Recall that we have previously assumed the endpoints of each request to reside in different subtrees of the decomposition  $\mathcal{T}$ . In particular, this implies that each request path must traverse at least one border (core) vertex. For this reason, as shown in Figure 3(b), we can divide each request path, with endpoints  $s_i$  and  $t_i$ , into five (some possibly empty) parts:

1. A subpath between  $s_i$  and its closest skeleton vertex  $v_{s_i}$ .
2. A subpath, along a partial skeleton segment, between  $v_{s_i}$  and its closest core vertex  $r_{s_i}$ .
3. A subpath between  $t_i$  and its closest skeleton vertex  $v_{t_i}$ .
4. A subpath, along a partial skeleton segment, between  $v_{t_i}$  and its closest core vertex  $r_{t_i}$ .
5. A subpath between  $r_{s_i}$  and  $r_{t_i}$ , along a sequence of complete skeleton segments.

With these definitions in mind, let us focus on some request  $i$  that is satisfied in the optimal orientation. We now argue that, with probability at least  $1/16$ , this request is satisfied in the random orientation constructed by the algorithm. Consequently, by linearity of expectation, the overall expected number of satisfied requests is  $\text{OPT}/16$ . A key observation one should make to establish this argument is that all the segments along the subpath between  $r_{s_i}$  and  $r_{t_i}$  must have a consistent direction in the optimal orientation; otherwise, this request would not have been satisfied. Accordingly, we may assume that our algorithm assigned the same direction to all the edges in these segments. Now, notice that the request under consideration is satisfied if

the following four probabilistic events occur: (1) the edges in the subpath between  $s_i$  and  $v_{s_i}$  are oriented towards  $v_{s_i}$ ; (2) the edges in the subpath between  $v_{s_i}$  and  $r_{s_i}$  are oriented towards  $r_{s_i}$ ; (3) the edges in the subpath between  $v_{t_i}$  and  $r_{t_i}$  are oriented towards  $v_{t_i}$ ; and (4) the edges in the subpath between  $t_i$  and  $v_{t_i}$  are oriented towards  $t_i$ . One can easily validate that these four events are independent, and that each one of them occurs with probability of at least  $1/2$ . For example, the edges in the subpath between  $s_i$  and  $v_{s_i}$  are oriented towards  $v_{s_i}$  if the underlying subtree  $T_{s_i}$  is selected as a sender. As a result, the probability that request  $i$  is satisfied in the random orientation is at least  $1/16$ . ■

**Derandomization.** The avid reader may have already noticed that the extent to which we utilize randomization is rather limited, and that its foremost purpose is to make the presentation of our algorithm simpler. Specifically, each segment in a mixed direction state is randomly assigned one of two possible directions, while each decomposition subtree is randomly assigned one of two possible roles. In other words, all we need to obtain a deterministic algorithm is a uniform sample space, with two possible values for  $O(\log n)$  random variables. This can be constructed in polynomial time either explicitly, as there are only  $n^{O(1)}$  possible outcomes, or in a more compact way, by observing that fourwise-independence is sufficient for the preceding analysis (see, for instance, [1, Chap. 15]).

**A semi-oblivious property.** In view of the derandomization procedure, we may reinterpret our algorithm as the following two-stage process: initially, we generate a set of polynomially-many potential orientations, determined by all possible outcomes of both the segment guessing and randomized assignment phases, and then, we select an orientation that maximizes the number of satisfied requests. Now, notice that the first stage of this process is semi-oblivious. Specifically, the set of generated orientations is independent of the collection of requests, and only builds on the structure of the underlying network. This property allows us to employ the algorithm in generalized requests settings. For instance, a natural generalization of maximum graph orientation is when each request is characterized by a collection of ordered source-target pairs, rather than a single pair. In this setting, a request is regarded as satisfied if at least one of its source-target pairs admits a directed path in the oriented graph. One can easily verify that our algorithm attains the same performance guarantees for this setting by applying nearly identical analysis.

