Efficient Computing of Disaster-Disjoint Paths: Greedy and Beyond

Balázs Vass, Erika Bérczi-Kovács, Péter Gyimesi, János Tapolcai

***Abstract*—In a network topology G, we say a set of st-paths are disaster-disjoint if no disaster strikes more than one path. In this poster, we explore the basic capabilities and limitations of greedy and more advanced algorithms for computing maximal collections of such paths in planar networks. An algorithm is *greedy* if it generates consecutive paths $P_1, P_2, \ldots$ according to a simple rule. In the simplest setting, the only rule is that P_{i+1} is the closest clockwise disaster-disjoint from P_i. We find that the simplest greedy may fail even when 1) G is planar, 2) each disaster region is connected, and 3) each node failure (apart from s and t) is considered possible. Adding a simple rule explained in [1] yields a correct polynomial-time algorithm for the above problem. Finally, we digest a recent related near-linear runtime algorithm of [2] solving a more general problem and discuss the underlying relations among the foundations of these algorithms.**

I. Introduction, Model, and Assumptions

The primary algorithmic problem for backbone network mechanisms that aim to preserve connectivity in the event of a disaster is to find disjoint paths between two nodes s and t in an undirected graph $G = (V, E)$. The most commonly employed approach for this task is to find either edge-disjoint or node-disjoint paths, which is particularly suitable for mechanisms designed to handle failures of a single network equipment. However, network failures can occur in the form of multi-point failures, in which a substantial physical area is affected by simultaneous equipment outages caused by catastrophic events such as earthquakes, hurricanes, tsunamis, tornadoes, and other similar occurrences [3]. These multi-point failures are often called regional failures or *regions* for brevity. The concept of Shared Risk Link Groups (SRLGs) is also in use [4], [5]. We assume the list of regions (or SRLGs) $\mathscr{R} \subseteq 2^E$ is also part of the input, which was already identified during the network design phase based on some historical data and exploration of network vulnerabilities. Two st-paths are $\mathscr{R}$-disjoint if there is no edge set in $\mathscr{R}$ intersecting both paths. We assume the network topology is planar, which allows the application of some modified greedy approaches [1], [6], [7]. Let the dual of G be denoted as G^*, which consists of vertices V^* and edges E^*. Each edge e in E corresponds to an edge in the dual graph G^*, which is denoted as e^*. Such an input graph can be stored in a rotation system [8], where the incident edges for every node are given in clockwise order. We assume that each disaster causing the outage has a connected destruction area. Consequently, the corresponding dual edges of each region $r \in \mathscr{R}$, form a connected subgraph in G^* [1]. It turns out, even with the above assumptions, finding the maximum number of disaster-disjoint st-paths (Problem 1) is $\mathscr{NP}$-hard [9]. Paper [1] showed that if paths are required to be interiorly node disjoint, the problem becomes polynomially solvable. In the case when node disjointness is not required, there are at most 2 more (crossing) disaster-disjoint paths, than the maximal number of non-crossing disaster-disjoint paths, and such non-crossing paths can be calculated in near-linear time [2, Thm. 1-3]. In other words, Problem 1 is efficiently additively 2-approximable. Here, intuitively speaking, paths are *non-crossing*, if they do not change their clockwise order on their way from s to t. In this study, we focus on Problem 1 and its node-disjoint or non-crossing variations.

Problem 1: Maximum number of disaster-disjoint st-paths

Input: A planar graph $G = (V, E)$, rotation system, nodes $s, t \in V$, disasters/regions $\mathscr{R} \subset 2^E$

Output: A maximum number of disaster-disjoint st-paths $P_1, P_2 \ldots, P_k$

II. Greedy Approaches

Simple Greedy: In the simplest version of greedy algorithms, we are given an st-path P_1. Then, for each $i \in \{2, 3, \ldots\}$, P_i is the nearest *clockwise disaster-disjoint* path to P_{i-1}. Here, clockwise disaster-disjoitness (defined in [1]) is just the usual disaster-disjointness if we have already found $k \geq 2$ disaster-disjoint paths, and a useful relaxation when searching for the first two of them. We have the following new contribution:

Theorem 1. *The Simple Greedy doesn't always find an optimal solution for Problem 1 even supposing node-disjointness.*

Proof: A counterexample is depicted in Fig. 1. More precisely, the input graph is drawn in Fig. 1a, while in the rest of the subfigures, t is drawn in multiple copies, for easier visualization. There are six disaster regions (depicted in Fig. 1b-Fig. 1d), in dashed-and-red, and dash-dotted-and-blue lines, respectively. Clearly, there are at most 3 disaster-disjoint paths. While paths Q_1, Q_2, Q_3 on Fig. 1b show a lucky run of the simple greedy, $P_1, P_2, P_3, P_{4,5}, P_6, P_1, \ldots$ on Fig. 1c-Fig. 1d show an infinite cycle of paths generated by the algorithm, among which there are no three, that are disaster-disjoints. ■

Balázs Vass and János Tapolcai are with Department of Telecommunication and Media Informatics, Faculty of Electrical Engineering and Informatics (VIK), Budapest University of Technology and Economics (BME) and HUN-REN-BME Information Systems Research Group and MTA-BME Future Internet Research Group. Balázs Vass is also affiliated to Faculty of Mathematics and Computer Science, Babeş-Bolyai University, Cluj Napoca, Romania. Contact them on {balazs.vass,tapolcai}@tmit.bme.hu.

