# An efficient algorithm to add up-links to a rooted tree to obtain a minimum cost 2-connected graph

**Gabriel Morete de Azevedo**[1], **Yoshiko Wakabayashi**[1]

[1]Instituto de Matemática e Estatística – Universidade de São Paulo (USP)
Av. Prof. Luciano Gualberto, 1171 - Butantã, São Paulo - SP, 05508-090

{morete, yw}@ime.usp.br

***Abstract.*** *We present an efficient algorithm to solve a special case of the following node-connectivity augmentation problem. Given a tree $T = (V, E)$ and an additional set $L \subset \binom{V}{2}$ of edges, called links, $L \cap E = \varnothing$, each one with a rational nonnegative cost, find a minimum cost set of links $F \subseteq L$ such that $T + F$ is 2-connected. In general form, this problem is NP-hard. We focus on the up-link variation, where the tree $T$ has a root, and every link is an edge from a node to its ancestor. We present a linear formulation for this problem together with a proof of integrality and an efficient combinatorial algorithm for it.*

## 1. Introduction

Connectivity augmentation problems were introduced by [Eswaran and Tarjan 1976] and rapidly became a central topic in the design of *survivable networks*. In these problems, we are given a graph $G = (V, E)$ and we wish to augment by 1 the node-connectivity or the edge-connectivity of $G$ by economically adding new edges. The new edges, called *links*, are elements of a given set $L \subset \binom{V}{2}$ and have nonnegative costs, specified as a cost vector $c \in \mathbb{Q}^L_{\geq 0}$. In both variations (node and edge), even in the special case in which $G$ is a tree, these problems are NP-hard [Frederickson and Ja'Ja' 1981]. Here, we restrict our attention to this special case and denote the node version as NC-WTAP. For the edge-connectivity version, many approximation algorithms have been designed, with the best approximation guarantees obtained so far being $1.393$ for uniform costs by [Cecchetto et al. 2021], and $1.5 + \varepsilon$ for general costs by [Traub and Zenklusen 2022]. For NC-WTAP, the results are scarcer. For instances with uniform costs, [Nutov 2021] proposed a $1.91$-approximation, the first with a better-than-two guarantee; later, [Angelidakis et al. 2023] improved the approximation ratio to $1.892$. For general costs the 2-approximation of [Frederickson and Ja'Ja' 1981] is still the best known.

We focus on a special case of NC-WTAP, named here **Up-link NC-WTAP**. For this problem, the input is a quadruple $(T, L, r, c)$, where $T$ is a tree with root $r$, $L$ is a set of up-links and $c$ is the cost vector of the up-links in $L$. In this setting, a link $\ell = uv$ in $L$ is called an *up-link* if $u$ is an ancestor of $v$ (that is, $u$ is contained in the $vr$-path in $T$) or $v$ is an ancestor of $u$. (When $u$ is an ancestor of $v$, we also say that $v$ is a descendant of $u$.) The up-link version for edge-connectivity augmentation is defined analogously, and for it, a large body of literature is known [Adjiashvili 2017, Traub and Zenklusen 2021, Bamas et al. 2022], but the node-connectivity variant has remained unexplored.

There are many linear formulations for NC-WTAP, see [Grout, Logan 2020]. We present a novel formulation for Up-link NC-WTAP, along with a proof of integrality of the corresponding polyhedron. Moreover, we present a combinatorial algorithm, which is the new state-of-the-art result in terms of efficiency.

## 2. A linear formulation for Up-link NC-WTAP

First, we present a linear formulation for Up-link NC-WTAP. Let $(T = (V, E), L, r, c)$ be an instance of this problem. For each $X, Y \subseteq V$, define $\delta_L(X, Y) := \{xy \in L : x \in X, y \in Y\}$. When $Y = \overline{X}$, we simply write $\delta_L(X)$. For a graph $H$, let $\Pi(H)$ be the family of non-empty partitions of the connected components of $H$. For a partition $\mathcal{P} \in \Pi(H)$, let $|\mathcal{P}|$ be the number of parts (or classes) of $\mathcal{P}$. For a part $P \in \mathcal{P}$, we consider that $P$ is the set of vertices of the connected components of $H$ in $P$. For each $v \in V - r$, let $v^-$ be the direct ancestor of $v$ in $T$, let $\mathrm{N}^+(v)$ be the set of direct descendants of $v$, and let $T_v$ be the subtree containing $v$ and its descendants. We may assume that $r$ has a single direct descendant, denoted by $r^+$. The following is a relaxed linear formulation for NC-WTAP, requiring that, after removing any node, the augmented graph contains a spanning tree.

