On the Maximum Connectivity Improvement Problem

Federico Corò[1(✉)], Gianlorenzo D'Angelo[1], and Cristina M. Pinotti[2]

[1] Gran Sasso Science Institute (GSSI), L'Aquila, Italy
{federico.coro,gianlorenzo.dangelo}@gssi.it

[2] Department of Computer Science and Mathematics, University of Perugia, Perugia, Italy
cristina.pinotti@unipg.it

Abstract. In this paper, we define a new problem called the Maximum Connectivity Improvement (MCI) problem: given a directed graph $G = (V, E)$, a weight function $w : V \rightarrow \mathbb{N}_{\geq 0}$, a profit function $p : V \rightarrow \mathbb{N}_{\geq 0}$, and an integer B, find a set S of at most B edges not in E that maximises $f(S) = \sum_{v \in V} w_v \cdot p(R(v, S))$, where $p(R(v, S))$ is the sum of the profits of the nodes reachable from node v when the edges in S are added to G. We first show that we can focus on Directed Acyclic Graphs (DAG) without loss of generality. We prove that the MCI problem on DAG is NP-Hard to approximate to within a factor greater than $1 - 1/e$ even if we restrict to graphs with a single source or a single sink, and MCI remains NP-Complete if we further restrict to unitary weights. We devise a polynomial time algorithm based on dynamic programming to solve the MCI problem on trees with a single source. We propose a polynomial time greedy algorithm that guarantees $(1-1/e)$-approximation ratio on DAGs with a single source or a single sink.

Keywords: Graph augmentation · Approximation algorithms · Greedy algorithms · Submodularity · DAG · Trees · Dynamic programming

1 Introduction

In this paper, we consider the problem of improving the reachability of a graph. We approach the problem from a graph augmentation perspective, in which a set of non-existing edges are added to the graph to increase the overall number of reachable nodes. There are several recent possible application scenarios for this problem. For example, suggesting friends in a social network with the objective of increasing the spreading of information [2,5] or performing faster network simulations by reducing the convergence time of random walk processes [13,14]. Graph augmentation problems are also well known in traditional graph theory. In [7], Tarjan et al. consider the problems of adding a minimum (or minimum-weight) set of edges to a graph so as to satisfy a given connectivity condition,

https://doi.org/10.1007/978-3-030-14094-6_4

such as to make a directed graph strongly connected or to make an undirected graph bridge-connected or biconnected. They have already proved that some variants of augmentation problems are *NP*-Complete.

More recently, several optimization problems related to graph augmentation have been addressed. Demaine and Zadimoghaddam [6] study the problem of minimising the eccentricity of a graph by adding a limited number of new edges. A 4-approximation algorithm is introduced and it is proven that the problem is *NP*-hard to be approximated within a factor smaller than 3/2. The problem of minimising the average all-pairs shortest path distance – characteristic path length – of the whole graph has been studied by Papagelis in [14]. The author considers the problem of adding a small set of edges to minimise the characteristic path length, and proves that the problem is *NP*-Hard. He proposes a path screening technique to select the edges to be added. The problem of adding a small set of links in order to maximise the centrality of a given node in a network has been addressed for different centrality measures: page-rank [1,13], eccentricity [6], average distance [11], harmonic and betweenness centrality [3,4], some measures related to the number of paths passing through a given node [10].

In this paper, we study the problem of adding at most B edges to a directed graph in order to maximise the overall weighted number of reachable nodes, which we call the Maximum Connectivity Improvement (MCI) problem. We first show that we can focus on Directed Acyclic Graphs (DAG) without loss of generality (Sect. 2). Then, we focus on the complexity of the problem (Sect. 3) and we prove that the MCI problem is *NP*-Hard to approximate to within a factor greater than $1-\frac{1}{e}$. This result holds even if the DAG has a single source or a single sink. Moreover, the problem remains *NP*-complete if we further restrict to the unweighted case. In Sect. 4, we give a dynamic programming algorithm for the case in which the graph is a rooted tree, where the root is the only source node. In Sect. 5, we present a greedy algorithm which guarantees a $(1-1/e)$-approximation factor for the case in which the DAG has a single source or a single sink. We end with some concluding remarks in Sect. 6.

2 Preliminaries

Let $G = (V, E)$ be a directed graph. Each node $v \in V$ is associated with a weight $w_v \in \mathbb{N}_{\geq 0}$ and a profit $p_v \in \mathbb{N}_{\geq 0}$. Given a node $v \in V$, we denote by $R(v, G)$ the set of nodes that are reachable from v in G, that is $R(v, G) = \{u \in V : \exists \text{ path from } v \text{ to } u \text{ in } G\}$. Moreover, we denote by $p(R(v, G)) = \sum_{u \in R(v,G)} p_u$ the sum of the profits of the nodes reachable from v in G. In the rest of the paper, we also use the form $p(R(v, G) \setminus R(u, G)) = \sum_{u \in R(v,G) \setminus R(u,G)} p_u$ to denote the sum of the profits of the nodes in G that are reachable from v, but not from u. Note that, in the case $R(u, G) \subseteq R(v, G)$, it holds $p(R(v, G) \setminus R(u, G)) = p(R(v, G)) - p(R(u, G))$. Given a set S of edges not in E, we denote by $G(S)$ the graph augmented by adding the edges in S to G, i.e., $G(S) = (V, E \cup S)$. Let $R(v, G(S))$ and $p(R(v, G(S)))$ be, respectively, the set of nodes that are reachable from v in $G(S)$ and the sum of the profits of the nodes in $R(v, G(S))$. Note that, augmenting G the connectivity cannot be worse, and thus: $R(u, G) \subseteq R(u, G(S))$. Let $f(G) = \sum_{v \in V} w_v p(R(v, G))$ be a weighted measure of the connectivity of G. When weights and profits are unitary, $f(G)$ represents the overall number of connected pairs in G.

In this paper, we aim to augment G by adding a set S of edges of at most size B, i.e., $|S| \leq B$ and $B \in \mathbb{N}_{\geq 0}$, that maximises the weighted connectivity of $f(G(S))$. We call this problem the *Maximum Connectivity Improvement* (MCI) problem because maximising $f(G(S))$ is the same as to maximise $f(G(S)) - f(G)$.

