MATHEMATICAL ENGINEERING
TECHNICAL REPORTS

On Shortest Disjoint Paths in Planar Graphs

Yusuke KOBAYASHI and Christian SOMMER

(Communicated by Kazuo MUROTA)

METR 2009–38 September 2009

DEPARTMENT OF MATHEMATICAL INFORMATICS
GRADUATE SCHOOL OF INFORMATION SCIENCE AND TECHNOLOGY
THE UNIVERSITY OF TOKYO
BUNKYO-KU, TOKYO 113-8656, JAPAN

WWW page: http://www.i.u-tokyo.ac.jp/mi/mi-e.htm

On Shortest Disjoint Paths in Planar Graphs

Yusuke Kobayashi* Christian Sommer†

Abstract

For a graph G and a collection of vertex pairs $\{(s_1,t_1),\ldots,(s_k,t_k)\}$, the k disjoint paths problem is to find k vertex-disjoint paths $P_1,\ldots,P_k$, where P_i is a path from s_i to t_i for each $i=1,\ldots,k$. In the corresponding optimization problem, the shortest disjoint paths problem, the vertex-disjoint paths P_i have to be chosen such that a given objective function is minimized. We consider two different objectives, namely minimizing the total path length (minimum sum, or short: min-sum), and minimizing the length of the longest path (min-max), for $k=2,3$.

min-sum: We extend recent results by Colin de Verdière and Schrijver to prove that, for a planar graph and for terminals adjacent to at most two faces, the Min-Sum 2 Disjoint Paths Problem can be solved in polynomial time. We also prove that, for six terminals adjacent to one face in any order, the Min-Sum 3 Disjoint Paths Problem can be solved in polynomial time.

min-max: The Min-Max 2 Disjoint Paths Problem is known to be **NP**-hard for general graphs. We present an algorithm that solves the problem for graphs with tree-width 2 in polynomial time. We thus close the gap between easy and hard instances, since the problem is weakly **NP**-hard for graphs with tree-width at least 3.

1 Introduction

The *vertex-disjoint paths problem* is one of the classic problems in algorithmic graph theory and combinatorial optimization, and has many applications, for example in transportation networks, VLSI-design [7, 16], or routing in networks [14, 22]. The input of the vertex-disjoint paths problem is a graph $G=(V,E)$ and k pairs of vertices $(s_1,t_1),\ldots,(s_k,t_k)$, for which the algorithm has to find k pairwise vertex-disjoint paths connecting s_i and t_i, if they exist. Paths are called *vertex-disjoint* if they have no vertices in common (except, possibly, at the end points).

In the optimization version of the problem, we are interested in *short* vertex-disjoint paths. We may want to minimize the total length (minimum sum) or the length of the longest path (min-max objective function). A more formal description of the problem is as follows.

Min-Sum k Disjoint Paths Problem (Min-Max k Disjoint Paths Problem)

Input: A graph $G=(V,E)$, k pairs of vertices $(s_1,t_1),(s_2,t_2),\ldots,(s_k,t_k)$ in G (which are sometimes called *terminals*), and a length function $l: E\to\mathbb{R}_+$.

Output : Vertex-disjoint paths $P_1,\ldots,P_k$ in G such that P_i is from s_i to t_i for $i=1,2,\ldots,k$, minimizing $\sum_{i=1}^{k} l(P_i)$ (or minimizing $\max_i l(P_i)$), where $l(P_i)=\sum_{e\in E(P_i)} l(e)$.

*Department of Mathematical Informatics, Graduate School of Information Science and Technology, University of Tokyo, Tokyo 113-8656, Japan E-mail: Yusuke_Kobayashi@mist.i.u-tokyo.ac.jp Supported by JSPS Research Fellowships for Young Scientists. This work was partially supported by the Global COE Program "The research and training center for new development in mathematics", MEXT, Japan.

†Department of Computer Science, Graduate School of Information Science and Technology, University of Tokyo, and National Institute of Informatics, Tokyo, Japan E-mail: sommer@nii.ac.jp

1.1 Related work

If k is part of the input, the vertex-disjoint paths problem is one of Karp's **NP**-hard problems [11], and it remains **NP**-hard even if G is constrained to be planar [13]. If k is a fixed number, k pairwise vertex-disjoint paths can be found in polynomial time in directed planar graphs [19] and in directed acyclic graphs [6], whereas the problem in general directed graphs is **NP**-hard even if $k = 2$ [6]. It is known that the disjoint paths problem in undirected graphs is solvable in polynomial time when $k = 2$ [20, 21, 24]. Perhaps the biggest achievement in this area is Robertson and Seymour's polynomial-time algorithm for the problem in undirected graphs when k is fixed [17].

