JOURNAL OF COMPUTATIONAL BIOLOGY
Volume 30, Number 7, 2023

Pp. 814–828
DOI: 10.1089/cmb.2022.0376

Growing Directed Acyclic Graphs: Optimization Functions for Pathway Reconstruction Algorithms

TUNÇ BAŞAR KÖSE,[1,*,†] JIARONG LI,[1,‡,*] and ANNA RITZ[2]

ABSTRACT

A major challenge in molecular systems biology is to understand how proteins work to transmit external signals to changes in gene expression. Computationally reconstructing these signaling pathways from protein interaction networks can help understand what is missing from existing pathway databases. We formulate a new pathway reconstruction problem, one that iteratively grows directed acyclic graphs (DAGs) from a set of starting proteins in a protein interaction network. We present an algorithm that provably returns the optimal DAGs for two different cost functions and evaluate the pathway reconstructions when applied to six diverse signaling pathways from the NetPath database. The optimal DAGs outperform an existing *k*-shortest paths method for pathway reconstruction, and the new reconstructions are enriched for different biological processes. Growing DAGs is a promising step toward reconstructing pathways that provably optimize a specific cost function.

Keywords: directed acyclic graphs, graph algorithms, pathway reconstruction, signaling pathways.

1. MOTIVATION

Intracellular signaling pathways describe the molecules and interactions that convert a particular external signal (such as growth, proliferation, movement, or death) to the change of expression of one or more genes, culminating in a cellular response through transcriptional regulation. Many signaling pathway databases such as Reactome (Fabregat et al, 2018), the Kyoto Encyclopedia of Genes and Genomes (KEGG) (Kanehisa et al, 2007), and NetPath (Kandasamy et al, 2010) document the interactions associated with signaling pathways, and efforts such as WikiPathways (Pico et al, 2008) and Pathway Commons (Cerami et al, 2010) have unified these databases by combining dozens of pathway resources.

Departments of [1]Computer Science and [2]Biology, Reed College, Portland, Oregon, USA.
*These authors contributed equally to this work.
[‡]Current affiliation: Microsoft Corporation, Mountain View, California, USA.
[†]Current affiliation: School of Science, Aalto University, Espoo, Finland.

However, even pathways that describe fundamental biological processes or pathways implicated in commonly studied diseases are not complete: they are likely missing canonical proteins and protein interactions. Further, it is difficult to use these pathway databases to study less well-known signaling events. Computationally reconstructing a signaling pathway of interest from experimental data would be a huge help for the (mostly manual) curation of pathway databases such as Reactome and KEGG, and it would provide a new lens to investigate signaling pathways that are not yet cataloged in these databases.

We and others have made use of large protein–protein interaction datasets as well as the annotated pathway databases to construct an *interactome*—a graph representation of physical interactions among all proteins in the organism under study. Since some interactions (especially those from existing databases) represent post-translational modifications where a direction of signal is clear, we consider an interactome as a weighted, directed graph $G=(V, E)$ where the edges are weighted according to the supporting evidence.

1.1. Pathway reconstruction problem

In an earlier work (Ritz et al, 2016), we formulated the following *Pathway Reconstruction Problem*: Given a weighted, directed interactome $G=(V, E)$, a set $R \subset V$ of membrane-bound receptors specific to a pathway of interest, and a set $T \subset V$ of transcriptional regulators specific to a pathway of interest, return a subgraph $G' \subseteq G$ that corresponds to the signaling pathway that connects nodes in R to nodes in T. We showed that a k-shortest paths (KSP) approach, PathLinker, outperformed existing methods for connecting nodes within a network to reconstruct pathways in pathway databases.

There are two main reasons why a KSP approach such as PathLinker reconstructed pathways better than other methods. First, a KSP approach has a parameter to smoothly ''grow'' the network. Many existing algorithms return a small reconstruction because they aim at minimizing the number of extraneous edges [such as Steiner forests (Tuncbag et al, 2013) and network flow (Lan et al, 2011)]. In KSP, we can increase the number of paths k to return iteratively larger reconstructions, capturing more of the pathway.

Second, a KSP approach is guaranteed to connect nodes in R to nodes in T. Although some methods such as Random Walks with Restarts (RWR) (Haveliwala, 2003) have a parameter that ''grows'' the network, they are not guaranteed to connect the nodes from R to T. In a KSP approach, all paths start and end within the pathway by construction. Finally, reconstructions that comprised the shortest paths are useful for generating hypotheses for follow-up validation.

For example, we experimentally validated a path (Ryk-CFTR-Dab2) from PathLinker's reconstruction of the Wnt signaling pathway and showed that Cystic Fibrosis Transmembrane Conductance Regulation (CFTR) is associated with Wnt signaling through interactions between Ryk, a non-canonical receptor, and Dab2, an inhibitor of canonical Wnt signaling (Ritz et al, 2016).

Despite PathLinker's success, calculating the first 20,000 shortest paths from receptors to transcriptional regulators for each signaling pathway in the NetPath databases captured only about 70% of the known signaling interactions (Ritz et al, 2016), leaving much room for improvement. Extensions of PathLinker have used auxiliary data such as protein localization information (Youssef et al, 2019) or additional downstream processing (Rubel and Ritz, 2020) to accurately reconstruct ground truth pathways. Other extensions have constrained the paths from R to T to follow regular expression patterns (Wagner et al, 2019).

1.2. Contributions

Inspired by the success of KSP approaches, we reframe the Pathway Reconstruction Problem to directly optimize an objective function related to path cost on directed acyclic graphs (DAGs). We acknowledge upfront that cycles (in the form of feedback loops) are important in signaling pathways, so this problem formulation is narrower in scope than the original Pathway Reconstruction Problem. Next, we present two pathway reconstruction methods that solve this new variant of the problem and apply them to six signaling pathways from the NetPath database (Kandasamy et al, 2010).

We show that DAG reconstructions exhibit different topologies; they often outperform a previous reconstruction method in recovering annotated proteins and interactions, and the predicted nodes are enriched for different biological processes. Finally, we highlight our method's ability to reconstruct pathways starting from a larger seed DAG, such as a DAG constructed from known pathway interactions.

2. METHODS

Let the interactome $G=(V, E)$ be directed with edge weights w_{uv} for every edge $(u, v) \in E$. Given $R, T \subset V$ denoting the set of receptors and set of transcription factors for a particular pathway of interest, we first modify the graph by Equation (1) introducing a super source node s to G, adding directed edges with 0 edge weight from s to each node in R, and removing all other incoming edges to nodes in R and Equation (2) introducing a super sink node t to G, adding directed edges with 0 edge weight from each node in T to t, and removing all other outgoing edges from nodes in T.

We now focus on finding s–t paths in G, which corresponds to finding paths from *any* receptor in R to *any* transcriptional regulator in T that have the same cost as the original path lengths from receptors to transcriptional regulators. Further, nodes in R and T will only start and end the paths (they will never be internal nodes on paths). KSP approaches are parameterized by k, the number of shortest paths from s to t. KSP approaches iteratively ''grow'' a subgraph G_k of G by taking the union of the first KSP from s to t: $G_1, G_2, \ldots, G_j, \ldots, G_k$. In this way, KSP approaches iteratively ''grow'' a subgraph of G. Our goal is to grow DAGs from a graph G.