From now on, for simplicity, we omit from the notations the original graph G. So, we simply use $R(v)$ and $R(v, S)$ to denote $R(v, G)$ and $R(v, G(S))$, respectively. Similarly, we simply denote with f and $f(S)$ the value of the weighted connectivity in G and in $G(S)$, respectively.

At first, we will show how to transform any directed graph G with cycles into a Directed Acyclic Graph (DAG) $G' = (V', E')$ and how to transform any solution for G' into a feasible solution for G.

Graph $G' = (V', E')$ has as many nodes as the number of strongly connected components of G. Specifically, G' selects one representative node for each strongly connected component of G and G' adds one directed edge between two nodes u' and v' of G' if there is a directed edge in G connecting any vertex of the strongly connected component represented by u' with any vertex of the strongly connected component represented by v'. Graph G' is called *condensation* of G and can be computed in $O(|V| + |E|)$ time by using Tarjan's algorithm which consists in performing a DFS visit [16].

The weight and the profit of a node v' in G' is given by the sum of the weight and profit of the nodes of G that belong to the strongly connected component $C_{v'}$ that is represented by v', i.e., $w_{v'} = \sum_{v \in C_{v'}} w_v$ and $p_{v'} = \sum_{v \in C_{v'}} p_v$.

Since the condensation preserves the connectivity of G, the following lemma can be proved:

Lemma 1. *Given a graph G and its condensation G', it yields: $f(G') = f(G)$.*

Proof. See Appendix.

□

Given a solution S' for the MCI problem in G', we can build a solution S with the same value for the MCI problem in G as follows: for each edge (u', v') in S', we add an edge (u, v) in S, where u and v are two arbitrary nodes in the connected component corresponding to u' and v', respectively.

This derives from the fact that applying the condensation algorithm to $G' \cup S'$ or to $G \cup S$ we obtain the same condensed graph, say G''. From Lemma 1, we can conclude that $f(G' \cup S') = f(G'') = f(G \cup S)$.

Observe that if we add an edge e within the same strongly connected component in G, we do not add any edge to G'. Since the condensation G'' of $(G \cup \{e\})$ is the same as G', we have $f(G \cup \{e\}) = f(G') = f(G)$. As a consequence, in the remainder of the paper, we will assume that the graph is a DAG.

Given a DAG, a node with no incoming edges is called a *source*, while a node with no outgoing edges is called a *sink*. The next lemma allows us to focus on solutions that contain only edges connecting sink nodes to source nodes.

Lemma 2. *Let S be a solution to the MCI problem, then there exists a solution S' such that $|S| = |S'|$, $f(S) \leq f(S')$, and all edges in S' connect sink nodes to source nodes.*

Proof. We show how to modify any solution S in order to find a solution S' with properties of the statement. To obtain S', we start from S and we repeatedly apply the following modifications to each edge (u, v) of S such that u is not a sink or v is not a source: (1) If u is not a sink then there exists a path from u to some sink u' and we swap edge (u, v) with edge (u', v). The objective function does not decrease and increases at least by the sum of the weights on a path from u to u'. Namely, after adding the edge (u', v), any node z on the path from u to u' will now reach v passing through u'. Note that the objective function will not decrease and, instead, may increase due to the fact that the nodes z now are able to reach the node v. (2) If v is not a source then there exists a path from a source v' to v and we swap edge (u, v) with edge (u, v'). The objective function does not decrease and increases at least by the number of nodes in a path from v' to v multiplied by w_u. Note that in both cases the gain of a node on the path we are extending can be zero if it was already able to reach the source/sink from another edge in the solution. □

3 NP-Hardness and Hardness of Approximation

In this section, we first show that the MCI problem is *NP*-Complete, even in the case in which all the weights and profits are unitary and the graph contains a single sink node or a single source node. Then, we show that it is *NP*-hard to approximate MCI to within a factor greater than $1 - \frac{1}{e}$. This last result holds also in the case of graphs with a single sink node or a single source node, but not in the case of unitary weights.

Theorem 1. *MCI is NP-Complete, even in the case in which all the weights and profits are unitary and the graph contains a single sink node or a single source node.*

Proof. We consider the decision version of MCI in which all the weights and profits are unitary (i.e., $w_v = p_v = 1$): Given a directed graph $G = (V, E)$ and two integers $M, B \in \mathbb{N}_{\geq 0}$, the goal is to find a set of additional edges $S \subseteq (V \times V) \setminus E$ such that $f(S) \geq M$ and $|S| = B$. The problem is in *NP* since it can be checked in polynomial time if a set of nodes S is such that $f(S) \geq M$ and $|S| = B$. We reduce from the Set Cover (SC) problem which is known to be *NP*-Complete [9]. Consider an instance of the SC problem $I_{SC} = (X, F, k)$ defined by a collection of subsets $F = \{S_1, \ldots, S_m\}$ for a ground set of items $X = \{x_1, \ldots, x_n\}$. The problem is to decide whether there exist k subsets whose union is equal to X. We define a corresponding instance $I_{MCI} = (G, M, B)$ of MCI as follows: (1) $B = k$; (2) $G = (V, E)$, where $V = \{v_{x_j} \mid x_j \in X\} \cup \{v_{S_i} \mid S_i \in F\} \cup \{v\}$ and $E = \{(v_{S_i}, v_{x_j}) \mid x_j \in S_i\} \cup \{(v_{x_j}, v) \mid x_j \in X\}$; (3) $M = (n+1+B)^2 + (m-B)(n+B+2)$.