$$\begin{aligned} \text{Minimize} \quad & \sum_{\ell \in L} c(\ell)x(\ell) && \text{LP}_{\text{NC}}(T, L, r, c) \\ \text{subject to} \quad & \sum_{P \in \mathcal{P}} x(\delta_L(P)) \geq 2|\mathcal{P}| - 2, \quad \text{for } w \in V \text{ and } \mathcal{P} \in \Pi(T - w), && (1) \\ & x \geq 0. \end{aligned}$$

For the case of Up-link NC-WTAP, we may simplify the formulation above to:

$$\begin{aligned} \text{Minimize} \quad & \sum_{\ell \in L} c(\ell)x(\ell) && \text{LP}_{\text{UNC}}(T, L, r, c) \\ \text{subject to} \quad & x(\delta_L(T_v) - \delta_L(v^-)) \geq 1, \quad \text{for } v \in V - \{r, r^+\}, && (2) \\ & x \geq 0. \end{aligned}$$

Restriction (2) enforces that, when a node is removed, every resulting child subtree is connected to one of its ancestors. In the up-link setting, every constraint of $\text{LP}_{\text{NC}}(T, L, r, c)$ is satisfied by a solution of $\text{LP}_{\text{UNC}}(T, L, r, c)$. Indeed, for each $v \in V$, consider a restriction of type (1) arising from a partition $\{P_1, \ldots, P_z\} \in \Pi(T - v)$ with $T - T_v \subseteq P_1$. Let $x$ be a feasible solution of $\text{LP}_{\text{UNC}}(T, L, r, c)$. Then, by restriction (2), we have that $x(\delta_L(P_1, P_i)) \geq 1$ for $i = 2, \ldots, z$, since there are no links crossing subtrees of child vertices of $v$. Hence, $\sum_{i \in [z]} x(\delta_L(P_i)) \geq 2z - 2$, and therefore, $x$ satisfies (1). Moreover, we have the following theorem.

**Theorem 1.** *For every instance $(T, L, r, c)$ of Up-link* NC-WTAP*, the polyhedron associated with LP$_{\text{UNC}}(T, L, r, c)$ is integral.*

The proof of Theorem 1 follows the same steps as the proof of an equivalent theorem for edge-connectivity (Lemma 2.1 of [Adjiashvili 2017]).

## 3. A fast combinatorial algorithm for Up-link NC-WTAP

We use dynamic programming to solve Up-link NC-WTAP. For each $v \in V - r$, define $\mathrm{DP}(v)$ as the least cost set of links that, when added to $T$, ensures that node

$v$, its ancestors, and its descendants in $T$, are in a same component, even after removing any node from $V(T_{v^-})$. If $v$ is a leaf node, then $\mathrm{DP}(v)$ is a minimum cost link incident to $v$. For $uw \in L$, define $P_{uv}$ as the path from $u$ to $w$ in $T$. Define $L_v := \{uw \in L : u \in V(T_v), w \in V(T - T_{v^-})\}$. In general, the cost of $\mathrm{DP}(v)$ is given by:

$$c(\mathrm{DP}(v)) := \min_{\ell \in L_v} \{c(\ell) + c(\mathrm{DP}(R_{v,\ell}))\}, \tag{3}$$

where $R_{v,\ell}$ are the root nodes of $T_v - P_\ell$ and $\mathrm{DP}(R_{v,\ell}) = \cup_{w \in R_{v,\ell}} \mathrm{DP}(w)$. Note that a straightforward approach to solving this recurrence leads to an $O(|V|^2|L|)$ algorithm. We present an efficient way to compute (3). The algorithm computes the recurrence in a bottom-up approach, where each node is processed before all its ancestors, following a reverse topological sorting of $(T, r)$. To handle the recurrence efficiently, we store candidate links in a Fibonacci heap[1] (see Chapter 19 of [Cormen et al. 2009]). For each $v \in V$, the cost of using $\ell \in L_v$ to solve the recurrence for $v$ is given by

$$b(v, \ell) := c(\ell) + c(\mathrm{DP}(R_{v,\ell})).$$

Furthermore, we introduce $bh(h, \ell)$ to represent the key within each heap, $h$ being the index of a heap. Although $b(v, \ell)$ and $bh(f(v), \ell)$ may differ, for links $\ell_1, \ell_2 \in H_{f(v)}$ we enforce that $b(v, \ell_1) - b(v, \ell_2) = bh(f(v), \ell_1) - bh(f(v), \ell_2)$. Let $L_v^{in} \subseteq L$ be the set of links whose farthest endpoint from the root is $v$ and $L_v^{out} \subseteq L$ be the set of links whose closest endpoint to the root is $v$. Define the leaf set of $T$, excluding $r$, by $\xi(T)$.