2.1. A problem formulation for growing DAGs

We now formulate an optimization problem to grow DAGs from a graph G. Let $G_0 \subset G$ be any DAG that connects s and t (e.g., the shortest s–t path). We will also keep track of all paths in a DAG: Let $P_j(s, t)$ be the set of paths from s to t in DAG G_j. Finally, we define the new edges added to G_j to be $E_\Delta = E_j \backslash E_{j-1}$ and the set of new paths added to be $P_\Delta = P_j(s, t) \backslash P_{j-1}(s, t)$.

2.1.1. The growing DAG problem. Given a weighted, directed graph $G=(V, E, w)$ modified with super-source and super-sink nodes s and t, a DAG $G_0 \subset G$ that connects s and t, and a parameter k. For $j=1, 2, \ldots, k$, find a DAG $G_j=(V_j, E_j, w_j)$ where $G_{j-1} \subset G_j \subseteq G$ such that

1. $P_\Delta \neq \emptyset$ (there exists at least one new s–t path).
2. $\bigcup_{P_\Delta} = E_\Delta$ (all new edges in G_j are on some s–t path).
3. G_j minimizes some cost function $c : G_j \mapsto \mathbb{R}$, which may be one of the following:

min_edge_cost: $c_1(G_j) = \sum_{(u, v) \in E_j} w_{uv}$.
min_paths_cost: $c_2(G_j, s, t) = \sum_{p \in P_j(s, t)} \sum_{(u, v) \in p} w_{uv}$.

Cost function **min_edge_cost** is simply the cost of the edges in the subgraph G_j, whereas cost function **min_paths_cost** computes the cost of all paths in G_j. An example iteration is shown in Figure 1; note that the G_1 that minimizes the two costs functions are quite different in (C) and (D), though they both add at least one new path and all new edges are on some s–t path.

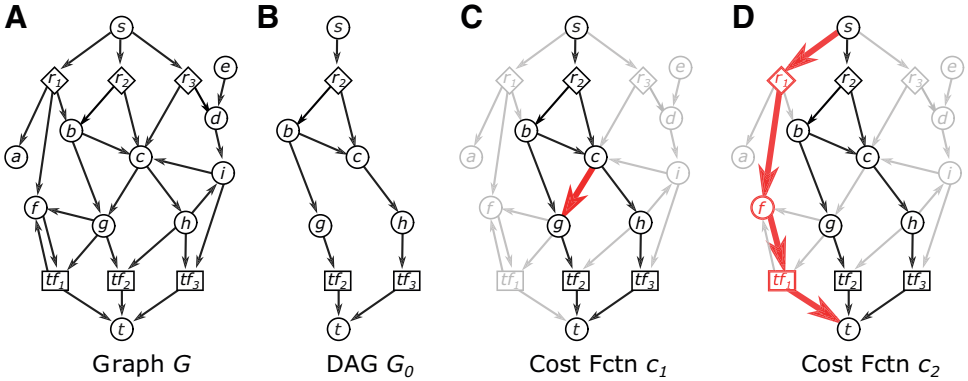


FIG. 1. The growing DAG problem. **(A)** Input graph G, which may contain cycles and bidirected edges. Node s is connected to three receptors r_1, r_2, r_3; three transcription factors tf_1, tf_2, tf_3 are connected to node t. In this example, all edges have unit cost. **(B)** An example DAG $G_0 \subset G$. **(C)** An example G_1 that minimizes **min_edge_cost** by adding a single edge to G_0. **(D)** An example G_1 that minimizes **min_paths_cost** by adding a path of length 4 to G_0. DAG, directed acyclic graph.

The shortest-path aspect of this problem comes from minimizing the cost functions. Note that, although KSP approaches will always calculate the jth shortest s–t path in G, this path may comprise edges already in G_{j-1} and thus fail the first condition cited earlier. The KSP approaches are also not guaranteed to produce a DAG at each iteration.

2.2. Calculating cost functions

To implement the Growing DAG Problem, we must show that we can efficiently determine whether a DAG G_j satisfies the properties listed and that we can efficiently identify such a DAG from the previous iteration. We first show that, given a DAG G_j, we can efficiently calculate the cost functions. Given G_{j-1} and G_j, we can efficiently calculate the cost functions described earlier. Calculating **min_edge_cost** is straightforward: simply sum the edge weights in G_j. However, calculating **min_paths_cost** requires an enumeration of all paths in G_j. To calculate **min_paths_cost**, we want to compute the cost of all s–t paths $P_j(s, t)$ in G_j without having to enumerate all paths. It can be rewritten to calculate, for each edge in E_j, the number of s–t paths that contain that edge. Let f_{uv} be the number of paths in $P_j(s, t)$ that contain edge (u, v):

$$c_2(G_j, s, t) = \sum_{p \in P_j(s,t)} \sum_{(u,v) \in p} w_{uv} = \sum_{(u,v) \in E_j} f_{uv} w_{uv}. \quad (1)$$

Although counting the number of s–t paths in a general graph is #*P*-complete* (Valiant, 1979), we can efficiently count the number of s–t paths in a DAG. Further, the dynamic program to count the number of s–t paths in a DAG will also compute the number of s–t paths f_{uv} that pass through every edge $(u, v) \in E_j$. Supplementary Section S1.1 describes the PathCounter() algorithm, which returns f_{uv} for every edge (u, v) in a graph. Once we have f from the PathCounter() algorithm, calculating **min_paths_cost** is straightforward using Equation (1).

2.3. Properties of an optimal G_j

Now that we have shown that we can efficiently calculate the cost functions given a DAG G_j, we will describe how to identify the possible extensions of G_{j-1} that guarantee that at least one new s–t path is added to G_j. There may be multiple edges that are added to grow G_j from G_{j-1}; let $G_\Delta = (V_j \backslash V_{j-1}, E_j \backslash E_{j-1})$ be the difference of G_j and G_{j-1}. Since G_{j-1} and G_j are DAGs, the graph G_Δ will also be a DAG (which may be disconnected). We first prove some properties of G_Δ that hold for either of the cost functions listed.

Lemma 1. *Given G_{j-1} and G_j that satisfy the Growing DAG problem, let $G_\Delta = G_j \backslash G_{j-1}$. The G_Δ that minimizes any of the cost functions will be exactly one path.*

Proof. G_Δ is a non-empty DAG, since at least one new s–t path must exist in G_j. If G_Δ is not exactly one path, then G_Δ must be composed of multiple paths. Since all new edges in G_j must be on some s–t path, then every maximal path in G_Δ must start and end on some node from G_{j-1} (which may include s and t). Let $P = \{p_1, p_2, \ldots\}$ be the set of distinct maximal paths from G_Δ.

If one path $p_i \in P$ establishes a new s–t path in G_j, then other paths are unnecessary. Let p_i start at some node v_0 and end at some node v_k; p_i establishes a new s–t path if $(s \rightsquigarrow v_0 \rightsquigarrow v_k \rightsquigarrow t)$ is in G_j. The other paths in P will add extraneous edges that are not necessary to establish a new path, which increases cost function **min_edge_cost**.

Further, the other paths in P will add additional s–t paths in G_j, which increases the cost function **min_paths_cost**. Since maximal paths can be dropped from P to minimize either cost function, G_Δ is not optimal.