See Fig. 1 (left, top) for an example. Note that G is a DAG. By Lemma 2, we can assume that any solution S of MCI contains only edges (v, v_{S_i}) for some $S_i \in F$. In fact, v is the only sink node and v_{S_i} are the only source nodes. Assume that there exists a set cover F', then we define a solution S to the MCI instance as $S = \{(v, v_{S_i}) \mid S_i \in F'\}$. It is easy to show that $f(S) = M$ and $|S| = k = B$. Indeed, all the nodes in G can reach: node v, all the nodes v_{x_j} (since F' is a set cover), and all the nodes v_{S_i} such that $S_i \in F'$. Moreover, each node v_{S_i} such that $S_i \notin F'$ can reach itself. Therefore there are $n+B+1$ nodes that reach $n+B+1$ nodes and $m-B$ that reach $n+B+2$ nodes, that is $f(S) = M$. On the other hand, assume that there exists a solution for MCI then S is in the form $\{(v, v_{S_i}) \mid S_i \in F\}$ and we define a solution for the set cover as $F' = \{S_i \mid (v, v_{S_i}) \in S\}$. We show that F' is a set cover. By contradiction, if we assume that F' is not a set cover and it cover only $n' < n$ elements of X then $f(S) = (n'+B+1)^2 + (n-n'+m-B)(n'+B+2) < M$. Note that in the above reduction, the graph G has a single sink node. We can prove the NP-hardness of the case of graphs with a single source node by using the same arguments on an instance of MCI made of the inverse graph of G, $M = (B+n+1)(n+m+1)+m-B$, and $B = k$ (see Fig. 1 (left, bottom) for an example). □

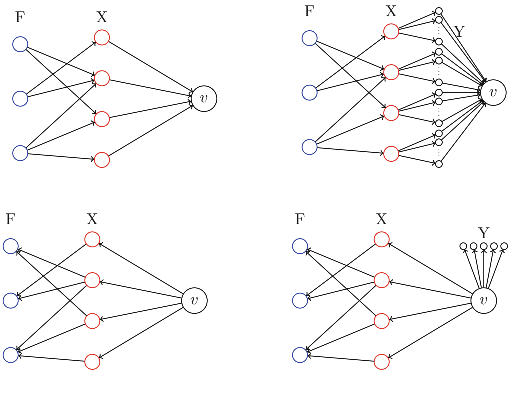


Fig. 1. (left) Example of reduction from SC to MCI used in Theorem 1. (right) Example of reduction from MC to MCI used in Theorem 2. (top) Single sink. (Bottom) single source.

Theorem 2. *MCI is NP-hard to approximate to within a factor* $1 - \frac{1}{e} + \epsilon$, *for any* $\epsilon > 0$, *even if graph contains a single sink node or a single source node.*

Proof. We give two approximation factor preserving reductions from the Maximum coverage problem (MC), which is known to be *NP*-hard to approximate to within a factor greater than $1 - \frac{1}{e}$ [8].

The MC problem is defined as follows: given a ground set of items $X = \{x_1, \ldots, x_n\}$, a collection of subsets $F = \{S_1, \ldots, S_m\}$ of subsets of X, and an integer k, find k sets in $\mathcal{F}$ that maximise the cardinality of their union.

We first focus on the single sink problem. Given an instance of the MC problem $I_{MC} = (X, F, k)$ we define an instance of the (maximisation) MCI problem $I_{MCI} = (G, k)$ similar to the one used in Theorem 1, but where we modify the weights and add Y paths of one node between each v_{x_j} and v, where Y is an arbitrarily high number (polynomial in $n + m$).

In detail I_{MCI} is defined as follows: (1) $B = k$; (2) $G = (V, E)$, where $V = \{v_{x_j} \mid x_j \in X\} \cup \{v_{S_i} \mid S_i \in F\} \cup \{v^l_{x_j} \mid x_j \in X \text{ and } l = 1, \ldots, Y\} \cup \{v\}$ and $E = \{(v_{S_i}, v_{x_j}) \mid x_j \in S_i\} \cup \{(v_{x_j}, v^l_{x_j}) \mid x_j \in X, l = 1, \ldots, Y\} \cup \{(v^l_{x_j}, v) \mid x_j \in X, l = 1, \ldots, Y\}$; (3) $w(v) = 1$ and $w(u) = 0$, for each $u \in V \setminus \{v\}$; (4) $p_v = 1$ for any node $v \in V$.

See Fig. 1 (right, top) for an example. We first show that there exists a solution $F' \subseteq F$ to I_{MC} that covers n' elements of X if and only if there exists a solution S to I_{MCI} such that $f(S) = n'(Y + 1) + B + 1$. Moreover, we can compute F' from S and vice versa in polynomial time. Indeed, given F', we define S as $S = \{(v, v_{S_i}) \mid S_i \in F'\}$. We can verify that $f(S) = n'(Y+1)+B+1$ and $|S| = k = B$. Indeed, only node v as a weight different from 0, and then $f(S) = R(v, S) = n'(Y + 1) + B + 1$, since v can reach the $n'(Y + 1)$ nodes v_{x_j} corresponding to the items x_j covered by F', the B nodes v_{S_i} it is connected to, and itself. On the other hand, given a solution S to I_{MCI}, by Lemma 2 we can assume that it has only edges from v to nodes v_{S_i}. Let n' be the number of nodes v_{x_j} in $R(v, S)$, then $f(S) = n'(Y+1)+B+1$ and $F' = \{S_i \mid (v, v_{S_i}) \in S\}$ covers n' elements in X.

If $OPT(I_{MC})$ and $OPT(I_{MCI})$ denote the optimum value for I_{MC} and I_{MCI}, respectively, then $OPT(I_{MCI}) \geq OPT(I_{MC})(Y + 1) + B + 1 \geq Y \cdot OPT(I_{MC})$. Moreover, given the above definition of S and F', then for any $\epsilon' > 0$ there exists a value of $Y = O(poly(n + m))$ such that $f(S) \leq (n' + \epsilon')Y$.

Let us assume that there exists a polynomial-time algorithm that guarantees an α approximation for I_{MCI}, then we can compute a solution S such that $f(S) \geq \alpha OPT(I_{MCI})$. It follows that:

$$\alpha Y \cdot OPT(I_{MC}) \leq \alpha OPT(I_{MCI}) \leq f(S) \leq (n' + \epsilon')Y,$$

Where n' is the number of nodes covered by the solution F' to MC obtained from S. Therefore we obtained an algorithm that approximates the MC problem with a factor α (up to lower order terms). Since it is *NP*-hard to approximate to within a factor greater than $1 - \frac{1}{e}$ [8], then the statement follows.