One interesting scenario that is captured by the above-mentioned generalization is of maximum graph orientation *with groups*. In this scenario, a request  $i$  is satisfied if there is a directed path from some vertex  $s_i \in S_i$  to some vertex  $t_i \in T_i$  in the oriented graph. Here,  $S_i$  and  $T_i$  are vertex sets that characterize the request.

### 3 Orienting Mixed Graphs

Thus far, we have restricted our attention to undirected graphs. In practice, signaling pathways contain various types of interactions whose direction is specified in advance, most notably protein-DNA interactions. This implies that the input to the graph orientation problem is, in its utmost general setting, a mixed graph. A key difficulty in this setting is that, unlike the seemingly easier-to-handle scenario of unoriented edges, there is no trivial reduction to tree instances. What prevents us from contracting cycles is the possible existence of cycles with oriented edges pointing in opposite directions, for which there does not seem to be an easy way to decide in advance on the orientation of remaining edges.

Despite the inherent difficulty in a mixed graph input, the biological setting provides us with several constraints on the input graph, which we exploit in our approximation algorithm. The first constraint relates to the occurrence of PDI edges along pathways. Reviewing real pathways, we observed that signaling pathways do not jump back and forth between PPIs and PDIs; rather, in the vast majority of the cases there is a single switch from PPIs to PDIs. More precisely, let

a PDI *segment* in a linear path be a series of consecutive PDIs along the path that is flanked by PPI edges or by the start/end of the path. To gather statistics on the number of PDI segments in real pathways, we downloaded 116 human pathways from KEGG [11]. For each pathway, we counted the number of PDI segments in its longest linear path. Only 35 of the 116 pathways contained PDIs, and 18 of which had at least one PDI segment in their longest path. Notably, 17 of the 18 contained a single PDI segment; the remaining pathway contained two segments.

A second biologically-motivated constraint is a refinement of the first one: In two thirds of the 18 KEGG pathways with at least one PDI segment, the segment occurred at the end of the pathway. Interestingly, our algorithm can be directly applied to this latter scenario (of a single PDI segment that occurs at the end of the pathway) as each cause-effect pair can then be translated into a group request where the cause should connect to any of the genes that have a directed PDIs path to the effect. Below, we consider the general case and show that if the sought pathways contain at most  $\ell$  segments then an  $O(\log^\ell n)$  approximation is possible.

### 3.1 The algorithm

Let  $G = (V, E)$  be a mixed graph whose edge set can be described as  $E = E_{\mathcal{O}} \cup E_{\mathcal{U}}$ , where  $E_{\mathcal{O}}$  consists of edges with predefined directions, and  $E_{\mathcal{U}}$  are unoriented edges. Even though the input graph may contain cycles with oriented edges pointing in opposite directions, we can still contract unoriented cycles, and more generally, cycles where all oriented edges are consistently pointing in the same direction. Therefore, from this point on we assume that such cycles have already been contracted. With this setting in mind, an *unoriented component* (or,  $\mathcal{U}$ -component, for short) is defined as a maximal connected component of the unoriented subgraph  $(V, E_{\mathcal{U}})$ . It is worth noting that any  $\mathcal{U}$ -component is necessarily a tree, or otherwise, there must be unoriented cycles, which should have been contracted earlier on. Also note that there are no oriented edges with both head and tail residing in the same  $\mathcal{U}$ -component since any such edge induces a cycle that should have been contracted before. As a consequence of the preprocessing steps described above, the input graph can be represented as a directed acyclic graph on the  $\mathcal{U}$ -components, as illustrated in Figure 4(a). That is, the collection of  $\mathcal{U}$ -components can be topologically sorted such that all oriented edges between them are pointing from left to right.