Suppose instead that multiple paths are needed to establish a new s–t path in G_j. Without loss of generality, let two paths from P be $v_0 \rightsquigarrow v_k$ and $u_0 \rightsquigarrow u_l$, and let a new s–t path in G_j be

$$s \rightsquigarrow v_0 \rightsquigarrow v_k \rightsquigarrow x \rightsquigarrow x' \rightsquigarrow u_0 \rightsquigarrow u_l \rightsquigarrow t,$$

*#P (Sharp-P) problems are the counting problems associated with decision problems in *NP*.

where $x \rightsquigarrow x'$ is a path in G_{j-1}. There must be at least one edge from G_{j-1} between v_k and u_0 because otherwise p_i and p_j would form a single maximal path in G_Δ. Since G_{j-1} is a connected DAG, then $x \rightsquigarrow x'$ must be upstream of t and/or downstream of s in a topological ordering of V_{j-1}. Therefore, at least one of the following paths must exist:

1. $s \rightsquigarrow v_0 \rightsquigarrow v_k \rightsquigarrow x \rightsquigarrow x' \rightsquigarrow t$ if x is upstream of t. In this case, p_i is used but p_j is extraneous and only increases the cost functions (by adding extra edges for **min_edge_cost** and adding extra s–t paths for **min_paths_cost**), so G_Δ is not optimal.
2. $s \rightsquigarrow x \rightsquigarrow x' \rightsquigarrow u_0 \rightsquigarrow v_l \rightsquigarrow t$ if x is downstream of s. In this case, p_j is used but p_i is extraneous and only increases the cost functions (by adding extra edges for **min_edge_cost** and adding extra s–t paths for **min_paths_cost**), so G_Δ is not optimal. □

Lemma 2. *G_Δ that minimizes the cost functions defines a path that starts at some node in G_{j-1} and ends at some node in G_{j-1}. All internal nodes on the path are in G_j but not in G_{j-1}.*

Proof. G_Δ is a single path $p = (v_0, v_1, \ldots, v_k)$ by Lemma 1. We need to show that v_0 and v_k are in V_{j-1}, and all other nodes are not in V_{j-1}. First observe that there must be at least two nodes in G_Δ that are in G_{j-1} for an s–t path to include new edges in G_j (which may include s and/or t); call these nodes x and y. Consider the path in G_j $s \rightsquigarrow x \rightsquigarrow y \rightsquigarrow t$ where $x \rightsquigarrow y$ is a path in G_Δ.

1. There are exactly two nodes $x, y \in V_{j-1}$ in the path $p = (v_0, v_1, \ldots, v_k)$. Suppose there was a third node, $z \in V_{j-1}$, in the path such that $x \rightsquigarrow z \rightsquigarrow y$. Since G_{j-1} is a connected DAG, then z must be upstream of t and/or downstream of s. Therefore, at least one of the following paths must exist:
   a. $s \rightsquigarrow x \rightsquigarrow z \rightsquigarrow t$ if z is upstream of t. In this case, $z \rightsquigarrow y$ is extraneous and increases the cost functions (by adding extra edges for **min_edge_cost** and adding extra s–t paths for **min_paths_cost**), so G_Δ is not optimal.
   b. $s \rightsquigarrow z \rightsquigarrow y \rightsquigarrow t$ if z is downstream of s. In this case, $x \rightsquigarrow z$ is extraneous and increases the cost functions (by adding extra edges for **min_edge_cost** and adding extra s–t paths for **min_paths_cost**), so G_Δ is not optimal.
2. $x = v_0$. Suppose x is some node on the path p that is not v_0; call it v_i. The path from $(v_0, \ldots, v_{i-1})$ could be dropped, because $s \rightsquigarrow x$ will bypass these nodes.
3. $y = v_k$. Same argument as for $x = v_0$. □

Together, Lemmas 1 and 2 prove the following theorem:

Theorem 1. *Given G_{j-1} and G_j that satisfy the Growing DAG problem, G_{j-1} and G_j differ by exactly one path that starts and ends in G_{j-1} and contains no other nodes from G_{j-1}.*

2.4. The growing DAG algorithm

At each iteration j, we keep track of candidate paths for DAG G_j using a modified Dijkstra's algorithm. We rely on topologically sorting the nodes in a DAG, which results in a partial ordering of the nodes. Let $\sigma(G)$ denote the partial ordering of a DAG G, with the position of each node v denoted by σ_v. For a node v in a DAG, its *ancestors* are all nodes u where $\sigma_u < \sigma_v$ and its *descendants* are all nodes x where $\sigma_v < \sigma_x$. Note that there may also be the node y that is neither an ancestor nor a descendant of v (e.g., where $\sigma_v = \sigma_y$); we say that v and y are *incomparable*.

Algorithm 1 takes as input a directed, weighted graph G, an initial DAG G_0, the number of iterations k, and a cost function c (either **min_edge_cost** or **min_paths_cost**). It returns a list of length k that denotes the min-cost paths for each iteration according to the specified cost function. To track the set of candidate paths, we use the distances dictionary that stores the cost of the best paths between topologically sorted nodes in the DAG. We assume that we also have the predecessors so we can determine the path $u \rightsquigarrow v$ from dist $[u][v]$ as well.

Lines 3–8 initialize the dist dictionary for the nodes in G_0. First, we build a G_{cand} graph that removes the existing DAG from G. Then, for every node u in G_0 we consider, we must also remove any node that is an ancestor of u to prevent paths that would induce cycles. Once we have the graph, we call a slightly modified Dijkstra's algorithm called MultiTargetDijkstra() to find the shortest path from a source node s to each target node, ensuring that a node in G_j is never considered an internal node on a path and returning early if all target nodes are reached (Supplementary Section S1.2).

Algorithm 1: GrowDAGs($G=(V, E)$, $G_0=(V_0, E_0)$, num iters k, cost function c)

1: paths $\leftarrow$[]
2: dist $\leftarrow$ {}
3: *# Initialize distances*
4: $G_{\text{cand}} \leftarrow G \backslash G_0$
5: **for** $u \in V_0$ **do**
6: $G_{\text{cand}}^{(u)} \leftarrow$ Remove any node v from G_{cand} where $\sigma_u > \sigma_v$ in $\sigma(G_0)$
7: dist[u] = MultiTargetDijkstra($G_{\text{cand}}^{(u)}, V_0, u, \{v : \sigma_u \leq \sigma_v\}$ in $\sigma(G_0)$)
8: **end for**
9: **for** $j = 1, 2, \ldots, k$ **do**
10: *# Get the min cost path according to cost function c*
11: **if** $c = c_1$ **then**
12: paths[j] $\leftarrow$ the path $u \rightsquigarrow v$ from dist[u][v] such that $\min_{x,y}$ dist[x][y] = dist[u][v]
13: **else if** $c = c_2$ **then**
14: $C \leftarrow$ {}
15: **for** u, v such that $\sigma_u \leq \sigma_v$ in $\sigma(G_j)$ **do**
16: $G' \leftarrow$ Add the path $u \rightsquigarrow v$ from dist[u][v] to G_{j-1}
17: $f \leftarrow$ PathCounter (G',s,t)
18: $C[u][v] = \sum_{(u,v) \in E'} f_{uv} w_{uv}$
19: **end for**
20: paths[j] $\leftarrow$ the path $u \rightsquigarrow v$ from dist[u][v] such that $\min_{x,y} C[x][y] = C[u][v]$
21: **end if**
22: *# Make G_j and update distances*
23: $G_j \leftarrow$ Add paths[j] to G_{j-1}
24: $G_{\text{cand}} \leftarrow G \backslash G_j$
25: **for** $u \in V_j$ **do**
26: $G_{\text{cand}}^{(u)} \leftarrow$ Remove any node v from G_{cand} where $\sigma_u > \sigma_v$ in $\sigma(G_j)$
27: dist[u] = MultiTargetDijkstra($G_{\text{cand}}^{(u)}, V_j, u, \{v : \sigma_u \leq \sigma_v\}$ in $\sigma(G_j)$)
28: **end for**
29: **end for**
30: **return** paths

Once the distances are initialized, we iteratively grow G_j for j from 1 to k. First, we identify the min-cost path from dist according to cost function c (Lines 10–21). This is simple for **min_edge_cost**, when we select the path $u \rightsquigarrow v$ with the lowest cost in dist. For **min_paths_cost**, we need to calculate the cost of all s–t paths for every possible choice of valid $u \rightsquigarrow v$ path in dist. We can call the PathCounter() algorithm (Supplementary Algorithm S1) on a graph G' that contains the path we are checking to calculate the all-paths-cost $C[u][v]$ (Lines 16–18).