For each $v \in \xi(T)$, we initialize each heap $H_v$ with the links $\ell \in L_v^{in}$ with key $bh(v, \ell) = c(\ell)$. Therefore, we have that $c(\mathrm{DP}(v)) = H_v.min()$. Moreover, we will not create any other heaps, each non-leaf node will be assigned to a heap used by a direct descendant. To achieve that, define a function $f : V \to \xi(T) \cup \{\varnothing\}$ which maps each node to its assigned heap (at first, $f(v) = v$ if $v \in \xi(T)$; and $f(v) = \varnothing$, otherwise).

Consider a non-leaf node $v \in V$. Since nodes are processed in reverse topological order, all descendant nodes will have been processed when solving for $v$. Let $u \in \mathrm{N}^+(v)$ and $\ell \in H_{f(u)} \cap L_v$. The cost change of using $\ell$ to solve the recurrence for $v$ compared to $u$ is given by

$$\Delta b(v, \ell) := b(v, \ell) - b(u, \ell) = c(\mathrm{DP}(\mathrm{N}^+(v))) - c(\mathrm{DP}(u)),$$

since $R_{v,\ell} - R_{u,\ell} = \mathrm{N}^+(v) - u$. We avoid updating the cost and copying each link in $L_v$ to prevent an $O(|V||L|)$ algorithm. Instead, to improve efficiency, we adopt a strategy commonly denoted by *small to large*, inspired by the analysis of the *disjoint union sets* structure (see Chapter 21 of [Cormen et al. 2009]). As the cost change of the links is uniform within each heap, we introduce a reduced cost $rc$ for each node so that $b(v, \ell) = bh(f(v), \ell) + rc(v)$. For a leaf node $v \in \xi(T)$, set $rc(v) = 0$. We build $H_{f(v)}$ as follows:

i) Let $u^* \in \mathrm{N}^+(v)$ be the direct descendant of $v$ associated with the largest heap. Assign $f(v) = f(u^*)$. Define the reduced cost of $v$ as

$$rc(v) := rc(u^*) + c(\mathrm{DP}(\mathrm{N}^+(v))) - c(\mathrm{DP}(u^*)),$$

which saves us from updating costs of links from $H_{f(u^*)}$ (*small to large* step).

[1] A Fibonacci heap supports the following operations. $H.insert()$ insert an element in $O(1)$ time, $H.min()$ returns the value of the minimum key in $O(1)$ time, $H.remove()$ removes an arbitrary element in $O(\log |H|)$ time, and traverse all elements in $O(|H|)$ time.

ii) For $u \in \mathrm{N}^+(v) - u^*$ and $\ell \in H_{f(u)}$, update $\ell$'s key to move it from $H_{f(u)}$ to $H_{f(v)}$. The change of the key assigned to $\ell$ is given by

$$\Delta bh(v,\ell) := \Delta b(v,\ell) + rc(u) - rc(v) = rc(u) - rc(u^*) - c(\mathrm{DP}(u)) + c(\mathrm{DP}(u^*)).$$

iii) Insert the links $\ell \in L_v^{in}$ in $H_{f(v)}$ with key $bh(f(v),\ell) = c(\ell) + c(\mathrm{DP}(\mathrm{N}^+(v))) - rc(v)$ and remove the links in $L_{v^-}^{out}$ from $H_{f(v)}$.

Thus, we obtain that that $c(\mathrm{DP}(v)) = H_{f(v)}.min() + rc(v)$ (see Algorithm 1).

Finally, `RevTopologicalSort(T,r)` can be implemented in linear time using a *depth first search* (DFS). Since computing the reduced costs and recovering the optimal value can be done in $O(|\mathrm{N}^+(v)|)$ for each $v \in V$, this sums up to a total of $O(|V|)$ operations. Also, since a link moves to a different heap only if the size of the resulting heap doubles, each link is moved at most $O(\log |L|)$ times, leading to a total complexity of $O(|L| \log |L|)$ for moving the links. Hence, the algorithm has a total time complexity of $O(|V| + |L| \log |L|)$. It is straightforward to recover the solution by saving the best links at each stage and using a DFS to build the solution. Finally, with little effort, one can adapt the algorithm above for the up-link edge-connectivity tree augmentation problem.

