Optimal Fast-ReRoute: A Tool for Generating and Visualizing Arborescence-based Routes

Leon Okida, Elias P. Duarte Jr., and André Vignatti
Dept. Informatics – Universidade Federal do Paraná (UFPR)
Curitiba, Brazil
{laogoncalves, elias, vignatti}@inf.ufpr.br

***Abstract*—This work presents a tool that, given an input topology, uses Tarjan's algorithm to generate and visually display arborescence-based routes, addressing the lack of publicly available implementations of algorithms for generating disjoint arborescences and supporting the research of Fast-ReRoute (FRR) fault-tolerant routing techniques.**

***Index Terms*—arborescences, fault-tolerant routing, fast-reroute, tool**

I. Introduction

As communication networks, especially the Internet, become increasingly important, their uninterrupted operation is a critical requirement for numerous systems and applications [1]. Yet, network instability is a common issue [2]. Therefore, it is crucial to employ fault-tolerant routing strategies to handle failures, such as Fast-ReRoute (FRR) [3]. FRR proactively uses pre-computed backup routes when the primary route fails.

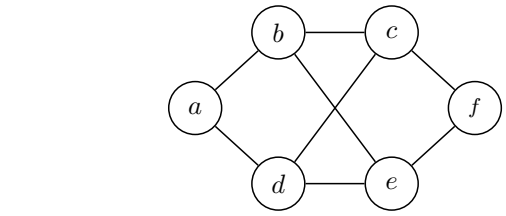
The effectiveness of FRR relies on the existence of disjoint routes, ensuring that a failure along one of the routes does not impact the others. A state of the art approach for constructing such routes is to decompose the network topology into edge-disjoint arborescences [4], [5]. An arborescence is a directed tree rooted at a single vertex, where all arcs point away from the root, and there is exactly one path from the root to every other vertex. By reversing the arc directions, paths from any vertex to the root can be obtained.

Edmonds's Theorem [6] states that a graph with edge connectivity λ can be decomposed into λ edge-disjoint arborescences rooted at the same vertex r. Consequently, this enables the construction of λ independent routes from any vertex to r.

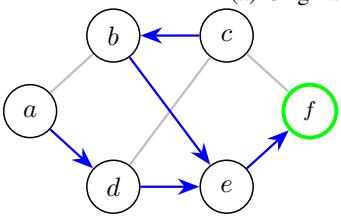
Arborescence-based routing employs one of the generated arborescences to determine the route from the origin to the destination. If a link fails on this route, it is possible to activate an alternative route by simply switching arborescences. The key advantage of this technique is that, by generating λ disjoint routes from any other vertex to r, routing is resilient even under $\lambda-1$ failures. This enables optimal fault-tolerant routing.

Figure 1 illustrates the process of generating arborescence-based routes. The topology shown in Subfigure (a) has edge-connectivity $\lambda = 2$, which enables its decomposition into two arborescences. Subfigures (b) and (c) present two disjoint arborescences that offer two independent routes from each vertex to f. We assume that an arc (u, v) is different from the arc (v, u).

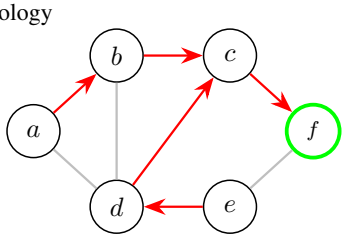
The process of building these arborescences is known as *arborescence packing*, and Tarjan's algorithm [7] is a classic method to solve it. It builds arborescences sequentially and incrementally by adding vertices under conditions that preserve the residual connectivity necessary to generate the subsequent structures. Section II describes the algorithm in depth.



(a) Original Topology



(b) Arborescence 1



(c) Arborescence 2

Fig. 1: Example of arborescence-based routes for destination f. The topology in (a) has edge-connectivity $\lambda = 2$, enabling its decomposition into two arborecences, (b) and (c).

Despite the critical role of arborescence packing in fault-tolerant routing, there is a scarcity of publicly available implementations of algorithms, hindering the research and study of FRR routing techniques. This work presents a tool to address this issue. Given a topology and an origin and destination pair of vertices, it computes arborescence-based routes and displays them visually. The design and functionality of the tool are presented in Section III.

II. Tarjan's Algorithm for Arborescence Packing

Tarjan's algorithm [7] is a classic method to decompose a graph into edge-disjoint arborescences. Given a directed graph $G = (V, E)$ with edge-connectivity λ and a vertex r, it builds λ r-rooted edge-disjoint arborescences sequentially. First, it uses Edmonds's Theorem [6] to evaluate if it is possible to

build the arborescences, independently of how the vertices are grouped around the root vertex:

$$|\delta^+(S)| \geq \lambda, \quad \forall S \subset V : r \in S, S \neq \emptyset, S \neq V. \tag{1}$$

In each iteration i, where $1 \leq i \leq \lambda$, Algorithm 1 builds the arborescence $A_i = (V_i, E_i)$ incrementally from the root r. Initiating with $V_i = \{r\}$ and $E_i = \emptyset$, the algorithm expands A_i until $V_i = V$. It tries to add new usable arcs, which were not used in previously generated arborescences, and adds an arc $e = (u, v)$ to the arborescence if Condition 2 holds:

$$|\delta^+_{G'}(S)| \geq \lambda - i + 1, \quad \forall S \subset V : u \in S, v \notin S, \tag{2}$$

where G' represents G, but without arcs already used in previously generated arborescences. This condition guarantees that if e is added to E_i, the remaining graph connectivity is sufficient to build the subsequent arborescences.