The optimization problem is considerably harder. The problem of finding disjoint paths minimizing the total length is wide open and only a few cases are known to be solvable in polynomial time (see also Table 1). First, finding k disjoint s-t paths (i.e., $s_1 = \cdots = s_k = s$ and $t_1 = \cdots = t_k = t$) with minimum total length (*min-sum*) is still possible in polynomial time, since it reduces to finding the standard minimum cost flow [23]. The min-sum problem is solvable in linear time for graphs with bounded tree-width [18]. For the following two cases, the min-sum problem can also be reduced to the minimum cost flow problem and can thus be solved in polynomial time:

- All sources (or sinks) coincide, that is, $s_1 = \cdots = s_k$ (or $t_1 = \cdots = t_k$, respectively).
- The graph is planar, all terminals are incident with a common face, and their cyclic order is $s_1, \ldots, s_k, t_k, \ldots, t_1$ (called *well-ordered*).

Another special case of the min-sum problem has recently been solved by Colin de Verdière and Schrijver [3]. They showed the following:

Theorem 1 (Colin de Verdière and Schrijver [3])**.** *If a given directed or undirected graph G is planar, all sources are incident to one face S, and all sinks are incident to another face $T \neq S$, then we can find k vertex-disjoint paths in G with minimum total length in $\mathrm{O}(kn \log n)$ time.*

If the length of the longest path is to be minimized (*min-max*), the problem seems to be harder than the min-sum problem. The problem of finding two s-t paths minimizing the length of the longer path is **NP**-hard for an acyclic directed graph [10], but 2-approximable [12] using the *min-sum* version. Moreover, the problem is strongly **NP**-hard for general directed graphs when s_1, s_2, t_1, and t_2 are distinct [12]. For an overview, see Table 2.

	Conditions	Complexity
$k = 2$	directed	**NP**-hard
	directed, planar, one face	OPEN
	undirected	OPEN
	undirected, planar, two faces	**P** (Theorem 2)
$k = 3$	undirected, planar, one face	**P** (Theorem 11)
k: fixed	undirected	OPEN
k: general	undirected	**NP**-hard
	$s_1 = \cdots = s_k$ and/or $t_1 = \cdots = t_k$	**P** (Min-cost flow)
	planar, one face, well-ordered	**P** (Min-cost flow)
	planar, $S \neq T$ faces	**P** [3]
	bounded tree-width	$\mathrm{O}(n)$ [18]

Table 1: Results for the Min-Sum Disjoint Paths Problem.

	Conditions	Complexity
$k = 2$	directed, acyclic, $s_1 = s_2$, $t_1 = t_2$	**NP**-hard [10]
		pseudo-polynomial [12]
	directed, $s_1 = s_2$, $t_1 = t_2$	2-approx. [12]
	directed	strongly **NP**-hard [12]
	undirected, tree-width ≥ 3, planar	**NP**-hard ([26] and Theorem 14)
	undirected, tree-width ≤ 2	**P** (Theorem 18)

Table 2: Results for the Min-Max Disjoint Paths Problem.

Yet another variant of the objective function for the problem of finding two disjoint paths for one pair of terminals is the following: Before summing up the path lengths, the length of the longer path is multiplied by a factor $\alpha \in (0, 1)$, which parameterizes the cost function. $\alpha = 1$ would yield the min-sum variant and $\alpha = 0$ would yield the so-called 'min-min' variant, in which the length of the shorter path is to be minimized, which is **NP**-hard [25]. For $\alpha \in (0, 1)$ there is an approximation algorithm with ratio $\frac{1+\alpha}{2\alpha}$, which, for directed graphs, is claimed to be optimal unless the polynomial hierarchy collapses completely [27]. If the length of the shorter path is multiplied by $\alpha \in (0, 1)$, there is an approximation algorithm with ratio $\frac{2}{1+\alpha}$, which is claimed to be tight as well [26].

1.2 Contribution

We extend the min-sum results of Colin de Verdière and Schrijver [3] for undirected graphs and $k = 2$ as follows: the two disjoint faces F_1, F_2 may be 'mixed' such that

- s_1, s_2, t_1 are incident to F_1 and t_2 is incident to F_2 (Theorem 3),
- s_1, t_1 are incident to F_1 and s_2, t_2 are incident to F_2 (Theorem 6).