Let us now focus on the single source case. Given I_{MC}, we define $I_{MCI} = (G, B)$ as follows: (1) $B = k$; (2) $G = (V, E)$, where $V = \{v_{x_j} \mid x_j \in X\} \cup \{v_{S_i} \mid S_i \in F\} \cup \{v\} \cup \{v^l \mid l = 1, \ldots Y\}$ and $E = \{(v_{x_j}, v_{S_i}) \mid x_j \in S_i\} \cup \{(v, v_{x_j}) \mid x_j \in X\} \cup \{(v, v^l) \mid l = 1, \ldots Y\}$; (3) $w(v_{x_j}) = 1$, for each $x_j \in X$ and

$w(u) = 0$, for each $u \in V \setminus \{v_{x_j} \mid x_j \in X\}$; (4) $p_v = 1$ for any node $v \in V$. Where Y is an arbitrarily high polynomial value in $m + n$. See Fig. 1 (right, bottom) for an example. We use similar arguments as above. In detail, there exists a solution $F' \subseteq F$ to I_{MC} that covers n' elements of X if and only if there exists a solution S to I_{MCI} such that $f(S) = n'(n + m - m' + Y) + n(m' + 1)$, where m' is the number of sets in F that do not cover any of the n' elements covered by F'. Moreover, we can compute F' from S and vice versa in polynomial time. Given F', we define S as $S = \{(v, v_{S_i}) \mid S_i \in F'\}$ and we can verify that $f(S) = \sum_{x_j \in X} p(R(v_{x_j}, S)) = \sum_{x_j \in X} |R(v_{x_j}, S)| = n'(n+m+Y+1)+(n-n')(m'+1) = n'(n + m - m' + Y) + n(m' + 1)$ and $|S| = B$. Given S, if n' is the number of nodes v_{x_j} such that $v \in R(v_{x_j}, S)$, then $f(S) = n'(n + m - m' + Y) + n(m' + 1)$ and $F' = \{S_i \mid (v_{S_i,v}) \in S\}$ covers n' elements in X.

As above, we can show that $OPT(I_{MCI}) \geq Y \cdot OPT(I_{MC})$ and that there exists a value of $Y = O(poly(n+m))$ such that $f(S) \leq (n' + \epsilon')Y$, for any $\epsilon' > 0$. Then, the statement follows by using the same arguments as above. $\square$

4 Polynomial-Time Algorithm for Trees

In this section, we focus on the case of directed weighted rooted trees in which the root of the tree is the only source node and all the edges are directed towards the leaves. We give a polynomial time algorithm based on dynamic programming that requires $O(|V|B^2)$ time and $O(|V|B)$ space. By Lemma 2, we can focus on edges that connect leaves to the root. We first assume that the tree is binary and give an algorithm to solve this special case, then we show how to transform any tree into a binary tree in such a way that each solution for the transformed instance has the same value as the corresponding solution in the original instance. The algorithm for binary trees requires $O(|V|B^2)$ time and $O(|V|B)$ space while the transformation requires $O(|V|)$ time and $O(|V|)$ space.

4.1 Binary Trees

We are given a directed weighted binary tree $T = (V, E)$, where all the edges are directed towards the leaves, the root $r \in V$ is the only source node, and $w : V \to \mathbb{N}_{\geq 0}$, $p : V \to \mathbb{N}_{\geq 0}$. Let us denote by $\psi(v)$ (left child) and $\delta(v)$ (right child) the children of node $v \in T$, moreover, we denote as $T(v)$ the sub-tree rooted at v. In the following, we introduce our dynamic-programming algorithm to solve the MCI problem starting from the leaves of T. Given a node v, let S_v be a solution that connects some leaves of $T(v)$ to r. The *gain* of solution S_v in $T(v)$ is the increase in weighted reachability of all the nodes in $T(v)$, that is the gain of S_v in $T(v)$ is equal to $\sum_{u \in T(v)} w_u(p(T(u, S_v)) - p(T(u)))$. For each node v and for each budget $b = 0, 1, \ldots, B$, the algorithm computes a solution that connects b leaves of $T(v)$ to r and maximises the gain in $T(v)$. We define $g(v, b)$ as the maximum gain in $T(v)$ achievable by adding at most b edges from b leaves of $T(v)$ to node r. Note that $g(v, 1) \leq g(v, 2) \leq \ldots \leq g(v, b)$. We now show how to compute $g(v, b)$ for each node v and for each budget $b = 0, 1, \ldots, B$ by

Algorithm 1. Dynamic programming algorithm for MCI

```
    Input: $T = (V, E)$, $B$, $w_v$, $p_v$ $\forall v \in V$
    Output: Set $S$ of edges
 1: for each node $v$ do
 2:     $g(v, 0) := 0$;
 3:     $S_{(v,b)} := \emptyset$;
 4: for each leaf $v$ and budget $b$, with $\{1, \dots, B\}$ do
 5:     $g(v, b) := w_v \cdot (p(T(r)) - p(T(v)))$;
 6:     $S_{(v,b)} := \{(v, r)\}$;
 7: for each node $v$ in post-ordering do
 8:     for $b \in 1, \dots, B$ do
 9:         $g(v, b) := \max_{\substack{b_l, b_r \in \{0,\dots,b\},\\ b_l + b_r = b}} \{g(\psi(v), b_l) + g(\delta(v), b_r)\} + w_v \cdot (p(T(r)) - p(T(v)))$;
    ▷ Let $b_l$ and $b_r$ the budgets that maximise Line 9,
10:         $S_{(v,b)} := S_{(\psi(v),b_l)} \cup S_{(\delta(v),b_r)}$;
11: $S := S_{(r,B)}$;
```