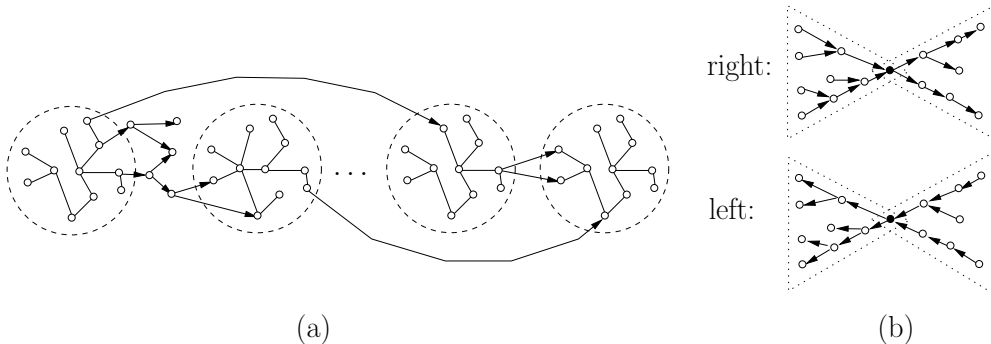


Figure 4: (a) A directed acyclic graph on  $\mathcal{U}$ -components. (b) The two possible orientations of a decomposed tree.

**Randomly connecting pairs in trees.** We temporarily deviate from the general theme of this section, and focus on one particular  $\mathcal{U}$ -component, designated by  $T = (V_T, E_T)$ . We proceed by devising an algorithm that computes a random orientation of  $E_T$  satisfying the following property: *any* pair of vertices in  $T$  is connected with probability  $\Omega(1/\log n)$ .

For this purpose, suppose we execute the classification process suggested in Section 2.1 where the collection of requests consists of all vertex pairs in  $T$ . However, rather than using almost-balanced  $k$ -decompositions with  $k = \lceil \log n \rceil$ , we will simplify the process by picking  $k = 2$ . Even

though the number of resulting request classes slightly blows up to  $O(\log n)$ , each time a tree is being decomposed, we obtain only two almost-balanced edge-disjoint subtrees which intersect in a common vertex and nothing more. On top of picking an alternative value of  $k$ , instead of testing all  $O(\log n)$  request classes as potential candidates for the class that separates the maximal number of pairs, we will pick one such class uniformly at random. Given this class, the orientation of each decomposed tree (two edge-disjoint subtrees) is determined, independently and uniformly at random, from one of the following two alternatives, as shown in Figure 4(b):

- *Right orientation*: all the edges of the first subtree are oriented towards the common root and all the edges of the second subtree are oriented away from that root.
- *Left orientation*: all the edges of the second subtree are oriented towards the common root and all the edges of the first subtree are oriented away from that root.

Following Section 2, it is not difficult to verify that any pair of vertices  $(s, t) \in V_T \times V_T$  is connected with probability  $\Omega(1/\log n)$  since the particular class that separates  $s$  and  $t$  is picked with probability  $\Omega(1/\log n)$ , and given that this class has been picked, there is a directed path from  $s$  to  $t$  with probability  $1/2$ .

We handle an arbitrary mixed graph  $G = (V, E_{\mathcal{O}} \cup E_{\mathcal{U}})$ , which has already been brought to the structural form of a directed acyclic graph on the collection of its  $\mathcal{U}$ -components, by independently running the randomized single-tree procedure in each  $\mathcal{U}$ -component. As a result, we obtain a random orientation of the graph, whose performance guarantee depends on the minimal number of  $\mathcal{U}$ -components that must be traversed in order to satisfy any request  $(s_i, t_i)$ . Specifically, suppose  $\ell_i$  stands for the minimal number of components that have to be traversed in an orientation that connects  $s_i$  to  $t_i$ . Then, we obtain the following theorem.

**Theorem 3.1.** *The algorithm constructs an orientation that satisfies  $\Omega(\text{OPT}/\log^\ell n)$  requests in expectation, where  $\ell = \max \ell_i$ .*

**Proof.** Let us focus on request  $i$ , and let  $P_i$  be a directed  $s_i$ - $t_i$  path that traverses  $\ell_i$  components in some orientation of  $G$ ; we denote these  $\mathcal{U}$ -components by  $\mathcal{U}^1, \dots, \mathcal{U}^{\ell_i}$ , indexed in topological order. Furthermore, let  $x_j$  and  $y_j$  be the entry vertex and exit vertex of  $P_i$  in  $\mathcal{U}^j$ , respectively. Notice that every  $(x_j, y_j)$  pair is connected with probability  $\Omega(1/\log n)$  since the randomized single-tree procedure is independently run in each  $\mathcal{U}$ -component. This implies that  $s_i$  and  $t_i$  are connected with probability  $\Omega(1/\log^{\ell_i} n)$ . Consequently, the expected number of satisfied requests is  $\Omega(\text{OPT}/\log^\ell n)$  by linearity of expectation. ■