Erika Bérczi-Kovács and Péter Gyimesi are with ELTE Eötvös Loránd University, Budapest, Hungary. Erika Bérczi-Kovács is also with HUN-REN–ELTE Egerváry Research Group on Combinatorial Optimization. Contact them on erika.berczi-kovacs@ttk.elte.hu, and peti1234@student.elte.hu.

This work was partly supported by Project nos. ANN_20 135606, FK 132524, K_23 146347, and ÚNKP-23-4-II-BME-345, which have been implemented with support from the National Research, Development and Innovation Fund of Hungary. This work is supported by the János Bolyai Research Scholarship of the Hungarian Academy of Science.

On the positive side, we claim without proof that the Simple Greedy finds the optimal *value* k^* for the maximum number of disaster-disjoint and node-disjoint paths (without the paths themselves) after a finite (possibly exponential) number of i iterations. That is, if, starting at P_1, the l^{th} newly generated path P_{l+1} is the first to equal P_1, and, in the meantime, the consecutive paths winded around S a number of w times, then we have $k^* = \lfloor l/w \rfloor$. E.g., on Fig. 1c-d, $k^* = 3$, $l = 6$, and $w = 2$.

Fixing the Simple Greedy: Dervish of [1], [10]: Fixing the simple greedy, algorithm *Dervish* of [1], [10] adds a second rule: when searching for the k^{th} disaster-disjoint path, it starts with disaster-disjoint paths $P_1, \ldots, P_{k-1}$, with $P_0 := P_{k-1}$. A new path P_l should be locally (non-strictly) clockwise to P_{l-k}. With this, in presence of the node failures, the Dervish is guaranteed to solve Problem 1 with node-disjoint paths in polynomial time, that is squared in $|V|$ in practice (cf. [10, Thm. 2]). On the example depicted on Fig. 1, when searching for the third disaster-disjoint path, starting from path pair P_1, P_2, the Dervish generates the following paths: $P_3^D = \{s, v_3, v_4, t\}$, $P_4^D = \{s, v_6, t\}$, $P_5^D = \{s, v_2, t\}$, $P_6^D = \{s, v_4, t\}$. Note that P_4^D, P_5^D, and P_6^D are pairwise node- and disaster-disjoint, thus the algorithm returns with these paths. Intuitively speaking, in this example, the key difference compared to the Simple Greedy is that the 4^{th} path is not let to have links that are locally anti-clockwise from path P_1. More in detail, P_4^D has to start out from s towards v_6 to be disaster-disjoint from P_3, and (unlike in the Simple Greedy) cannot lean anti-clockwise to v_5, because then it would be anti-clockwise from P_1.

III. An Efficient Alternative

While the Dervish was the first algorithm to solve Problem 1 with all node failures in polynomial time, it still had some drawbacks. First, it is painfully technical when searching for the first two disaster-disjoint paths or disproving their existence. Second, the Dervish generates new paths *explicitly*, and in order to prove maximality, Dervish needs to generate up to $\Theta(|V|)$ new paths, prohibiting it from having a near-linear runtime. Naturally arises the question whether there is a closer connection between the Dervish and its brand new efficient alternative [2]. We claim that the generation of each new path can be translated in [2] to some steps of the Bellman-Ford (B-F) algorithm in a newly introduced directed weighted graph, the so-called *regional dual graph* $G^*_{\mathscr{R}}$. Here, weights on arcs of $G^*_{\mathscr{R}}$ depend on the number k of supposedly existing non-crossing disaster-disjoint paths. The existence of a negative cycle is a witness of non-existence of such k paths. Otherwise, the result of the B-F algorithm encodes k different non-crossing disaster-disjoint paths.[1] As the key is computing some appropriate distances, the B-F can be substituted with more efficient algorithms. With this, [2] can solve Problem 1 with requiring non-crossing paths in $O\left(\log(k^*)\|\mathscr{R}\|^{\frac{3}{2}}\log(\|\mathscr{R}\|)\right)$ deterministic worst case time complexity, or with high probability in $O\left(\log(k^*)\|\mathscr{R}\|\log^9(\|\mathscr{R}\|)\right)$ expected time, where $\|\mathscr{R}\| := \sum_{R\in\mathscr{R}}|R|$. Also, this solution is a provably 2-additive approximation on Problem 1, where paths of the solution may cross. For details, please refer to [2].