We select the path $u \rightsquigarrow v$ with the lowest all-paths-cost $C[u][v]$, add the selected path to G_{j-1} to make the G_j DAG, and update the distances dictionary.

We implemented a number of speedups to improve the runtime of Algorithm 1, two of which are notable. First, in the MultiTargetDijkstra() function call, we can set the target nodes T to be only the nodes v for which $u \rightsquigarrow v$ needs to be recalculated (Supplementary Section S1.2). In practice, this dramatically reduced the number of target nodes that MultiTargetDijkstra() needed to find, re-running between 0.2% and 14% of the possible number of target nodes across all runs.

A second speedup comes from avoiding re-computations when minimizing **min_paths_cost**. Lines 16–18 generate a new DAG G', calls PathCounter() on this new graph, and uses these values to compute the **min_paths_cost** for every ordered node pair u, v. In practice, we first calculate f for G_j as the first step in the main for loop, and we use functions to update f as if we have added the path $u \rightsquigarrow v$ (without needing to create a new graph G'). These update functions, similar to determining the upstream and downstream dictionaries in the PathCounter() algorithm, adjust all paths from s to u and all paths from v to t assuming that path $u \rightsquigarrow v$ has been added.

3. RESULTS

We ran GrowDAGs for both cost functions to reconstruct six diverse signaling pathways from the NetPath database (Kandasamy et al, 2010). We identified receptors and transcriptional regulators for each pathway using previously established resources of these protein types (Almén et al, 2009; Ravasi et al, 2010; Vaquerizas et al, 2009). The six networks range in size, density, and the number of receptors and transcriptional regulators (Table 1).

We used an interactome G with 17,513 nodes and 577,617 directed edges weighted by experimental evidence (Youssef et al, 2019). This interactome is built from a combination of molecular interaction data (BioGrid, DIP, InnateDB, IntAct, MINT, PhosphositePlus) and annotated signaling pathway databases (KEGG, NetPath, and SPIKE). Since NetPath nodes and edges are part of the interactome, we can use these pathways as ground truth networks to assess reconstruction methods [similar to the analyses in previous work (Ritz et al, 2016; Youssef et al, 2019)]. The edge weights are negative log transformed to become edge costs, which we aim at minimizing.

3.1. Topological differences between DAG reconstructions

We first provide some examples of reconstructed pathways using GrowDAGs by examining the topological differences between DAG reconstructions of the Wnt signaling pathway. We grew DAGs from each pathway-specific G_0 defined as the shortest s–t path in G, where super source node s is connected to the pathway's receptors and the pathway's transcriptional regulators are connected to super sink t (Fig. 1A).

As expected, **min_edge_cost** tended to produce DAGs whose paths reuse the same nodes whereas **min_paths_cost** tended to produce DAGs with more non-overlapping s–t paths (Fig. 2 shows $k = 75$ for the Wnt pathway reconstructions). Network visualizations of all pathways up to $k = 200$ are available on GraphSpace (Bharadwaj et al, 2017).[†]

We also compared the resulting GrowDAGs reconstructions with those from PathLinker, the KSP approach (Ritz et al, 2016). We ran the GrowDAGs methods (starting from the shortest-path G_0) and PathLinker on each of the six pathways until the reconstruction contained the same number of nodes or edges as the ground truth pathway (or until $k = 1000$). We call these *size-matched reconstructions*, and they depend on whether the reconstructions are matched by the number of nodes or the number of edges.

Supplementary Table S1 shows the k values needed for size-matched reconstructions for all six pathways; two node-based reconstructions and three edge-based reconstructions from the BCR pathway and the EGFR1 pathway did not reach the ground truth number of nodes or edges by $k = 1000$, so these comparisons are not necessarily size-matched.

The Wnt size-matched reconstruction for **min_paths_cost** took fewer iterations than **min_edge_cost** or PathLinker for both nodes and edges, indicating that **min_paths_cost** tends to add new nodes and edges to reconstructions rather than reusing existing nodes and edges (Fig. 3A). This observation about **min_paths_cost** holds across size-matched reconstructions for all six pathways, though Wnt tends to be an outlier when comparing **min_edge_cost** with PathLinker (Supplementary Figs. S2 and S3). For all other pathway reconstructions, **min_edge_cost** takes the most iterations to reach size-matched reconstructions for nodes, and PathLinker falls between the two DAG methods.

3.2. Comparison to ground truth pathways

We then evaluated how well the GrowDAG pathways captured annotated proteins and interactions from the six NetPath pathways. For this analysis, we considered the NetPath pathways ground truth nodes and edges, though it is widely acknowledged that signaling pathway databases are incomplete. For each of the six pathways, we calculated the precision and recall for size-matched reconstructions for nodes and edges (Fig. 3B; Supplementary Figs. S4 and S5).

When computing precision and recall of edges, we compare undirected edges with undirected ground truth edges, which is consistent with previous evaluations (Ritz et al, 2016). Overall, the methods achieve relatively low recall (e.g., 0.15–0.5 recall for nodes) despite relatively high precision (Supplementary Figs. S4 and S5). The GrowDAG reconstructions have a higher area under the precision-recall (AUPRC)

[†]http://graphspace.org/graphs/?query=tags:DAG#

TABLE 1. SIGNALING PATHWAYS CHOSEN FOR THIS STUDY (KANDASAMY ET AL, 2010)

Name	*Abbreviations*	*Nodes*	*Edges*	*Receptors*	*TRs*
B cell receptor pathway	BCR	137	456	1	18
Epidermal growth factor receptor pathway	EGFR1	231	1456	6	33
Interleukin 1 pathway	IL1	43	178	3	5
T cell receptor pathway	TCR	154	504	7	20
Transforming growth factor β pathway	TGFβ	209	863	5	77
Wnt pathway	Wnt	106	428	14	14

TRs, transcriptional regulators.

than PathLinker in all but one case (T cell receptor), whose max AUPRC is an order of magnitude smaller than any other case (Table 2). Of the GrowDAG cost functions, **min_paths_cost** has the highest AUPRC for seven cases compared with four cases for **min_edge_cost**.

Although the methods have low recall, overall size matched-sized reconstructions are often too large for visual exploration (e.g., the networks visualized in Fig. 2 are only a subset of the matched-sized reconstructions). Instead, considering the precision for the top 50 predictions (e.g., the first 50 nodes or first 50 edges in a reconstruction) is a complementary assessment in terms of reconstruction usefulness. When considering the first 50 predictions, DAG **min_paths_cost** no longer consistently outperforms the other two methods, though one of the GrowDAG reconstructions has the highest precision for five of the node reconstructions and five of the edge reconstructions (Table 3).

One of the reasons for the low precision, especially for edges, is that the number of negative examples is vastly larger than the number of positive examples for a single pathway. Thus, previous evaluation frameworks have subsampled the negative examples from the interactome to ensure a 50:1 ratio of negatives to positives (Ritz et al, 2016). This not only inflates the precision but can also more clearly separate performances of different approaches.

