Variants of the Min-Sum Link-Disjoint Paths Problem

Anteneh Beshir and Fernando Kuipers

Faculty of Electrical Engineering, Mathematics and Computer Science
Delft University of Technology, Mekelweg 4, 2628 CD Delft, The Netherlands
Email:{A.A.Beshir, F.A.Kuipers}@tudelft.nl

A survivable connection usually requires computing link-disjoint primary and backup paths. Finding a min-sum pair of link-disjoint paths whose total cost is minimized can be solved in polynomial time. However, adding extra requirements may render the problem NP-complete. In this paper, we study different variants of the min-sum link-disjoint paths problem. We examine the complexity of these problem variants and provide exact and heuristic algorithms for them.

1 Introduction

In networks, such as optical networks, that transport a tremendous amount of traffic, survivability, which is the ability to recover from failures, is indispensable. In order to prevent single-link failures, which are the most prevalent types of failures, it is necessary to establish connections on link-disjoint primary and backup paths between the source and destination nodes. The primary path is used during normal operations, while the backup path takes over during the failure of the primary path. There can be several objectives associated with finding link-disjoint paths. The most common and simpler one is the min-sum link-disjoint paths problem, which is finding a pair of link-disjoint paths whose combined cost is minimized. Depending on how frequently failures occur on the primary path, it may be desirable to minimize the cost of the primary (shorter) path (min-min problem) [7] or the backup (longer) path (min-max problem) [4]. In constrained routing, the costs or bandwidths of the primary and backup paths need to be bounded [3]. In load balancing, it may be necessary to find a pair of paths with the largest residual bandwidth so that heavily loaded links are avoided (shortest-widest problem) [5].
Among the aforementioned objectives, only the min-sum [6] and the shortest-widest [5] problems are polynomially solvable, while the others are NP-complete. We will investigate whether we can use these other objectives as secondary objectives to the min-sum link-disjoint problem. We show that the NP-complete secondary objectives turn the polynomially solvable min-sum problem to NP-complete min-sum problem variants. However, through simulations we show that due to the strongly reduced search space, exact algorithms can, in practice, solve the respective problem variants in a reasonable time.
In Section 2, a formal definition and the complexity of each problem variant is presented. In Section 3, we provide algorithms for these problem variants. In Section 4, we present our simulation results, and in Section 5, we give our conclusions.

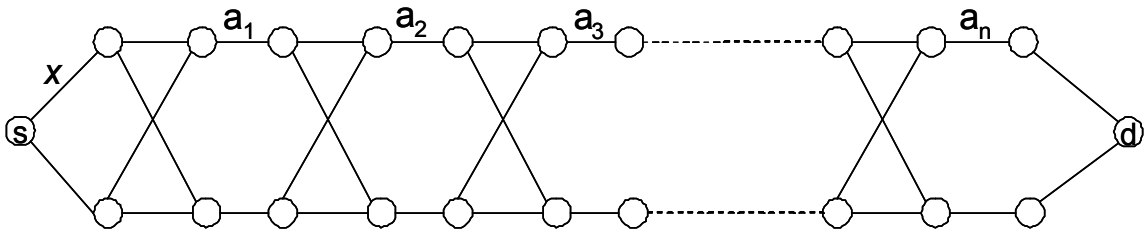

Figure 1: All links without labels have zero cost.

2 Problem Definition

Given a graph $G(\mathcal{N}, \mathcal{L})$, where $|\mathcal{N}| = N$ and $|\mathcal{L}| = L$, a cost $c(l)$ and a bandwidth $B(l)$ associated with each link $l \in \mathcal{L}$, a source node s and a destination node d, two bounds $\Delta_1 \geq 0$ and $\Delta_2 \geq 0$, find a pair of link-disjoint paths from s to d such that
Min-Sum Min-Min Link-Disjoint Paths Problem: The total cost of the pair of link-disjoint paths is minimized and if there is a tie, the cost of the shorter path is minimized.
Min-Sum Min-Max Link-Disjoint Paths Problem: The total cost of the pair of link-disjoint paths is minimized and if there is a tie, the cost of the longer path is minimized.
The Bounded Min-Sum Link-Disjoint Paths Problem: The total cost of the pair of link-disjoint paths is minimized, and then the cost of the shorter path should be less than or equal to Δ_1 and the cost of the longer path should be less than or equal to Δ_2.
The Widest Min-Sum Link-Disjoint Paths Problem: The total cost of the link-disjoint paths is minimized, and if there is a tie, the smallest bandwidth of all links in the two paths is maximized.
The min-sum min-min link-disjoint paths problem is proven to be NP-complete by Yang *et al* [8].