Our algorithms consist of non-trivial reductions to Theorem 1. By combining Theorem 1 with our new results, flow reductions, and trivially infeasible inputs, we obtain the following theorem.

Theorem 2. *Let $G = (V, E)$ be an undirected planar graph, and let F_1 and F_2 be its faces. If each terminal is on one of the boundaries of F_1 and F_2, then the Min-Sum 2 Disjoint Paths Problem in G is solvable in polynomial time.*

We also give a polynomial-time algorithm for the Min-Sum Disjoint Paths Problem when $k = 3$ and all terminals are incident to one face (Theorem 11). Our contribution is to give an algorithm for the case when the terminals are not well-ordered, by a non-trivial reduction to Theorem 1. For a summary, see Table 3.

	one face	two faces
$k = 2$	flow	Theorems 1, 3, 6
$k = 3$	flow, Theorem 11	OPEN

Table 3: *min-sum* results in planar undirected graphs.

For the Min-Max 2 Disjoint Paths Problem, we draw the line between tractable and hard problems: We prove weak **NP**-hardness of the Min-Max 2 Disjoint Paths Problem for planar graphs with tree-width 3 using a reduction from the Partition problem (Theorem 14). We

later learned that the reduction was used independently in almost the same manner in [26] already, without an explicit link to the tree-width and the min-max variant. For graphs with tree-width 2 (including series-parallel graphs and outer-planar graphs), we provide a polynomial-time algorithm (Theorem 18). The same algorithm also works for the Min-Min 2 Disjoint Paths Problem and the α-variants from [26, 27].

For both the min-sum and the min-max versions and for the variants with cost functions parameterized by α as defined in [26, 27], we give a pseudo-polynomial-time algorithm for graphs with bounded tree-width (Theorem 19). The algorithm runs in polynomial time for the min-sum objective function [18].

2 Preliminaries

Let $G = (V, E)$ be an undirected graph with vertex set V and edge set E, and let $n = |V|$ denote the number of vertices. Since we consider vertex-disjoint paths, in what follows, we may assume that the graph has no multiple edges and no self-loops. An edge connecting $u, v \in V$ is denoted by uv, whereas (u, v) represents the arc from u to v in a directed graph. For a subgraph H of G, the vertex set and the edge set of H are denoted by $V(H)$ and $E(H)$, respectively. Let $\delta(v)$ denote the set of edges incident to $v \in V$. For $U \subseteq V$, let $G[U]$ be the subgraph of $G = (V, E)$ induced by U, that is, its vertex set is U and its edge set consists of all edges in E with both ends in U. A graph G is *planar* if it can be embedded in a plane Σ such that no edges intersect, except at their end points. To simplify notation, we do not distinguish between a vertex of G and the point of Σ used in the embedding to represent the vertex, and we do not distinguish between an edge and the curve on Σ representing it. A *region* is a subset of Σ, and for a region R, let $G[[R]]$ denote the subgraph of G consisting of the vertices and the edges in R. For a face F of a planar graph, let ∂F denote the boundary of F. A planar graph is *outer-planar* if it allows for a planar embedding such that all its vertices are on the outer face. A *path* P, which is denoted by $P = (v_1, v_2, \ldots, v_l)$, is a subgraph consisting of vertices $v_1, \ldots, v_l$ and edges $e_1 = v_1v_2, \ldots, e_{l-1} = v_{l-1}v_l$. When $v_1 = v_l$, it is called a *cycle*. A path (or a cycle) is *simple* if $v_i \neq v_j$ for distinct i, j (except for $v_1 = v_l$). For a simple path $P = (v_1, v_2, \ldots, v_l)$, let $P^{[v_i, v_j]}$ denote the path $(v_i, v_{i+1}, \ldots, v_j)$, and it is called a *subpath* of P. For a length function $l : E \to \mathbb{R}_+$, the length of a path P is denoted by $l(P)$, and for a pair of vertices $u, v \in V$, let $d_G(u, v)$ denote the length of a shortest path connecting u and v in G.