using dynamic programming. For each leaf $v \in T$ and for each $b = 1, 2, \dots, B$, $g(v, b) = w_v \cdot (p(T(r)) - p(T(v)))$, that is, the sum of profits p of the new nodes that v can reach thanks to the new edge (v, r). Moreover $g(v, 0) = 0$ for each $v \in V$. Then, the algorithm visits T in post order. For each internal node v we compute $g(v, b)$ by using the solutions of its sub-trees, i.e., $T(\psi(v))$ and $T(\delta(v))$. Let us assume that we have computed $g(\psi(v), b)$ and $g(\delta(v), b)$, for each $b = 0, 1, \dots, B$. Note that if a solution adds an edge between any leaf of $T(v)$ and r, then the gain of node v is $w_v(p(T(r)) - p(T(v)))$ since v will now reach all the nodes in T. This gain is independent of the number of edges that are added from the leaves of $T(v)$ to r. In fact, given $g(\psi(v), b_l)$ be the maximum gain for $T(\psi(v))$ and budget $b_l \in \{1, 2, \dots, B\}$, then the gain in $T(v)$ of a solution that connects b_l leaves of $T(\psi(v))$ to r is equal to $g(\psi(v), b_l) + w_v \cdot (p(T(r)) - p(T(v)))$. Similarly the gain in $T(v)$ of the solution that connects, for some $b_r \in \{1, 2, \dots, B\}$, b_r leaves of $T(\delta(v))$ to r is equal to $g(\delta(v), b_l) + w_v \cdot p(T(r)) - p(T(v))$.

Then, to compute $g(v, b)$ once we have decided how many edges to add in $\psi(v)$ and how many edges in $\delta(v)$, we increase the reachability function of the same quantity, i.e., $w_v \cdot (p(T(r)) - p(T(v)))$.

Hence, $g(v, b)$ is given by the combination of b_l and b_r such that $b_l + b_r = b$ that maximises the sum $g(\psi(v), b_l) + g(\delta(v), b_r) + w_v \cdot (p(T(r)) - p(T(v)))$.

Precisely:

$$g(v, b) = \max_{\substack{b_l, b_r \in \{0,\dots,b\}\\ b_l + b_r = b}} \{g(\psi(v), b_l) + g(\delta(v), b_r)\} + w_v \cdot (p(T(r)) - p(T(v))). \quad (1)$$

The optimal value of the problem $f(S) = g(r, B) + f(T)$, where $f(T)$ is the value of the objective function on T (i.e., when no edges have been added). The pseudocode of the algorithm is reported in Algorithm 1.

Theorem 3. *Algorithm 1 finds an optimal solution for MCI if the graph is a binary tree.*

Proof. Let us assume by contradiction that v and b are, respectively, the first node and the first budget for which Algorithm 1 computes a non-maximum gain at line 9 of Algorithm 1, that is $g(v, b) < g^*(v, b)$, where $g^*(v, b)$ is the maximum gain for tree $T(v)$ and budget b. Let S^* be an optimal solution that achieves $g^*(v, b)$ and let S_l^*, S_r^* be the edges in S^* that starts from leaves in $T(\psi(v))$ and $T(\delta(v))$, respectively. Let $b_l^* = |S_l^*|$ and $b_r^* = |S_r^*|$. Then, the gain of the optimal solution S^* is: $g^*(v, b) = g(\psi(v), b_l^*) + g(\delta(v), b_r^*) + w_v \cdot (p(T(r)) - p(T(v)))$.

Since by hypothesis $g(v, b)$ is the first time for which Algorithm 1 does not find the maximum gain and since the cost $(p(T(r))-p(T(v)))\cdot w_v$ does not depend on the edges selected in the left and right sub-trees of v, this implies that at line 9 Algorithm 1 must select $g(v, b) = g(\psi(v), b_l^*) + g(\delta(v), b_r^*) + (p(T(r)) - p(T(v))) \cdot w_v = g^*(v, b)$. Thus contradicting $g(v, b) < g^*(v, b)$. □

For each node v, the algorithm computes $B + 1$ values. From Eq. 1, it follows that Algorithm 1 takes $O(|V|B^2)$ time. Note that $B \in O(|V|)$ because we limit the new edges to be of the form leaf-root.

4.2 General Trees

We can transform a generic rooted tree $T = (V, E)$ into an equivalent binary tree $T' = (V \cup U, E')$, following a tree transformation proposed in [15] by adding at most $|V| - 3$ *dummy* node.

The transformation requires $O(|V|)$ time and space (see Appendix 1.C for a detailed description of the algorithm). The nodes in T' will have $w'_v = w_v$ and $p'_v = p_v$ if $v \in V$ and $w'_v = p'_v = 0$ for any dummy node. Note that $p(R(v, T')) = p(R(v, T))$ for any node v in T due to the fact that the added dummy nodes have $p'_v = 0$, moreover, dummy nodes do not increase the objective function because they have the weight set to zero, i.e., any dummy node v will have $w'_v \cdot p(R(v, T')) = 0$.

For each node $v \in T'$ and solution S to MCI in T', let $f'(S) = \sum_{v \in V} w_v p(T(v, S))$. It is easy to see that by applying Algorithm 1 to T' we will obtain an optimal solution with respect to f'. Moreover, for each solution S to T', $f'(S) = f(S)$. Note that a solution S for T' that connects leaves of T' to its root are feasible solution also for T since T and T' have the same root and leaves.

5 Polynomial-Time Algorithm for DAG with a Single Source or a Single Sink

In this section, we focus on the case of weighted DAG in which we have a single source node or a single sink node. We first describe our greedy algorithm to approximate MCI on DAGs with a single source. Then, we will show how to modify the algorithm for the case of DAGs with a single sink.

In the case of single source, by Lemma 2, we restrict our choices to the edges that connect sinks nodes to the source. Recall S is the set of edges added to G. With a little abuse of notation, we also use S to denote the set of sinks from which the edges in S start. The Greedy algorithm for MCI on DAGs with a single source (see Algorithm 2), starts from the empty solution, i.e., $S = \emptyset$ and repeatedly adds to S the edge e' that maximises the function $f(S \cup \{e'\})$. The edge e' is chosen from the set E' of edges (u, s), where u is a sink in V, not already inserted in S, and s is the single source in V (see lines 2 and 4).