Theorem 1 *The min-sum min-max link-disjoint paths problem is NP-complete.*

To prove this theorem, we make use of the NP-complete partition problem [2], which is defined as follows.
The partition problem: Given a set of values $a_i \in A$, $a_i \geq 0$ for $i = 1, \ldots, n$, where $S = \sum_{i=1}^{n} a_i$. Find a subset $I \subseteq A$ such that $\sum_{a_i \in I} a_i = \sum_{a_i \in A \setminus I} a_i = \frac{S}{2}$.
Proof. We will only provide a proof for undirected graphs. The directed case follows analogously. In Figure 1, the labels on the links represent their cost and all links without labels have zero cost. Let $x = 0$. Clearly, the shortest link-disjoint pair of paths from s to d have a total cost of S. Thus, for any link-disjoint paths pair $\{P_1, P_2\}$, the best possible value for the min-sum min-max problem in this network is when $c(P_1) = c(P_2) = \frac{S}{2}$. However, finding this pair of link-disjoint paths requires solving the NP-complete partition problem. ■

Theorem 2 *The bounded min-sum problem link-disjoint paths problem is NP-complete.*

Proof. Here also, we use Figure 1 and the partition problem. Let $\Delta_1 = \frac{S}{2}$ and $\Delta_2 = x + \frac{S}{2}$. If $x = 0$, $\Delta_1 = \Delta_2$, otherwise $\Delta_1 \neq \Delta_2$. In either case, finding a pair of link-disjoint paths, where the shorter path is bounded by Δ_1 and the longer path is bounded by Δ_2 involves solving the NP-complete partition problem. ■
The widest min-sum link-disjoint paths problem is not NP-compete and in the next section, we provide an exact polynomial-time algorithm for it.

3 Algorithms

The MIN-SUM+ algorithm given below is an outline of algorithms that can be used for exactly solving the three NP-complete variants of the min-sum problem. The algorithm basically goes through all the possible candidate pairs of link-disjoint paths, namely those with a total cost equal to that of the min-sum link-disjoint paths pair returned by algorithms such as Bhandari's algorithm [1]. As in Bhandari's algorithm, in Steps $1-3$, MIN-SUM+ finds the shortest path p between s and d, and modifies the graph in such a way that the links along the shortest path are redirected from d to s and their cost is set to the negative of their original cost. If a shortest path q_1 exists in the modified graph G', $c(q_1)$ is used to identify the other candidate paths. This is due to the fact that any path q_k with a cost greater than $c(q_1)$ will lead to a pair of link-disjoint paths whose total cost (which is equal to $c(q_k) + c(p)$) is higher than the total cost of the shortest pair. Hence, the **while** loop in Step $4b$ exits when $c(q_k) > c(q_1)$. In G', all the shortest simple paths whose cost is equal to $c(q_1)$ can be obtained using such algorithms as the one given in [9]. Once all equal cost shortest paths are computed in G', their corresponding links that overlap with p in the original graph G are removed to obtain the corresponding shortest pairs of link-disjoint paths in the **while** loop of Step $4d$. Among these link-disjoint paths, the pair that satisfies the corresponding objective of the different problem variants is chosen as a solution.

MIN-SUM+(G,s,d)

1. Find the shortest path p between s and d.
2. Graph G' is obtained by directing each link (i, j) of p from d to s, and setting the cost of the links on the shortest path as $cost(j, i) = -cost(j, i)$.
3. Find the shortest path q_1 in G'.
4. **if** q_1 exists:
   (a) Set $k := 1$ and $C := c(q_1)$
   (b) **while** $(c(q_k) = C)$
      i. Find the $(k + 1)$-th shortest simple path q_{k+1}.
      ii. Set $k := k + 1$
   (c) Set $K = k, k := 1, min\_len := INF$
   (d) **while** $(k \leq K)$

i. In the original graph G, remove the interlacing links between p and q_k to obtain a pair of link-disjoint paths $\{q_{k1}, q_{k2}\}$