Suppose that two simple paths $P_1 = (v_1^1, v_2^1, \ldots, v_{l_1}^1)$ and $P_2 = (v_1^2, v_2^2, \ldots, v_{l_2}^2)$ in a planar graph have a common vertex $v_i^1 = v_j^2 = v$. We say that P_1 and P_2 *cross at* v if $vv_{i-1}^1, vv_{j-1}^2, vv_{i+1}^1$, and v, v_{j+1}^2 are incident with v cyclically in this order. Similarly, suppose that two paths P_1 and P_2 have a common subpath P in a planar graph. Assume that if we contract all edges in P, then the two paths correspoding to P_1 and P_2 cross at the vertex corresponding to P. In this case, we say that P_1 and P_2 *cross at* P. We say that a path or a cycle *crosses with itself* if it has two crossing subpaths.

For a simple cycle C in a planar graph, the *inside* of C is the bounded closed region whose boundary is C. We define the inside of C when C is not simple but does not cross with itself (see Fig. 1).

The tree-width of a graph was introduced by Halin [8], but it went unnoticed until it was rediscovered by Robertson and Seymour [15] and, independently, by Arnborg and Proskurowski [1]. The *tree-width* of a graph is defined as follows.

Definition 1. Let G be a graph, T a tree and let $\mathcal{V} = \{V_t \subseteq V(G) \mid t \in V(T)\}$ be a family of vertex sets of G indexed by the vertices t of T. The pair $(T, \mathcal{V})$ is called a *tree-decomposition* of G if it satisfies the following three conditions:

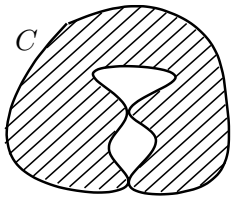

Figure 1: Inside of a cycle.

- $V(G) = \bigcup_{t \in T} V_t$
- for every edge $e \in G$ there exists a $t \in T$ such that both ends of e lie in V_t
- If $t, t', t'' \in V(T)$ and t' lies on the path of T between t and t'', then $V_t \cap V_{t''} \subseteq V_{t'}$.

The *width* of $(T, \mathcal{V})$ is the number $\max\{|V_t| - 1 \mid t \in T\}$ and the *tree-width* $\mathrm{tw}(G)$ of G is the minimum width of any tree-decomposition of G.

The tree-width is a good measure of the algorithmic tractability of graphs. It is known that a number of hard problems on graphs, such as "Hamiltonian cycle" and "chromatic number", can be solved efficiently when the given graph has small tree-width [1]. A graph has tree-width 1 if and only if it is a forest, and families of graphs with tree-width at most 2 include outer-planar graphs and series-parallel graphs.

3 Min-Sum Objective Function

In this section, we deal with the Min-Sum k Disjoint Paths Problem for $k = 2, 3$. To simplify the arguments, we use a perturbation technique such that all shortest paths are unique (see [5]). For $E = \{e_1, e_2, \ldots, e_m\}$ and $l : E \to \mathbb{R}_+$, we use a new length function $l' : E \to \mathbb{R}_+$ defined by $l'(e_i) = l(e_i) + \varepsilon^i$ for each i, where ε is an infinitely small positive number. Then, each path has a different length. In particular, the Min-Sum k Disjoint Paths Problem has a unique optimal solution if it has a feasible solution. In what follows, in this section, we simply denote the perturbed length function by l.

3.1 Min-Sum 2 Disjoint Paths Problem

In this section, we prove Theorem 2, which we restate here.

Theorem. *Let $G = (V, E)$ be an undirected planar graph, and let F_1 and F_2 be its faces. If every terminal is on $\partial F_1 \cup \partial F_2$, then the Min-Sum 2 Disjoint Paths Problem in G is solvable in polynomial time.*

Proof. The four terminals s_1, s_2, t_1, and t_2 may lie on two faces as follows:

- s_1, s_2, t_1, and t_2 are incident to F_1 (min-cost flow [23] or trivially infeasible),
- s_1, s_2 are incident to F_1 and t_1, t_2 are incident to F_2 (Theorem 1 due to [3]),
- s_1, s_2, t_1 are incident to F_1 and t_2 is incident to F_2 (Theorem 3), or
- s_1, t_1 are incident to F_1 and s_2, t_2 are incident to F_2 (Theorem 6).

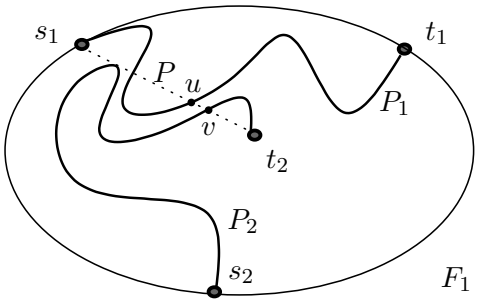

Figure 2: Definitions of P, u, and v.