IV. Conclusion

In this poster, we proved that the *Simple Greedy* algorithm for computing disaster-disjoint paths is incorrect. Here *Simple Greedy* is yielded by the elimination of the non-trivial rule for greedily generating the new paths the *Dervish* algorithm of [1]. Further, we claimed that the *Dervish* and recent efficient alternative [2] have strong connections in their foundations.

References

[1] B. Vass *et al.*, "Polynomial-time algorithm for the regional SRLG-disjoint paths problem," in *Proc. IEEE INFOCOM*, May 2022.

[2] E. Bérczi-Kovács *et al.*, "Efficient Algorithm for Region-Disjoint Survivable Routing in Backbone Networks," in *IEEE INFOCOM 2024*.

[3] S. Neumayer *et al.*, "Assessing the vulnerability of the fiber infrastructure to disasters," *IEEE/ACM Trans. Netw.*, vol. 19, 2011.

[4] D. Zhou *et al.*, "Survivability in optical networks," *IEEE network*, vol. 14, pp. 16–23, 2000.

[5] J. Tapolcai *et al.*, "List of shared risk link groups representing regional failures with limited size," in *IEEE INFOCOM*, Atlanta, USA, 2017.

[6] C. McDiarmid *et al.*, "Non-interfering dipaths in planar digraphs," 1991.

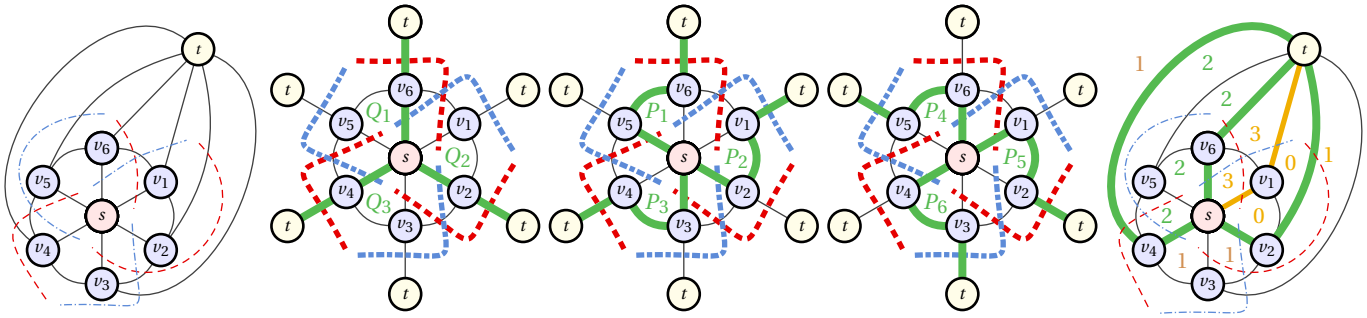
[7] Y. Kobayashi *et al.*, "Max-flow min-cut theorem and faster algorithms in a circular disk failure model," in *IEEE INFOCOM 2014*, April 2014.

[8] J. L. Gross *et al.*, "The topological theory of current graphs," *Journal of Combinatorial Theory, Series B*, vol. 17, pp. 218–233, 1974.

[9] D. Bienstock, "Some generalized max-flow min-cut problems in the plane," *Mathematics of Operations Research*, vol. 16, 1991.

[10] B. Vass *et al.*, "A Whirling Dervish: Polynomial-Time Algorithm for the Regional SRLG-disjoint Paths Problem," *IEEE/ACM ToN*, 2023.

[1] Fig. 1e depicts the result of the B-F on $G^*_{\mathscr{R}}$, for searching for the optimal number of $k = 3$ pahts. Here, distances were measured from face F_0 surrounded by edges sv_1, v_1v_2 and sv_2. Oversimplified, the distance $d(F_0, F_i) = j$, if in the dual G^*, the minimum number one has to change regions on a F_0F_i-path is j. In addition, crossing sv_1t clockwise costs $-k$, while crossing it anti-clockwise costs k. The 3 disaster-disjoint paths computed are the boundaries of faces with the same distances $\mod(k = 3)$.



(a) Graph G of our examples (b) Paths Q_1, Q_2, Q_3 (c) Paths P_1, P_2, P_3 (d) Paths P_4, P_5, P_6 (e) Distances computed in [2]

Fig. 1. Examples on the algorithms. While a) and e) shows the input graph, in the rest of the subfigures, for easier visualization, t is drawn in multiple copies. b) shows three disaster-disjoint paths, c) and d) combined show a cycle of paths the simple greedy generates, in which no 3 consecutive is disaster-disjoint. Paths traversing vertices s-v_1-t and s-v_2-t are *non-crossing*, while those traversing s-v_1-v_2-t and s-v_2-v_1-t are *crossing*.