ii. For the **Min-Sum Min-Min** problem:
if $(min\_len > \min\{c(q_{k1}), c(q_{k2})\})$
A. Set $min\_len := \min\{c(q_{k1}), c(q_{k2})\}$
B. Set $P_1 := q_{k1}$ and $P_2 := q_{k2}$
For the **Min-Sum Min-Max** problem:
if $(min\_len > \max\{c(q_{k1}), c(q_{k2})\})$
A. Set $min\_len := \max\{c(q_{k1}), c(q_{k2})\}$
B. Set $P_1 := q_{k1}$ and $P_2 := q_{k2}$
For the **Bounded Min-Sum** problem:
if $(\min\{c(q_{k1}), c(q_{k2})\} \leq \Delta_1$ and $\max\{c(q_{k1}), c(q_{k2})\} \leq \Delta_2)$
A. Set $P_1 := q_{k1}$ and $P_2 := q_{k2}$
B. **return** $\{P_1, P_2\}$

iii. Set $k := k + 1$

(e) **return** $\{P_1, P_2\}$

5. **else return** no solution

The major operation in MIN-SUM+ is finding all possible shortest paths. For $k > 1$, finding each k-th shortest path using the algorithm in [9] takes $O(N(L + N \log N))$ time. Let K be the total number of such paths. Thus, the total running time of MIN-SUM+ is $O(K \cdot N(L + N \log N))$. The size of K, which is dependent on the type of network and the distribution of the link costs, can in the worst case grow exponentially. But by fixing K to a given constant, and exiting the algorithm after at most K link-disjoint paths are computed, heuristic algorithms can be obtained for the three NP-complete problem variants.

We also provide an outline of the WIDE-MIN-SUM algorithm, which is an exact algorithm for the widest min-sum problem. The algorithm begins by computing the shortest pair of link-disjoint paths in the original graph G. In each iteration k, a new graph G_{k+1} is obtained from G_k (G_1 is the original graph) by dropping all links with a bandwidth less or equal to that of the bottleneck link of the shortest link-disjoint paths in G_k. This process stops either when there are no link-disjoint paths in G_k or when the total cost of the shortest link-disjoint paths in G_k exceeds that of the shortest pair in the original graph. Finally, the pair with the highest bandwidth is returned. The WIDE-MIN-SUM algorithm is an exact algorithm because,

1. By dropping links with bandwidth less than that of a bottleneck link in G_k, only pairs of link-disjoint paths which use any of these links are affected. Hence, no better solution is dropped in the process.

2. If the shortest pair of link-disjoint paths in G_k have a total cost higher than that of the shortest pair in the original graph, dropping more links from G_k will not lead to a better result.

Since the major operation in WIDE-MIN-SUM is finding the shortest link-disjoint paths and in the worst case $O(L)$ links are dropped before exiting the algorithm, the complexity of the algorithm is $O(L^2 + LN \log N)$.

4 Results and Discussion

We present simulation results for random and lattice networks comparing the exact algorithms (MIN-SUM+), heuristic algorithms (MIN-SUM+ with $K = 2$) and the min-sum Suurballe's algorithm [6]. The results we have provided are only for min-sum min-min and min-sum min-max problem variants, because they represent extreme cases of the bounded min-sum problem, where the primary or the backup bounds are tight, respectively. Since the exact algorithm goes through all the possible pairs of min-sum link-disjoint paths, its complexity depends on the total number of such pairs of paths. If there is high granularity in the link costs (e.g., fractional costs) the number of equal cost (min-sum) link-disjoint paths is likely to be small and if there is no granularity (e.g., equal link costs), the heuristic and Suurballe's algorithms will more likely find the optimal solution. Therefore, to increase the possibility of having more min-sum pairs of paths, we use integral link costs that are randomly generated in the range $[1, 100]$. In these simulation results, the number of nodes is varied, and for each number of nodes, we have considered 1000 networks, each network with 1000 randomly generated requests. It can be seen that the heuristic algorithms (with $K = 2$) perform close to the their respective exact algorithms. The exact algorithms also perform in a reasonable time (order of tens of ms) as shown in Figure 2 (similar results have been obtained for the min-sum min-max problem).

Table 1: The average number of times that the heuristic algorithms and Suurballe's algorithm fail to find the optimal solution out of 1000 requests.