The remaining cases (e.g. the case with s_2 alone on one face) are symmetric for undirected graphs. □

Theorem 3. *Let $G = (V, E)$ be an undirected planar graph, and F_1 and F_2 be its faces. If three terminals are on ∂F_1 and the remaining terminal is on ∂F_2, then the Min-Sum 2 Disjoint Paths Problem in G is solvable in $\mathrm{O}(n^3 \log n)$ time.*

Proof. Let $s_1, s_2, t_1 \in \partial F_1$ and $t_2 \in \partial F_2$ be terminals. Let P be the shortest path connecting s_1 and t_2. The basic idea of the algorithm is to, for all pairs of vertices u, v on P, transform the original problem to an instance of the problem that can be solved using the algorithm by Colin de Verdière and Schrijver. The transformation is described in Lemma 5. In Lemma 4 we prove that the solution remains optimal.

Lemma 4. *Suppose that a pair of paths (P_1, P_2) is the unique optimal solution of the Min-Sum 2 Disjoint Paths Problem. Let u be the vertex in $V(P_1) \cap V(P)$ closest to t_2 in the ordering along P, and let v be the vertex in $V(P_2) \cap V(P^{[u,t_2]})$ closest to u in the ordering along P (Fig. 2). Then, $P^{[v,t_2]}$ is a subpath of P_2.*

Proof. Suppose that $P^{[v,t_2]}$ is not a subpath of P_2, and define $P_2' = P_2^{[s_2,v]} \cup P^{[v,t_2]}$. By definition of u and v, P_1 and P_2' are disjoint. Since P is the shortest path, every subpath $P^{[a,b]}$ is the shortest path between a and b, thus, $l(P^{[v,t_2]}) < l(P_2^{[v,t_2]})$, which implies that $l(P_2') < l(P_2)$. Here we use the fact that the shortest path is unique. Then, (P_1, P_2') is a shorter solution, which contradicts the optimality of (P_1, P_2). □

Lemma 5. *For distinct vertices u, v on P such that u is closer to s_1 than v, in $\mathrm{O}(n \log n)$ time, we can either find two simple disjoint paths P_1 and P_2 minimizing the total length $l(P_1) + l(P_2)$ such that*

1. *P_i connects s_i and t_i for $i = 1, 2$,*
2. *$u \in V(P_1)$ and $V(P_1) \cap V(P) \subseteq V(P^{[s_1,u]})$, and*
3. *$P_2 \cap P^{[u,t_2]} = P^{[v,t_2]}$,*

or conclude that such P_1 and P_2 do not exist.

Proof. Delete all vertices in $V(P^{[u,t_2]}) \setminus \{u, v, t_2\}$. This yields a graph G'. Note that u and v are on the boundary of the same face F' in G', because all internal vertices of $P^{[u,v]}$ have been removed.

We find three paths Q_1, Q_2, and Q_3 in G' minimizing the total length such that

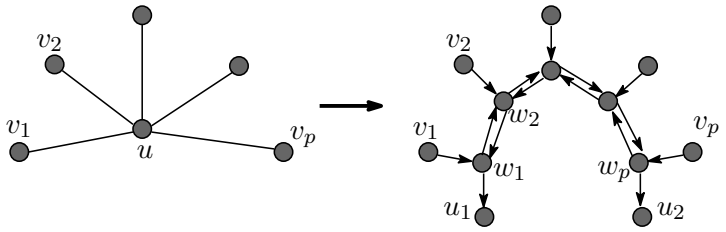

Figure 3: Construction of D_2.

- Q_1 connects s_1 and u, Q_2 connects t_1 and u, Q_3 connects s_2 and v,
- $V(Q_2) \cap V(Q_3) = V(Q_3) \cap V(Q_1) = \emptyset$, and $V(Q_1) \cap V(Q_2) = \{u\}$.