To implement the Greedy Algorithm with single source, some preprocessing is required. First, for each node $v \in V$, we perform a DFS on G to compute $R(v)$ and $p(R(v))$. We store $p(R(v, S))$ in a vector ρ of size $|V|$. Every time a new edge is added to the solution S, each entry of ρ is updated in constant time because for each node v, $p(R(v, S))$ is either equal to $p(R(v))$ or $p(R(s))$, as we explain below. To compute the gain of adding the edge $e = (u, s)$, we need to find all the nodes $R^T(u, S)$ that reach u in $G(S)$. $R^T(u, S)$ is computed by performing a DFS starting from u on the reverse graph $G^T(S)$ of $G(S)$. Note that the reverse graph G^T of G is initially computed in a preprocessing phase in $O(|V| + |E|)$ time, and after every new edge is added to S, $G^T(S)$ is updated in constant time. Finally, to compute $f(S \cup \{e\})$ for $e = (u, s)$, observe that $f(S \cup \{e\}) = f(S) + \sum_{z \in R^T(u,S)} w_z(p(R(s)) - p(R(z, S)))$. After selecting the edge $e' = (u, s)$ that maximise $f(S \cup \{e'\})$, we update S and we set $p(R(z, S)) = p(R(s))$ for each node $z \in R^T(u, S)$ in vector ρ because z reaches s traversing the edge $e' = (u, s)$ and inherits the reachability of $R(s)$.

The Greedy algorithm with a single source requires $O(B|V||E|)$ time. Namely, for each edge $e = (u, s) \in E'$, to compute $f(S \cup \{e\})$ it is required $O(|E|)$ time to compute $R^T(u, S)$ on G^T, and $O(|V|)$ time to compute $\sum_{z \in R^T(u,S)} w_z(p(R(s)) - p(R(z, S)))$. Since there are $O(|V|)$ sinks, the computation of the maximum value at line 4 costs $O(|V||E|)$ time. Selected e', $O(|V|)$ time is spent to update ρ. Since at most B edges are added, the Greedy algorithm requires $O(B|V||E|)$.

In the case of a single sink, we use the same Greedy Algorithm 2 only substituting E' in Line 2 with the set of edges (d, v), where v is a source in V and d is the only sink in V (see line 2) by Lemma 2. However, the implementation in the case of DAGs with a single sink is slightly different. In fact, $R^T(d, S) = V$ because (by definition) all the nodes reach the single sink, but at the same time, the reachability of a node v does not assume only the values V or $R(v)$. As a consequence to compute $f(S \cup \{(d, v)\})$ it is not required to perform any DFS visit of G^T, but the vicinity of any other node depends on the set of added edges and has to be recomputed by performing a DFS on the augmented graph. Hence, $f(S \cup \{(d, v)\}) = f(S) + \sum_{z \in V} w_z \cdot p(R(v, S) \setminus R(d, S))$. Since for computing $f(S \cup \{(d, v)\})$, we must compute $|V|$ DFSs, the overall cost of the Greedy algorithm with a single sink increases by a factor of $|V|$ with respect to the case of DAG with a single source, thus becoming $O(B|V|^2|E|)$.

Observe that Algorithm 2 with a single source can also be used on trees in place of Algorithm 1. However, the complexity of the Greedy algorithm will be $O(|V|^2 B)$ that is greater than the complexity of Algorithm 1, which is $O(|V|B^2)$.

Algorithm 2. Greedy Algorithm for single source

```
    Input: DAG $G = (V, E)$, $B$, $w_v, p_v\ \forall v \in V$
    Output: set $S$ of edges
1: $S := \emptyset$;
2: $E' := \{e = (u, s) \mid u$ is sink and $s$ is the only source$\}$;
3: while $|S| \leq B$ do
4:     $e' := \arg\max_{e \in E' \setminus S} f(S \cup \{e\})$;
5:     $S := S \cup \{e'\}$;
```

To give a lower bound on the approximation ratio of Algorithm 2, we show that the objective function $f(S)$ is *monotone and submodular*[1]. This allows us to apply the result by Nemhauser et al. [12]:

Given a finite set N, an integer k', and a real-valued function z defined on the set of subsets of N, the problem of finding a set $S \subseteq N$ such that $|S| \leq k'$ and $z(S)$ is maximum can be $1 - \frac{1}{e}$ approximated by starting with the empty set, and repeatedly adding the element that gives the maximal marginal gain, if z is monotone and submodular. We recall that $f(S) = \sum_{v \in V} w_v p(R(v, S))$ and $w_v, p_v \in \mathbb{N}_{\geq 0}$, therefore in order to prove that $f(S)$ is monotone increasing and submodular, we show that $p(R(v, S))$ is monotone increasing and submodular, for each $v \in V$ and solution S, because a non-negative linear combination of a monotone submodular functions is also monotone and submodular.

Theorem 4. *Function $f(S)$ is monotone and submodular with respect to any feasible solution for MCI on DAGs with a single source.*

Proof. To prove that $f(S)$ is monotone, we prove that for each $v \in V$, $S \subseteq E'$, and $e = (t', s) \in E' \setminus S$, we have $p(R(v, S \cup \{e\})) \geq p(R(v, S))$. We first notice that for each $v \in V$ and solution S, if there exists an edge $(t, s) \in S$ such that $t \in R(v)$, then $p(R(v, S)) = p(R(s))$; otherwise, $p(R(v, S)) = p(R(v))$. The same holds for $p(R(v, S \cup \{e\}))$.

We analyse the following cases recalling that $e = (t', s)$:

- If there exists an edge $(t, s) \in S$ such that $t \in R(v)$, then, $p(R(v, S \cup \{e\})) = p(R(v, S)) = p(R(s))$;
- Otherwise,
  - If $t' \in R(v)$, then, $p(R(v, S \cup \{e\})) = p(R(v))$ and $p(R(v, S)) = p(R(v))$
  - If $t' \notin R(v)$, then, $p(R(v, S \cup \{e\})) = p(R(v, S)) = p(R(v))$

It follows that $p(R(v, S \cup \{e\})) \geq p(R(v, S))$.

To prove that $f(S)$ is submodular, we prove that for any node $v \in V$, any two solutions S, T of MCI such that $S \subseteq T$, and any edge $e = (t', s) \notin T$, where t' is a sink node, it holds:

$$p(R(v, S \cup \{e\})) - p(R(v, S)) \geq p(R(v, T \cup \{e\})) - p(R(v, T)). \qquad (2)$$

[1] For a ground set N, a function $z : 2^N \to \mathbb{R}$ is submodular if for any pair of sets $S \subseteq T \subseteq N$ and for any element $e \in N \setminus T$, $z(S \cup \{e\}) - z(S) \geq z(T \cup \{e\}) - z(T)$.