		Random Networks								
	N	100	200	300	400	500	600	700	800	900
min-min	Suurballe	3.9	6.67	7.78	8.1	7.83	7.35	6.69	5.73	5.19
	Heuristic	0.16	0.49	0.76	0.91	0.98	0.92	0.90	0.82	0.74
min-max	Suurballe	10.7	21.1	23.1	23.4	22.7	21.2	19.9	18.3	16.9
	Heuristic	1.05	5.36	7.46	8.84	9.82	10.1	10.3	10.2	10.0
		Lattice Networks								
	N	121	225	324	441	529	625	729	841	961
min-min	Suurballe	2.73	3.61	4.07	4.52	4.58	5.07	5.23	5.56	5.68
	Heuristic	0.04	0.1	0.12	0.14	0.14	0.2	0.22	0.22	0.27
min-max	Suurballe	5.08	6.22	6.74	7.41	7.74	7.78	8.09	8.29	8.48
	Heuristic	0.13	0.19	0.21	0.25	0.26	0.29	0.29	0.36	0.4

5 Conclusion

In this paper, we have considered the effect of having secondary objectives in the min-sum link-disjoint paths problem. We have shown that NP-complete secondary objectives lead to NP-complete min-sum problems. From the simulations, it can be inferred that the our heuristic algorithms in each case outperform Suurballe's algorithm, and the results obtained are close to the corresponding exact algorithms. In addition, because of the reduced search space, the exact algorithms can solve the

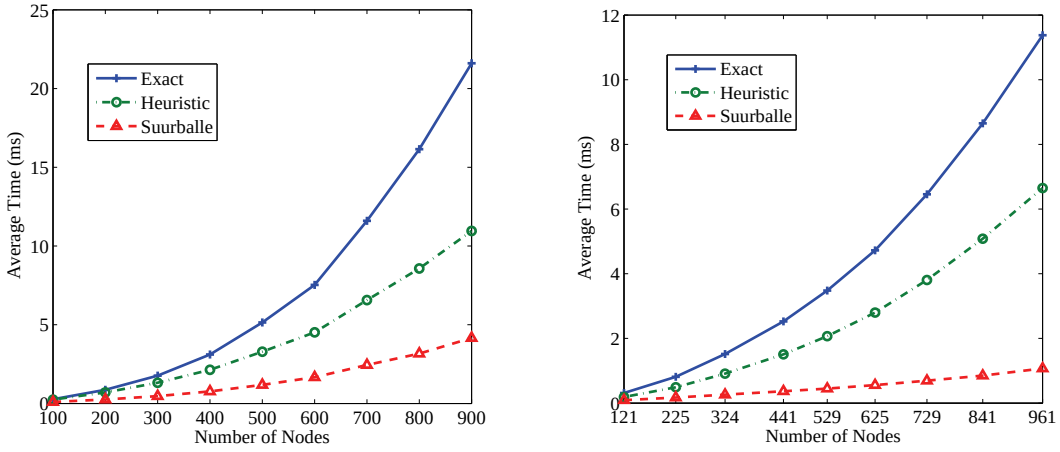

Figure 2: The average times (in ms) of the three algorithms for the min-sum min-min problem variant in (a) random networks, and (b) lattice networks.

respective problems in a reasonable running time. Therefore, for practical purposes, it is possible to use the exact algorithms.

References

[1] R. Bhandari, *Survivable Networks: Algorithms for Diverse Routing*, Kluwer Academic, 1999.

[2] M.R. Garey and D.S. Johnson, *Computers and Intractability: a Guide to the Theory of NP-Completeness*, Freeman, San Francisco, 1979.

[3] A. Itai, Y. Perl, and Y. Shiloach, "The complexity of finding maximum disjoint paths with length constraints," Networks, vol. 12, no. 3, pp. 277-286, 1982.

[4] C.-L. Li, S. T. McCormick, and D. Simchi-Levi, "The complexity of finding two disjoint paths with min-max objective function," *Discrete App. Math.*, vol 26, no 1, pp. 105–115, Jan. 1990.

[5] B.H. Shen, B. Hao, and A. Sen, "On multipath routing using widest pair of disjoint paths," *Workshop on High Performance Switching and Routing,* pp. 134-140, 2004.

[6] J. Suurballe and R. Tarjan,"A quick method for finding shortest pairs of disjoint paths," *Networks*, vol. 14, pp. 325–336, 1984.

[7] D. Xu, Y. Chen, Y. Xiong, C. Qiao, and X. He, "On the complexity of and algorithms for finding the shortest path with a disjoint counterpart," *IEEE/ACM Transactions on Networking*, vol. 14, no. 1, Feb. 2006.

[8] B. Yang, S.Q. Zheng, and E. Lu, "Finding two disjoint paths in a betwork with minsum-minmin objective function", *Proc. of the International Conference on Foundations of Computer Science*, Las Vegas, Nevada, Jun. 2007.

[9] J.Y. Yen, "Finding the k-shortest loopless paths in a network," *Management Science*, vol. 17, no. 11, pp. 712-716, July 1971.