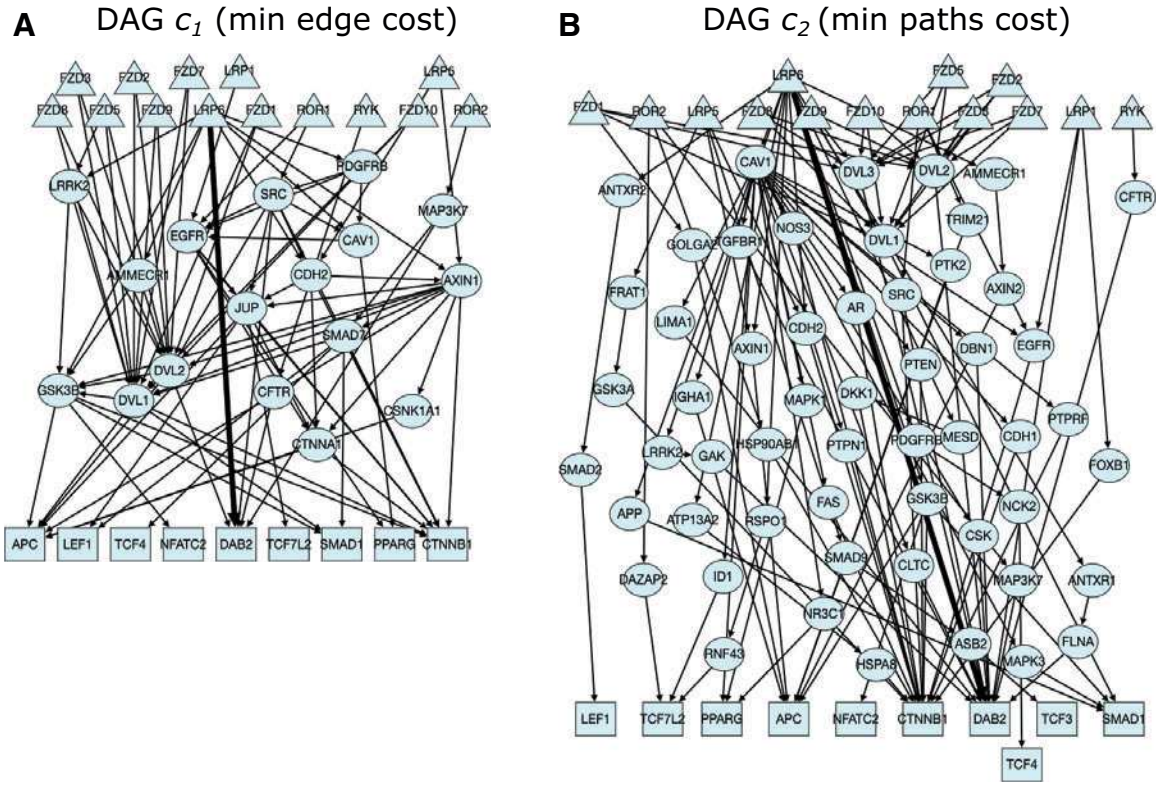


FIG. 2. Wnt pathway GrowDAGs reconstructions ($k = 75$) for **(A) min_edge_cost** and **(B) min_paths_cost**. The shortest path from any receptor to any transcriptional regulator was used as G_0 (shown as the thick edge).

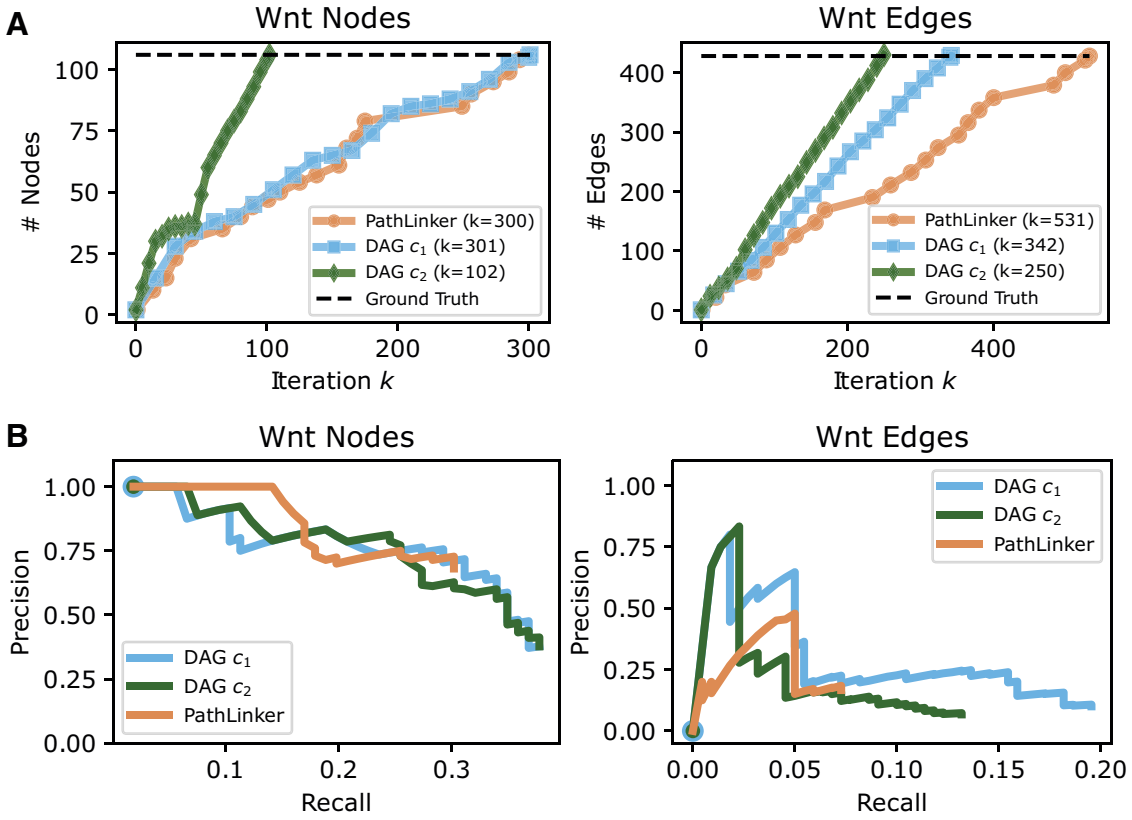


FIG. 3. **(A)** Number of nodes and edges at each iteration of the Wnt pathway reconstructions. Horizontal dashed line indicates the number of nodes and edges in the Wnt NetPath pathway. **(B)** Precision-recall curves for size-matched reconstructions from the Wnt pathway. c_1: **min_edge_cost**; c_2: **min_paths_cost**.

We conducted the same precision-recall analysis by subsampling negatives in a 50:1 ratio and repeating for 10 iterations—the GrowDAG reconstructions outperform PathLinker even when subsampling negatives, though more **min_edge_cost** reconstructions have the highest AUPRC across methods (Supplementary Figs. S6 and S7 and Supplementary Table S2).