We analyse the following cases:

- If there exists an edge $(t, s) \in S$ such that $t \in R(v)$, then, $p(R(v, S \cup \{e\})) = p(R(v, S)) = p(R(v, T \cup \{e\})) = p(R(v, T)) = p(R(s))$;
- Otherwise,
  - If there exists $(t'', s) \in T$ such that $t'' \in R(v)$, then $p(R(v, S \cup \{e\})) - p(R(v, S)) \geq 0 = p(R(v, T \cup \{e\})) - p(R(v, T))$ because $p(R(v, T \cup \{e\})) = p(R(v, T)) = p(R(s))$;
  - If for each $(t'', s) \in T$, $t'' \notin R(v)$, then $p(R(v, S \cup \{e\})) = p(R(v, T \cup \{e\}))$ and $p(R(v, S)) = p(R(v, T))$.

In all the cases Inequality (2) holds. □

Theorem 5. *Function $f(S)$ is monotone and submodular with respect to any feasible solution for MCI on DAGs with a single sink.*

Proof. To prove that $f(S)$ is monotone, we show that for each $v \in V$, $S \subseteq E'$, and $e = (d, s') \in E' \setminus S$, we have $p(R(v, S \cup \{e\})) \geq p(R(v, S))$. We observe that

$$R(v, S) = R(v) \cup \bigcup_{(d,s_i) \in S} R(s_i)$$

$$R(v, S \cup \{e\}) = R(v) \cup \bigcup_{(d,s_i) \in S} R(s_i) \cup R(s') = R(v, S) \cup R(s') \tag{3}$$

Thus, $p(R(v, S)) \leq \sum_{u \in R(v,S) \cup R(s')} p_u$.

To prove that $f(S)$ is submodular, we prove that for any node $v \in V$, any two solutions S, T of MCI such that $S \subseteq T$, and any edge $e = (d, s') \notin T$, where s' is a source node:

$$p(R(v, S \cup \{e\})) - p(R(v, S)) \geq p(R(v, T \cup \{e\})) - p(R(v, T)). \tag{4}$$

We first make the following observations based on Eq. 3:

- $R(v, S \cup \{e\}) = R(v, S) \cup R(s') = R(v, S) \cup (R(s') \setminus R(v, S))$ and
- $R(v, T \cup \{e\}) = R(v, T) \cup R(s') = R(v, T) \cup (R(s') \setminus R(v, T))$

Then, $p(R(v, S \cup \{e\})) - p(R(v, S)) = p(R(s') \setminus R(v, S))$ and $p(R(v, T \cup \{e\})) - p(R(v, T)) = p(R(s') \setminus R(v, T))$.

Inequality (4) follows by observing that $R(v, S) \subseteq R(v, T)$. □

Corollary 1. *Algorithm 2 provides a $\left(1 - \frac{1}{e}\right)$-approximation for the MCI problem either on DAG with a single source or with a single sink.*

6 Conclusion and Future Works

In this paper, we first defined the maximum connectivity improvement problem, that is the problem of adding k edges to a directed graph in order to maximise the overall weighted number of reachable nodes. We proved that the MCI problem on DAGs with a single source or a single sink is *NP*-Complete and *NP*-Hard to approximate to within a factor greater than $1 - 1/e$. We proposed a polynomial time greedy algorithm that guarantees a $1 - 1/e$ approximation ratio on DAG with a single source or a single sink. For rooted trees, to solve the MCI problem on single source, we devised a polynomial time algorithm based on dynamic programming faster than the greedy algorithm.

As future works, we plan to extend our approach to general DAG, i.e., with multiple sources and multiple sinks. Another possible extension is to solve the MCI problem by considering the budgeted version of the problem in which each edge can be added at a different budget cost.

Appendix 1.A Omitted Proofs

Lemma 3. *Given a graph G and its condensation G', it yields: $f(G') = f(G)$.*

Proof. First, consider two nodes u and v that belong to the same strongly connected component $C_{v'}$ in G'. Clearly, $R(u,G) = R(v,G)$.

Moreover, it holds $p(R(v,G)) = p(R(v',G'))$ because $R(v',G')$ contains one node for each different strongly connected component in $R(u,G)$ and thus:

$$p(R(v',G')) = \sum_{u' \in R(v',G')} p_{u'} = \sum_{u' \in R(v',G')} \sum_{u \in C_{u'}} p(u) = \sum_{u \in R(v,G)} p(u) = p(R(v,G))$$

Denoted $C_{v'}$ the strongly connected component represented by v', we have:

$$\begin{aligned}
f(G') &= \sum_{\mathbf{v}' \in V'} w_{\mathbf{v}'} p(R(\mathbf{v}', G')) \\
&= \sum_{\mathbf{v}' \in V'} w_{\mathbf{v}'} \left(\sum_{u' \in R(\mathbf{v}',G')} \sum_{u \in C_{u'}} p(u) \right) \\
&= \sum_{\mathbf{v}' : C_{\mathbf{v}'} \in \mathcal{C}} w_{\mathbf{v}'} \left(\sum_{u' \in R(\mathbf{v}',G')} \sum_{u \in C_{u'}} p(u) \right) \\
&= \sum_{\mathbf{v}' : C_{\mathbf{v}'} \in \mathcal{C}} \left(\sum_{v \in C_{\mathbf{v}'}} w_v \right) \left(\sum_{u' \in R(\mathbf{v}',G')} \sum_{u \in C_{u'}} p(u) \right) \\
&= \sum_{\mathbf{v}' : C_{v'} \in \mathcal{C}} \sum_{v \in C_{\mathbf{v}'}} \left(w_v \left(\sum_{u' \in R(\mathbf{v}',G')} \sum_{u \in C_{u'}} p(u) \right) \right) \\
&= \sum_{\mathbf{v}' : C_{\mathbf{v}'} \in \mathcal{C}} \sum_{v \in C_{\mathbf{v}'}} \left(w_v \sum_{u \in R(v,G)} p(u) \right) \\
&= \sum_{\mathbf{v}' : C_{\mathbf{v}'} \in \mathcal{C}} \sum_{v \in C_{\mathbf{v}'}} w_v p(R(v,G)) \\
&= \sum_{v \in V} w_v p(R(v,G)) = f(G)
\end{aligned}$$