**Algorithm 1:** Algorithm for Up-link NC-WTAP

**Input:** An Up-link NC-WTAP instance $(T = (V,E), r, L, c)$.
**Output:** The cost of an optimal solution.
$rc(v) \leftarrow 0 \quad \forall v \in V$
$f(v) \leftarrow v \quad \forall v \in \xi(T)$
**for** $v$ ***in*** $RevTopologicalSort(T,r)$ **do**
    $u^* \leftarrow \arg\max_{u \in \mathrm{N}^+(v)} \{|H_{f(u)}|\}$
    $f(v) \leftarrow f(u^*)$
    $rc(v) \leftarrow rc(u^*) + c(\mathrm{DP}(\mathrm{N}^+(v))) - c(\mathrm{DP}(u^*))$
    **for** $u \in \mathrm{N}^+(v) - u^*$ **do**
        **for** $\ell \in H_{f(u)}$ **do**
            $H_{f(v)}.insert(\ell, bh(f(u),\ell) + rc(u) - rc(u^*) - c(\mathrm{DP}(u)) + c(\mathrm{DP}(u^*)))$
    **for** $\ell \in L_{v^-}^{out}$ **do**
        $H_{f(v)}.remove(\ell)$
    **for** $\ell \in L_v^{in}$ **do**
        $H_{f(v)}.insert(\ell, c(\ell) + c(\mathrm{DP}(\mathrm{N}^+(v))) - rc(v))$
    $c(\mathrm{DP}(v)) \leftarrow H_{f(v)}.min() + rc(v)$
**return** $c(\mathrm{DP}(r^+))$

## 4. Conclusion

It remains open whether there exists a linear-time algorithm for Up-link NC-WTAP. Another direction is to see whether there are applications analogous to the ones for the Up-link edge-connectivity tree augmentation problem.

## References


Adjiashvili, D. (2017). Beating approximation factor two for weighted tree augmentation with bounded costs. In *Proceedings of the 28th Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017*, pages 2384–2399. SIAM, Philadelphia, PA.

Angelidakis, H., Hyatt-Denesik, D., and Sanità, L. (2023). Node connectivity augmentation via iterative randomized rounding. *Math. Program.*, 199(1-2):995–1031.

Bamas, E., Drygala, M., and Svensson, O. (2022). A simple LP-based approximation algorithm for the matching augmentation problem. In *Proceedings of the 23rd International Conference on Integer Programming and Combinatorial Optimization, IPCO 2022*, volume 13265 of *Lecture Notes in Comput. Sci.*, pages 57–69. Springer, Cham.

Cecchetto, F., Traub, V., and Zenklusen, R. (2021). Bridging the gap between tree and connectivity augmentation: unified and stronger approaches. In *Proceedings of the 53rd Annual ACM-SIGACT Symposium on Theory of Computing, STOC 2021*, pages 370–383. ACM, New York.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. (2009). *Introduction to algorithms*. MIT Press, Cambridge, MA, third edition.

Eswaran, K. P. and Tarjan, R. E. (1976). Augmentation problems. *SIAM J. Comput.*, 5(4):653–665.

Frederickson, G. N. and Ja'Ja', J. (1981). Approximation algorithms for several graph augmentation problems. *SIAM J. Comput.*, 10(2):270–283.

Grout, Logan (2020). Augmenting trees to achieve 2-node-connectivity. Master's thesis, University of Waterloo.

Nutov, Z. (2021). 2-node-connectivity network design. In *Approximation and online algorithms*, volume 12806 of *Lecture Notes in Comput. Sci.*, pages 220–235. Springer, Cham.

Traub, V. and Zenklusen, R. (2021). A better-than-2 approximation for Weighted Tree Augmentation. In *Proceedings of the 62nd Annual Symposium on Foundations of Computer Science, FOCS 2021*, pages 1–12. IEEE, Los Alamitos, CA.

Traub, V. and Zenklusen, R. (2022). Local search for weighted tree augmentation and Steiner Tree. In *Proceedings of the 33rd Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2022*, pages 3253–3271. SIAM, Philadelphia, PA.