3.3. Reconstructions capture diverse biological processes

Next, we evaluated whether the three pathway reconstruction methods recover different biological processes. Here, we considered the nodes in the sized-matched reconstructions and found that, for five of

Table 2. Area Under the Precision-Recall Values for Nodes (Left) and Edges (Right) for Size-Matched Reconstructions

	Node AUPRC			*Edge AUPRC*		
Name	*DAG c_1*	*DAG c_2*	*PL*	*DAG c_1*	*DAG c_2*	*PL*
BCR	0.179	**0.242***	0.132	0.018	**0.029**	0.003
EGFR1	**0.275**	0.249*	0.234	**0.132***	0.013*	0.012*
IL1	0.148	**0.259**	0.225	0.020	**0.047**	0.030
TCR	0.130	**0.187**	0.176	0.003	0.003	**0.006**
TGFβ	**0.397**	0.377	0.355	0.052	**0.060**	0.036
Wnt	0.274	**0.276**	0.272	**0.057**	0.030	0.028

c_1: **min_edge_cost**; c_2: **min_paths_cost**. Largest AUPRC values are shown in bold; asterisks (*) denote reconstructions that did not reach the number of ground truth nodes or edges by $k = 1000$ (see Supplementary Table S1).

AUPRC, area under the precision-recall; DAG, directed acyclic graphs; PL, PathLinker.

Table 3. Precision of the First 50 Predictions (Nodes on the Left, Edges on the Right) for the Reconstructions

	Node precision @50			*Edge precision @50*		
Name	*DAG* c_1	*DAG* c_2	*PL*	*DAG* c_1	*DAG* c_2	*PL*
BCR	0.580	**0.720**	0.392	0.040	**0.231**	0.040
EGFR1	**0.880**	0.725	0.660	0.120	**0.180**	0.100
IL1		**0.380**	0.320	0.120	0.137	**0.157**
TCR	0.480	0.615	**0.640**	**0.120**	0.020	0.118
TGFβ	**0.960**	0.860	0.640	0.360	**0.440**	0.255
Wnt	**0.660**	0.627	0.640	**0.240**	0.200	0.220

c_1: **min_edge_cost**; c_2: **min_paths_cost**. Largest values are shown in bold; missing entries denote that 50 unique predictions weren't reached at the specified value of k for sized-matched reconstructions.

the six pathways, each method contains distinct nodes not found in the other reconstructions (Supplementary Fig. S8). For example, only 17% of the predicted nodes for Wnt size-matched reconstructions are predicted by all methods (Fig. 4).

We performed gene function enrichment using PantherDB (Mi and Thomas, 2009; Mi et al, 2019) on the uniquely-predicted nodes for each method (e.g., 54 DAG **min_edge_cost** nodes, 46 DAG **min_paths_cost** nodes, and 48 PathLinker nodes in Fig. 4) to determine whether any Panther pathways are enriched in each of these unique sets. Specifically, we ran the PANTHER Over-representation Test (version 17.0) using Fisher's exact test and an FDR correction of 0.05.

We found that each set of unique nodes was enriched for at least three Panther pathways (tables in Fig. 4). The DAG **min_paths_cost** has 24 enriched Panther pathways; the full table is in Supplementary Table S3. All three reconstructions contain nodes that are enriched in Wnt signaling (DAG **min_paths_cost** captures three nodes with an adjusted p-value of 3.62×10^{-2}; Supplementary Table S3), each capturing different aspects of the Wnt pathway that is separate from the other two methods.

The DAG **min_paths_cost** and PathLinker reconstructions are also enriched for Parkinson Disease, which has a known connection to Wnt signaling through inflammatory pathways (L'Episcopo et al, 2014). The DAG **min_paths_cost** also appears to be enriched in many signaling pathways that are known to cross-talk with Wnt, including T cell signaling (Li et al, 2019), B cell signaling (Ma and Hottiger, 2016), and EGFR signaling (Hu and Li, 2010), among others.

3.4. G_0 *as NetPath DAGs*

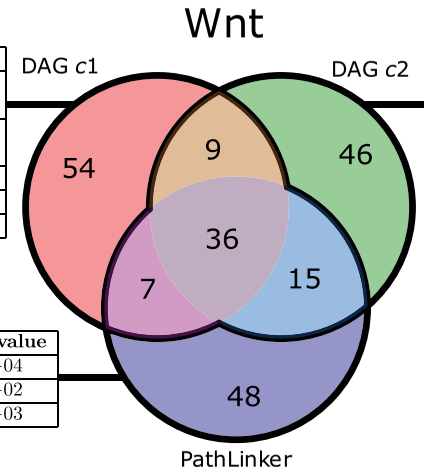
One of the powerful aspects of GrowDAGs is the ability to begin the reconstruction with any DAG. For example, using the ground truth pathways as input DAGs, our methods will propose new nodes and edges that are not currently annotated to that pathway. To illustrate this potential, we converted each ground truth pathway from NetPath into a G_0 DAG by considering only the nodes and edges in the ground truth network (Table 1) after connecting super source node s to all receptors and super sink node t to all transcriptional regulators (similar to the modifications made to the full interactome).

We calculated up to 1000 shortest s–t paths in the ground truth network (e.g., by running PathLinker) and built G_0 by adding each path only if it did not introduce a cycle in G_0. Since the NetPath edges are used to build the interactome, each ground truth DAG conversion G_0 is, by construction, a subgraph of the interactome. The total number of nodes and edges in the DAG conversion of G_0 were correlated with the overall size of the ground truth pathway (Table 4). We ran GrowDAGs for $k = 25$ to be able to visualize the newly-added nodes and edges.

Running GrowDAGs on the six signaling pathways with the ground truth DAGs G_0 produced similar trends in network topology for **min_edge_cost** and **min_paths_cost** that were shown in Section 3.1. For the 25 new paths added to the NetPath DAGs, we also counted the number of directed edges that were present in Netpath or KEGG pathway databases, indicating that a known signaling interaction was recovered.

We found that, on average, 16.9 ± 3.4 directed edges from DAGs generated by **min_edge_cost** and 13.2 ± 3.0 directed edges from DAGs generated by **min_paths_cost** that were present in the databases. An example reconstruction for interleukin (IL)-1, the smallest pathway, is shown in Figure 5. These

Panther Pathway	n	k	Adj. P-value
Axon guidance mediated by semaphorins (P00007)	21	2	4.37e−02
Angiotensin II-stimulated signaling through G proteins and beta-arrestin (P05911)	37	3	1.25e−02
Ubiquitin proteasome pathway (P00060)	58	4	3.34e−03
CCKR signaling map (P06959)	173	4	3.79e−02
Wnt signaling pathway (P00057)	314	6	9.75e−03



Panther Pathway	n	k	Adj. P-value
Parkinson disease (P00049)	101	5	3.74e−04
EGF receptor signaling pathway (P00018)	142	4	1.47e−02
Wnt signaling pathway (P00057)	314	6	5.02e−03

Panther Pathway	n	k	Adj. P-value
Interferon-gamma signaling pathway (P00035)	30	3	6.64e−04
Toll receptor signaling pathway (P00054)	61	4	2.35e−04
Insulin/IGF pathway-mitogen activated protein kinase kinase/MAP kinase cascade (P00032)	33	2	2.21e−02
Parkinson disease (P00049)	101	6	7.12e−06
VEGF signaling pathway (P00056)	68	4	3.18e−04
FAS signaling pathway (P00020)	34	2	2.22e−02
B cell activation (P00010)	71	4	3.40e−04
T cell activation (P00053)	92	5	7.79e−05
Angiotensin II-stimulated signaling through G proteins and beta-arrestin (P05911)	37	2	2.38e−02
Ras Pathway (P04393)	76	4	4.03e−04
TGF-beta signaling pathway (P00052)	102	5	7.90e−05
Apoptosis signaling pathway (P00006)	125	6	1.77e−05
Gonadotropin-releasing hormone receptor pathway (P06664)	237	9	2.52e−07

...