□

Appendix 1.B Omitted Images

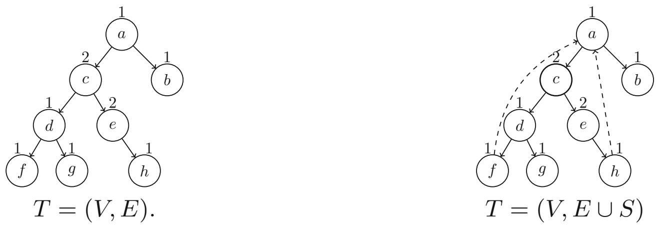


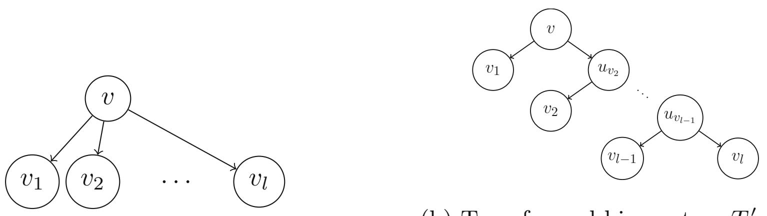
Fig. 2. Example of Algorithm 1. Consider the node c with $w_c = 2$, $p_v = 1\ \forall v \in V$ and $B = 2$. We have: $g(d, 2) = 19$, $g(d, 1) = 12$, $g(e, 1) = 7 + 2(6) = 19$. Therefore $g(c, 2) = g(e, 1) + g(d, 1) + w_c \cdot (p(T(r)) - p(T(v))) = 35$.

Appendix 1.C Generic Trees Algorithm

Given a generic rooted tree $T = (V, E)$, let us transform it into a rooted binary tree $T' = (V \cup U, E')$ with weights w', p' by adding *dummy* nodes U as follows:

1. Let the root r of T be the root of T'.
2. For each non-leaf node v, let $v_1, v_2, \ldots v_l$ be the children of v:
   (a) Add edge (v, v_1) to E';
   (b) If $l = 2$ add (v, v_2) to E';
   (c) If $l > 2$, add $l - 2$ dummy nodes $u_{v_2}, u_{v_3}, \ldots, u_{v_{l-2}}, u_{v_{l-1}}$
   (d) Add edge (v, u_{v_2}) and edges $(u_{v_i}, u_{v_{i+1}})$ to E', for each $2 \leq i \leq l - 2$;
   (e) Add edge (u_{v_i}, v_i) to E', for each $2 \leq i \leq l - 1$;
   (f) If $l > 2$, add edge $(u_{i_{l-1}}, v_l)$ to E'.
3. If $v \in V$, then $w'_v = w_v$, otherwise $w'_v = 0$ and $p'_v = p_v$, otherwise $p'_v = 0$.

See Fig. 3 for an example of the transformation.



(a) General tree $T = (V, E)$

(b) Transformed binary tree $T' = (V \cup U, E')$

Fig. 3. Example of transformation from general tree to binary tree

References

1. Avrachenkov, K., Litvak, N.: The effect of new links on Google PageRank. Stoc. Models **22**(2), 319–331 (2006)
2. Cordasco, G., et al.: Whom to befriend to influence people. CoRR abs/1611.08687 (2016). http://arxiv.org/abs/1611.08687
3. Crescenzi, P., D'Angelo, G., Severini, L., Velaj, Y.: Greedily improving our own centrality in a network. In: Bampis, E. (ed.) SEA 2015. LNCS, vol. 9125, pp. 43–55. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-20086-6_4
4. Crescenzi, P., D'Angelo, G., Severini, L., Velaj, Y.: Greedily improving our own closeness centrality in a network. TKDD **11**(1), 9:1–9:32 (2016)
5. D'Angelo, G., Severini, L., Velaj, Y.: Selecting nodes and buying links to maximize the information diffusion in a network. In: 42nd International Symposium on Mathematical Foundations of Computer Science, MFCS 2017, LIPIcs, vol. 83, pp. 75:1–75:14 (2017)
6. Demaine, E.D., Zadimoghaddam, M.: Minimizing the diameter of a network using shortcut edges. In: Kaplan, H. (ed.) SWAT 2010. LNCS, vol. 6139, pp. 420–431. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-13731-0_39
7. Eswaran, K.P., Tarjan, R.E.: Augmentation problems. SIAM J. Comput. **5**(4), 653–665 (1976)
8. Feige, U.: A threshold of ln n for approximating set cover. J. ACM **45**(4), 634–652 (1998)
9. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. W.H. Freeman and Company, New York (1979)
10. Ishakian, V., Erdös, D., Terzi, E., Bestavros, A.: A framework for the evaluation and management of network centrality. In: Proceedings of the 12th SIAM International Conference on Data Mining (SDM), pp. 427–438. SIAM (2012)
11. Meyerson, A., Tagiku, B.: Minimizing average shortest path distances via shortcut edge addition. In: Dinur, I., Jansen, K., Naor, J., Rolim, J. (eds.) APPROX/RANDOM -2009. LNCS, vol. 5687, pp. 272–285. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-03685-9_21
12. Nemhauser, G.L., Wolsey, L.A., Fisher, M.L.: An analysis of approximations for maximizing submodular set functions-i. Math. Program. **14**(1), 265–294 (1978)
13. Olsen, M., Viglas, A.: On the approximability of the link building problem. Theor. Comput. Sci. **518**, 96–116 (2014)
14. Papagelis, M.: Refining social graph connectivity via shortcut edge addition. ACM Trans. Knowl. Discovery Data (TKDD) **10**(2), 12 (2015)
15. Tamir, A.: An o(pn^2). algorithm for the p-median and related problems on tree graphs. Oper. Res. Lett. **19**(2), 59–64 (1996)
16. Tarjan, R.: Depth-first search and linear graph algorithms. SIAM J. Comput. **1**(2), 146–160 (1972)