## 4 Conclusions

We have designed a novel approximation algorithm for maximum graph orientation that achieves an  $O(\log n/\log \log n)$  ratio. We have further shown an extension of the algorithm that handles mixed graphs and provides a polylogarithmic approximation ratio under biologically-motivated assumptions. On the theoretical side, we believe that the techniques presented here are of independent interest, and may be applicable in other settings as well. On the practical side, the algorithmic extension to mixed graphs tackles a major open problem posed in [14], and is expected to yield much more realistic network orientations by integrating knowledge on PDIs into the orientation process.

**Acknowledgments:** The authors would like to thank Yael Silberberg for her help in gathering statistics on the KEGG pathways.

## References

- [1] N. Alon and J. H. Spencer. *The Probabilistic Method*. Wiley, New York, second edition, 2000.
- [2] E. M. Arkin and R. Hassin. A note on orientations of mixed graphs. *Discrete Applied Mathematics*, 116(3):271–278, 2002.
- [3] G. Even, N. Garg, J. Könemann, R. Ravi, and A. Sinha. Covering graphs using trees and stars. In *Proceedings 6th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems*, pages 24–35, 2003.
- [4] U. Feige and M. X. Goemans. Approximating the value of two prover proof systems, with applications to MAX 2SAT and MAX DICUT. In *Proceedings 3rd Israel Symposium on Theory and Computing Systems*, pages 182–189, 1995.
- [5] G. N. Frederickson and D. B. Johnson. Generating and searching sets induced by networks. In *Proceedings 7th International Colloquium on Automata, Languages and Programming*, pages 221–233, 1980.
- [6] I. Gamzu and D. Segev. A sublogarithmic approximation for highway and tollbooth pricing. In *Proceedings of the 37th International Colloquium on Automata, Languages and Programming*, 2010. To appear. Available online at: <http://arxiv.org/abs/1002.2084>.
- [7] O. Goldschmidt, D. S. Hochbaum, A. Levin, and E. V. Olinick. The SONET edge-partition problem. *Networks*, 41(1):13–23, 2003.
- [8] S. L. Hakimi, E. F. Schmeichel, and N. E. Young. Orienting graphs to optimize reachability. *Information Processing Letters*, 63(5):229–235, 1997.
- [9] R. Hassin and D. Segev. Robust subgraphs for trees and paths. *ACM Transactions on Algorithms*, 2(2):263–281, 2006.
- [10] J. Håstad. Some optimal inapproximability results. *Journal of the ACM*, 48(4):798–859, 2001.
- [11] M. Kanehisa, S. Goto, M. Furumichi, M. Tanabe, and Mika Hirakawa. KEGG for representation and analysis of molecular networks involving diseases and drugs. *Nucleic Acids Research*, 38(2):D355–D360, 2010.
- [12] S. Khot, G. Kindler, E. Mossel, and R. O’Donnell. Optimal inapproximability results for MAX-CUT and other 2-variable CSPs? *SIAM Journal on Computing*, 37(1):319–357, 2007.
- [13] M. Lewin, D. Livnat, and U. Zwick. Improved rounding techniques for the MAX 2-SAT and MAX DI-CUT problems. In *Proceedings 9th International Conference on Integer Programming and Combinatorial Optimization*, pages 67–82, 2002.
- [14] A. Medvedovsky. An algorithm for orienting graphs based on cause-effect pairs and its application to orienting protein networks. Master’s thesis, Tel-Aviv University, Israel, 2009.
- [15] A. Medvedovsky, V. Bafna, U. Zwick, and R. Sharan. An algorithm for orienting graphs based on cause-effect pairs and its application to orienting protein networks. In *Proceedings 8th International Workshop on Algorithms in Bioinformatics*, pages 222–232, 2008.