FIG. 4. Venn diagram of predicted nodes for GrowDAG methods and PathLinker for size-matched Wnt reconstructions. Tables show gene function enrichment of uniquely predicted nodes using PantherDB pathways. n: number of proteins in the Panther pathway; k: number of predicted proteins in the Panther pathway; c_1: **min_edge_cost**; c_2: **min_paths_cost**. Full table for DAG **min_paths_cost** is shown in Supplementary Table S3.

Table 4. The Ground Truth Pathways Converted to Directed Acyclic Graphs, Which Are Used as G_0 in Section 3.4

Name	*PathLinker Paths*	*Nodes in DAG conversion*	*Edges in DAG conversion*
BCR	958	58	131
EGFR1	>1000	105	461
IL1	713	22	55
TCR	952	57	139
TGFβ	70	33	70
Wnt	984	40	89

The total number of PathLinker paths are also reported.

reconstructions potentially suggest new proteins and interactions that should be considered in the annotated pathways, and the percentage of interactions from signaling pathway databases are a promising sign of identifying signaling edges in these DAGs.

4. DISCUSSION

We have described the Growing DAG Problem, a pathway reconstruction formulation that directly adds paths to a DAG that explicitly optimizes one of two cost functions (minimizing edge costs or minimizing *s–t* path costs). We presented the GrowDAGs algorithm that iteratively finds the most optimal DAG G_j. Applying this algorithm to six diverse NetPath pathways, we show that **min_edge_cost** and **min_paths_cost** admit DAGs with different network topologies that often outperform a KSP approach compared with the annotated proteins and interactions in NetPath.

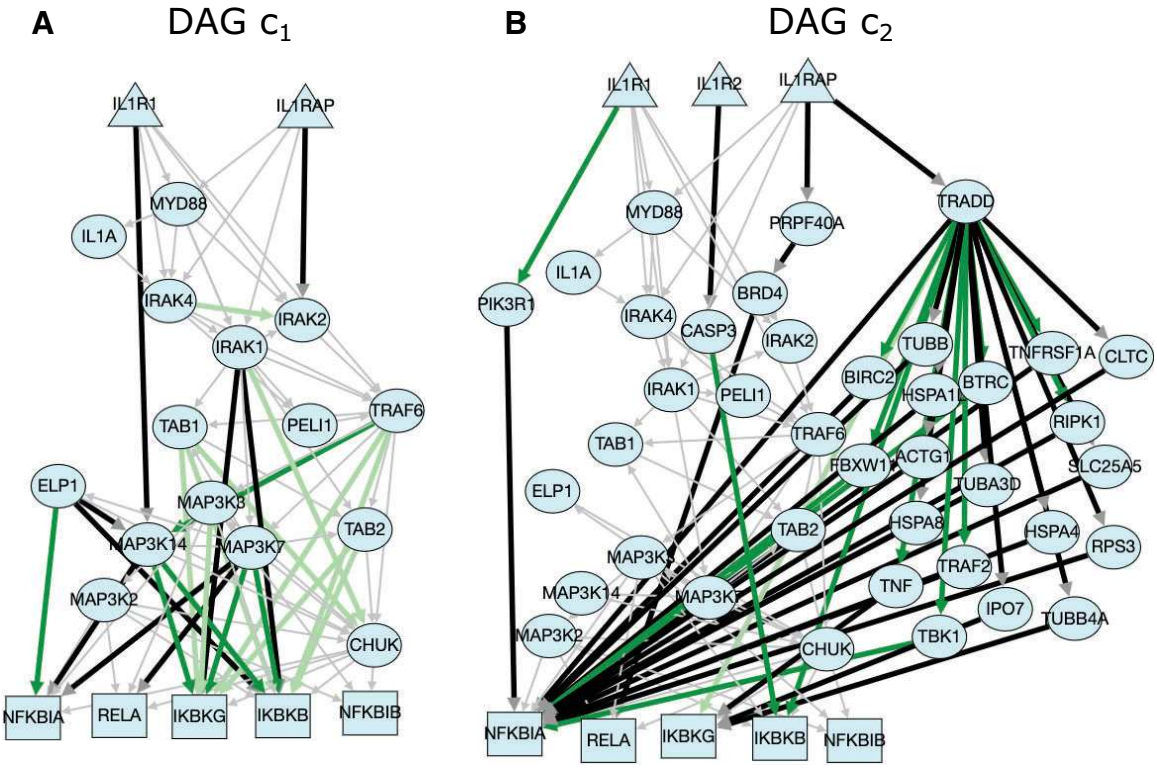


FIG. 5. IL-1 reconstructions for **(A) min_edge_cost** and **(B) min_paths_cost** for $k = 25$ starting from a ground truth G_0. Here, G_0 is shown in thin gray edges. Dark green edges appear in the NetPath database; light green edges appear in the Kyoto Encyclopedia of Genes and Genomes (KEGG) database. IL, interleukin.

Further, the reconstructions return proteins that are enriched for different biological processes. We also demonstrate that we can begin with a larger DAG G_0 such as a DAG converted from the NetPath ground truth. To our knowledge, this is the first pathway reconstruction algorithm that can grow a network based on a seeded subnetwork (rather than a set of nodes).

In our analysis of GrowDAGs and PathLinker reconstructions from six pathways, the GrowDAGs methods outperformed PathLinker for size-matched reconstructions in all scenarios except one. Between the two cost functions, DAG **min_paths_cost** had a higher AUPRC for more scenarios than DAG **min_edge_cost** (Table 2). It is worth noting, however, that DAG **min_edge_cost** performed the best on the EGFR1 reconstructions, which is the largest ground truth network we tested. DAG **min_edge_cost** also performed better than the other reconstructions for TFGβ when reconstructing size-matched nodes (Table 2).

Both EGFR1 and TGFβ node reconstructions have a high overlap between DAG **min_paths_cost** and PathLinker (as evidenced by the Venn diagrams in Supplementary Fig. S8). This suggests that, when DAG **min_paths_cost** and PathLinker produce similar reconstructions, DAG **min_edge_cost** may better reflect nodes in the ground truth pathway.

The GrowDAG formulations rely on a parameter k, which determines how large to grow the pathway reconstruction. The choice of k dramatically changes the size of the reconstructions, and this user-defined parameter should be chosen carefully with the user's goal in mind. For example, we calculated up to $k = 1000$ iterations to assess how well entire pathways could be reconstructed. When looking for potentially new proteins and interactions that are missing from an annotated pathway, the first few predictions (e.g., the top 50 predictions shown in Table 3) may be more appropriate. For visualization purposes, $k = 200$ begins to reach the limit of useful network visualization (see, e.g., the reconstructions available at http://graphspace.org/graphs/?query=tags:DAG#). In general, larger values of k will provide a better sense of how much of the pathway is captured in a reconstruction, whereas looking at the top predictions for small values of k will provide a ranked list of potentially new proteins and interactions that are missing from the annotated pathway.

The Growing DAG Problem and subsequent algorithms are limited by the DAG constraints—namely, that reconstructions will never contain cycles. Feedback loops are an integral part of signaling (Alon, 2007), and many of the signaling pathways in NetPath, KEGG, and other pathway databases have documented examples of these cycles.

Our reasons for starting with DAGs were twofold: first, intracellular signaling that begins at a cell membrane and ends with transcriptional regulation has a general direction; and second, DAGs provide a reduced search space when growing networks. Beyond the search space for optimal subgraphs at each iteration, the **min_paths_cost** optimization criterion could not be efficiently computed on a subgraph with cycles, though there may very well be another optimization function that would be useful to consider in general graphs. Generalizing the computational problem and the general framework to graphs with cycles would be an important future step in pathway reconstruction.