In order to apply Theorem 1, we divide u into two distinct vertices and construct a digraph as follows. Let $v_1, v_2, \ldots, v_p$ be the vertices in G' adjacent to u such that $v_1, v_p \in \partial F'$ and $uv_1, uv_2, \ldots, uv_p$ are incident to u in this order. Let $D_1 = (V_1, E_1)$ be the digraph obtained from $G - u$ by replacing each edge with two parallel arcs of opposite direction. Define a digraph $D_2 = (V_2, E_2)$ (see Fig. 3) by

$$V_2 = V_1 \cup \{w_1, w_2, \ldots, w_p, u_1, u_2\},$$
$$E_2 = E_1 \cup \bigcup_{i=1}^{p} \{(v_i, w_i)\} \cup \bigcup_{i=1}^{p-1} \{(w_i, w_{i+1}), (w_{i+1}, w_i)\} \cup \{(w_1, u_1), (w_p, u_2)\}.$$

Define a new length function $l' : E_2 \to \mathbb{R}_+$ as

$$l'(e) = \begin{cases} l(xy) & \text{if } e = (x, y) \text{ or } (y, x) \text{ for } xy \in E, \\ l(v_i u) & \text{if } e = (v_i, w_i), \\ 0 & \text{otherwise.} \end{cases}$$

By finding three disjoint paths Q'_1, Q'_2, Q'_3 with minimum total length such that Q'_1 is from s_1 to u_1 (or u_2, respectively), Q'_2 is from t_1 to u_2 (or u_1, respectively), and Q'_3 is from s_2 to v, we can obtain the desired paths Q_1, Q_2, and Q_3. This can be done in $\mathrm{O}(n \log n)$ time by Theorem 1.

Then, $P_1 = Q_1 \cup Q_2$ and $P_2 = Q_3 \cup P^{[v,t_2]}$ are the desired disjoint paths in G. □

By Lemma 4, we can find the optimal solution of the Min-Sum 2 Disjoint Paths Problem by executing the procedure described in Lemma 5 for each pair of vertices u and v on the shortest path between s_1 and t_2. This concludes the proof of Theorem 3. □

Theorem 6. *Let $G = (V, E)$ be an undirected planar graph, and F_1 and F_2 be its faces. If $s_1, t_1 \in \partial F_1$ and $s_2, t_2 \in \partial F_2$ are terminals, then the Min-Sum 2 Disjoint Paths Problem in G is solvable in $\mathrm{O}(n^3 \log n)$ time.*

Proof. Let C_i^1 and C_i^2 be components of $\partial F_i - \{s_i, t_i\}$ for $i = 1, 2$. We say that a path P connecting s_1 and t_1 (or s_2 and t_2) is *in the C_1^j side of F_2* (or *in the C_2^j side of F_1*, respectively) if F_1 and F_2 are not on the inside of $P \cup C_1^j$ (or $P \cup C_2^j$, respectively) for $j = 1, 2$ (Fig. 4).

The following lemma directly implies Theorem 6.

Lemma 7. *For $j_1, j_2 \in \{1, 2\}$, there is an $\mathrm{O}(n^3 \log n)$ time algorithm to find two paths P_1 and P_2 such that*

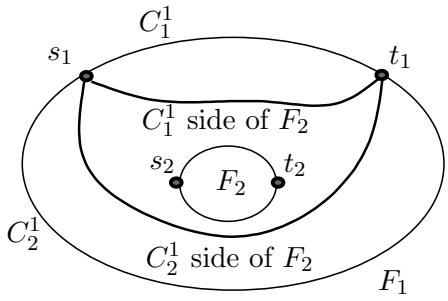

Figure 4: C_1^1 side and C_2^1 side.

(1) P_i *connects* s_i *and* t_i*,*

(2) P_1 *is on the* $C_1^{j_1}$ *side of* F_2*,* P_2 *is on the* $C_2^{j_2}$ *side of* F_1*, and*

(3) if the optimal solution of the original Min-Sum 2 *Disjoint Paths Problem satisyfies (1) and (2),* (P_1, P_2) *is the optimal solution.*

In what follows, we show the lemma. By symmetry, it suffices to consider the case $j_1 = j_2 = 1$.

We take the shortest path connecting s_i and t_i that is on the C_i^1 side for $i = 1, 2$. Note that J_i is not necessarily simple, but J_i does not cross with itself (see Fig. 5). The inside of $C_i^1 \cup J_i$ is denoted by R_i. A precise description of the algorithm of this part is as follows.