Algorithm 1 Tarjan's algorithm to generate λ edge-disjoint arborescences given a graph $G = (V, E)$ and a root vertex r

Condition 1: $|\delta^+(S)| \geq \lambda, \quad \forall S \subset V : r \in S, S \neq \emptyset, S \neq V$
Condition 2: $|\delta^+_{G'}(S)| \geq \lambda - i + 1, \quad \forall S \subset V : u \in S, v \notin S$

if Condition 1 is **false** for any S **then**
  return **error**: it is not possible to generate λ edge-disjoint arborescences
else
  for i:=1 **to** λ **do**
    $V_i := \{r\}, E_i := \emptyset$
    mark edges not used in other arborescences as usable
    while $V_i \neq V$ **do**
      find usable edge $e = (u, v)$, with $u \in V_i$ and $v \notin V_i$
      mark e as not usable
      if Condition 2 is **true** for e **then**
        $V_i := V_i \cup \{v\}$
        $E_i := E_i \cup \{e\}$
      end if
    end while
    $A_i := (V_i, E_i)$
  end for
end if

III. An Arborescence-Based Routing Visualization Tool

The proposed tool offers interactive graphical visualization of arborescence-based routes for any given topology. It uses our implementation of Tarjan's algorithm in Python using the NetworkX library [8] to compute arborescences. The graphical user interface was developed using the *Streamlit* and *PyVis* packages. The tool is available on the *Streamlit Share*[1] platform, and its source code, along with its complete documentation, is available on *GitHub*[2].

The user flow is as follows. First, the user uploads a text file containing a topology in the edge list format. Then, the user is prompted to select an origin and a destination. By clicking on the "Compute Routes" button, the tool displays the arborescence-based routes computed from the origin to the destination in the main panel, as illustrated in Figure 2.

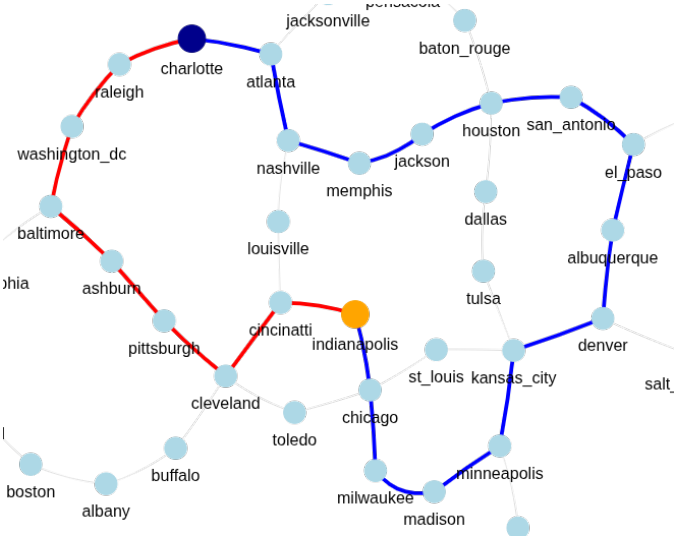


Fig. 2: Arborescence-based routes as computed by the tool between Charlotte and Indianapolis in the Internet2 topology.

IV. Conclusion

This work introduced a tool, developed in Python with the NetworkX package, that generates and visually displays arborescence-based routes for any given network topology and an origin and destination pair. Decomposing topologies into λ arborescences is a cutting-edge approach in Fast-ReRoute (FRR) fault-tolerant routing, enabling optimal resilience even against $\lambda - 1$ failures. By providing an accessible implementation of Tarjan's algorithm for generating disjoint arborescences, it addresses a significant gap in available tools to support the study and development of Fast-ReRoute (FRR) fault-tolerant routing techniques.

[1] https://arborescencebasedrouting.streamlit.app/
[2] https://github.com/leonokida/arborescence_based_routing

References

[1] M. Chiesa, A. Kamisiński, J. Rak, G. Rétvári, and S. Schmid, "A Survey of Fast-Recovery Mechanisms in Packet-Switched Networks," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 2, pp. 1253–1301, 2021.

[2] E. Duarte Jr, T. Garrett, L. Bona, R. Carmo, and A. Züge, "Finding stable cliques of PlanetLab nodes," in *2010 IEEE/IFIP International Conference on Dependable Systems & Networks (DSN)*, pp. 317–322, IEEE, 2010.

[3] L. Okida, E. Maverson, and E. Duarte Jr., "Fast Reroute with Highly Connected Routes Based on Maximum Flow Evaluation," *arXiv preprint arXiv:2410.10528*, 2024.

[4] M. Chiesa, I. Nikolaevskiy, S. Mitrović, A. Gurtov, A. Madry, M. Schapira, and S. Shenker, "On the Resiliency of Static Forwarding Tables," *IEEE/ACM Transactions on Networking*, vol. 25, no. 2, pp. 1133–1146, 2017.

[5] K.-T. Foerster, A. Kamisiński, Y.-A. Pignolet, S. Schmid, and G. Tredan, "Improved Fast Rerouting Using Postprocessing," *IEEE Transactions on Dependable and Secure Computing*, vol. 19, no. 1, pp. 537–550, 2022.

[6] J. Edmonds, "Edge-disjoint branchings," *Combinatorial algorithms*, pp. 91–96, 1973.

[7] R. E. Tarjan, "A good algorithm for edge-disjoint branching," *Information Processing Letters*, vol. 3, no. 2, pp. 51–53, 1974.

[8] A. A. Hagberg, D. A. Schult, and P. J. Swart, "Exploring Network Structure, Dynamics, and Function using NetworkX," in *Proceedings of the 7th Python in Science Conference*, pp. 11 – 15, 2008.