Although the GrowDAGs framework is a promising way to reconstruct signaling pathways with certain topologies, a few limitations remain that prohibit its widespread use for large pathway reconstructions. First, the algorithm is still slow in practice, namely due to having to check all pairs of topologically sorted nodes of G_{j-1} at each iteration.

Runtimes for $k = 100$ averaged 2.1 hours, ranging from 11 minutes to 16 hours (a full table of runtimes is available in Supplementary Table S4). However, computing the next iterations for $k = 101, \ldots, 200$ averaged 19.6 hours, ranging from 44 minutes to nearly 52 hours. While we mention a few speedups in Section 2.4, more improvements are needed to reconstruct pathways at the scale of PathLinker (which calculated reconstructions up to $k = 20{,}000$). Second, other cost functions may produce different topologies than those reconstructed here; for example, minimizing the average cost of new paths or taking a weighted average of **min_edge_cost** and **min_paths_cost**. However, Lemma 2 does not hold for such cost functions.

Despite these challenges, the Growing DAG Problem presents a first step toward explicitly optimizing a cost function related to pathway reconstruction. Existing pathway reconstruction methods will always return networks with the same topologies—KSP, Steiner forests, random walks, etc.—and the proposed cost functions provide control over the topologies of the GrowDAG reconstructions. This work opens the door for improved cost functions that reflect ground truth pathways and provides a framework for designing algorithms to grow networks.

ACKNOWLEDGMENTS

The authors thank Layla Oesper, Ibrahim Youssef, and Tobias Rubel for feedback on early versions of this manuscript.

AUTHORS' CONTRIBUTIONS

T.B.K. and J.L.: Methodology, software, formal analysis, and writing—original draft. A.R.: Conceptualization, methodology, software, formal analysis, writing—original draft, writing—review and editing, supervision, and funding acquisition.

DATA AND CODE AVAILABILITY

All code and data related to Growing DAGs are available on GitHub: https://github.com/Reed-CompBio/growing-dags. Pathway reconstruction visualizations for the six pathways and $k = 200$ are available on GraphSpace (Bharadwaj et al, 2017): http://graphspace.org/graphs/?query=tags:DAG#

AUTHOR DISCLOSURE STATEMENT

The authors declare they have no conflicting financial interests.

FUNDING INFORMATION

This work was supported by the National Science Foundation (DBI-1750981) and the M.J. Murdock Charitable Trust (Lynwood W. Swanson Promise for Scientific Research Award).

SUPPLEMENTARY MATERIAL

Supplementary Figure S1
Supplementary Figure S2
Supplementary Figure S3
Supplementary Figure S4
Supplementary Figure S5
Supplementary Figure S6
Supplementary Figure S7
Supplementary Figure S8
Supplementary Table S1
Supplementary Table S2
Supplementary Table S3
Supplementary Table S4
Supplementary Data

REFERENCES

Almén MS, Nordström KJ, Fredriksson R, et al. Mapping the human membrane proteome: A majority of the human membrane proteins can be classified according to function and evolutionary origin. BMC Biol 2009;7(1):1–14.

Alon U. Network motifs: Theory and experimental approaches. Nat Rev Genet 2007;8(6):450–461.

Bharadwaj A, Singh DP, Ritz A, et al. Graphspace: Stimulating interdisciplinary collaborations in network biology. Bioinformatics 2017;33(19):3134–3136.

Cerami EG, Gross BE, Demir E, et al. Pathway commons, a web resource for biological pathway data. Nucleic Acids Res 2010;39(suppl_1):D685–D690.

Fabregat A, Jupe S, Matthews L, et al. The reactome pathway knowledgebase. Nucleic Acids Res 2018;46(D1):D649–D655.

Haveliwala TH. Topic-sensitive pagerank: A context-sensitive ranking algorithm for web search. IEEE Trans Know Data Eng 2003;15(4):784–796.

Hu T, Li C. Convergence between wnt-β-catenin and egfr signaling in cancer. Mol Cancer 2010;9(1):1–7.

Kandasamy K, Mohan SS, Raju R, et al. Netpath: A public resource of curated signal transduction pathways. Genome Biol 2010;11(1):1–9.

Kanehisa M, Araki M, Goto S, et al. Kegg for linking genomes to life and the environment. Nucleic Acids Res 2007;36(suppl_1):D480–D484.

Lan A, Smoly IY, Rapaport G, et al. Responsenet: Revealing signaling and regulatory networks linking genetic and transcriptomic screening data. Nucleic Acids Res 2011;39(suppl_2):W424–W429.

L'Episcopo F, Tirolo C, Caniglia S, et al. Targeting wnt signaling at the neuroimmune interface for dopaminergic neuroprotection/repair in Parkinson's disease. J Mol Cell Biol 2014;6(1):13–26.

Li X, Xiang Y, Li F, et al. Wnt/β-catenin signaling pathway regulating t cell-inflammation in the tumor microenvironment. Front Immunol 2019;10:2293.

Ma B, Hottiger MO. Crosstalk between wnt/β-catenin and nf-κb signaling pathway during inflammation. Front Immunol 2016;7:378.

Mi H, Muruganujan A, Huang X, et al. Protocol update for large-scale genome and gene function analysis with the panther classification system (v. 14.0). Nat Protoc 2019;14(3):703–721.

Mi H, Thomas P. Panther Pathway: An Ontology-Based Pathway Database Coupled with Data Analysis Tools. In: Protein Networks and Pathway Analysis; Nikolsky Y, Bryant J (eds). Springer. 2009; pp. 123–140. doi: 978-1-60761-17s-2_7

Pico AR, Kelder T, Van Iersel MP, et al. Wikipathways: Pathway editing for the people. PLoS Biol 2008;6(7):e184.

Ravasi T, Suzuki H, Cannistraci CV, et al. An atlas of combinatorial transcriptional regulation in mouse and man. Cell 2010;140(5):744–752.

Ritz A, Poirel CL, Tegge AN, et al. Pathways on demand: Automated reconstruction of human signaling networks. NPJ Syst Biol Appl 2016;2(1):1–9.

Rubel T, Ritz A. Augmenting signaling pathway reconstructions. In Proceedings of the 11th ACM International Conference on Bioinformatics, Computational Biology and Health Informatics. Association for Computing Machinery, New York, NY, USA. 2020; pp. 1–10.

Tuncbag N, Braunstein A, Pagnani A, et al. Simultaneous reconstruction of multiple signaling pathways via the prize-collecting Steiner Forest Problem. J Comput Biol 2013;20(2):124–136.

Valiant LG. The complexity of enumeration and reliability problems. SIAM J Comput 1979;8(3):410–421.

Vaquerizas JM, Kummerfeld SK, Teichmann SA, et al. A census of human transcription factors: Function, expression and evolution. Nat Rev Genet 2009;10(4):252–263.

Wagner MJ, Pratapa A, Murali T. Reconstructing signaling pathways using regular language constrained paths. Bioinformatics 2019;35(14):i624–i633.

Youssef I, Law J, Ritz A. Integrating protein localization with automated signaling pathway reconstruction. BMC Bioinformatics 2019;20(16):1–15.

Address correspondence to:
Dr. Anna Ritz
Biology Department
Reed College
3203 SE Woodstock Blvd.
Portland, OR 97202
USA

E-mail: aritz@reed.edu