Lemma 8. *We can find the shortest path* J_i *among all paths connecting* s_i *and* t_i *that are on the* C_i^1 *side in* $\mathrm{O}(n \log n)$ *time.*

Proof. We only deal with the case of $i = 1$. The case for $i = 2$ is analog. We find the shortest paths J_s from s_1 to ∂F_2 and J_t from t_1 to ∂F_2. Then, J_s and J_t do not cross. Let v_s and v_t be end vertices of J_s and J_t in ∂F_2, respectively. We choose a path $J \subseteq \partial F_2$ connecting v_s and v_t such that $J_s \cup J \cup J_t$ is on the C_1^1 side of F_2.

Then, the desired J_1 is contained in the inside of $C_1^1 \cup J_s \cup J \cup J_t$. By finding the shortest path from s_1 to t_1 in the inside of $C_1^1 \cup J_s \cup J \cup J_t$, we can find the desired J_1. This can be done in $\mathrm{O}(n \log n)$ time by Dijkstra's algorithm [4]. □

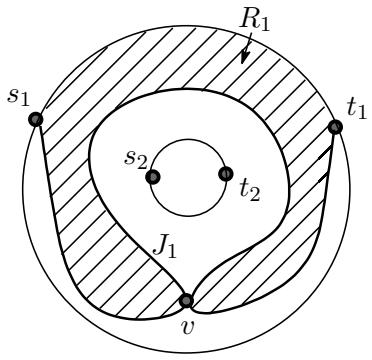

Figure 5: Case when J_1 is not simple.

Then, one can see that $R_1 \cap R_2 \subseteq J_1 \cap J_2$. In other words, J_1 and J_2 do not cross.

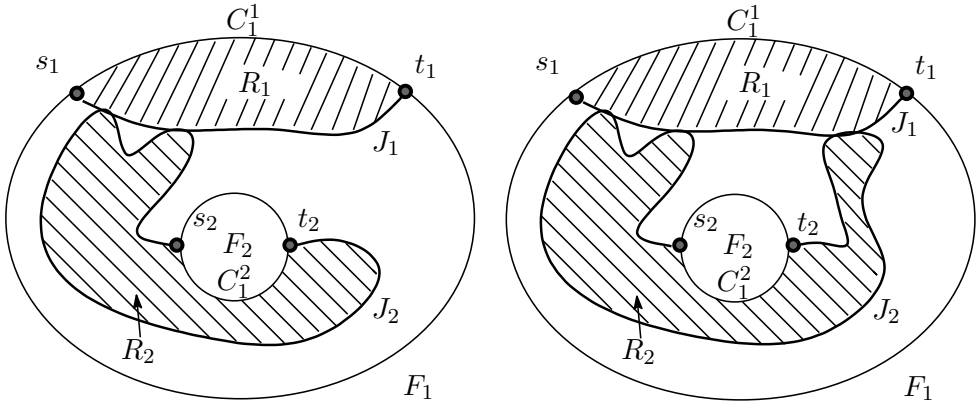

Figure 6: Relation betwen F_1' and F_2'.

Lemma 9. *Suppose that J_1, J_2, R_1, R_2 are defined as above. Consider the problem of finding the pair of paths minimizing the total length among the pairs of disjoint paths (P_1, P_2) satisfying (1) and (2) of Lemma 7. If this problem has a feasible solution, then the optimal solution (P_1, P_2) satisfies that $P_i \subseteq R_i$ for $i = 1, 2$.*

Proof. Suppose that (P_1, P_2) is the optimal solution of the problem, and the interior of a path $P_1^{[u,v]}$ is in $\Sigma - R_1$ for some $u, v \in V(P_1) \cap V(J_1)$.

Then, replace P_1 by a new path P_1' defined by $P_1' = (P_1 \setminus P_1^{[u,v]}) \cup J_1^{[u,v]}$. Since J_1 is the shortest path, we have $l(J_1^{[u,v]}) < l(P_1^{[u,v]})$, and hence $l(P_1') < l(P_1)$. P_1' and P_2 are mutually disjoint, because P_1 and P_2 are mutually disjoint. This contradicts the optimality of (P_1, P_2). Hence, we have $P_1 \subseteq R_1$, and $P_2 \subseteq R_2$ is shown in the same way. □

By this lemma, it suffices to find the disjoint paths in $G[[R_1 \cup R_2]]$. We distinguish the following two cases. In the first case, we consider simple paths J_i. In the second case, we modify J_i such that we get a simple path, for which the procedure of the first case can be applied.

Case 1: First, we consider the case when both J_1 and J_2 are simple. Define $G' = G[[R_1 \cup R_2]]$, and let F_1' and F_2' be its faces containing F_1 and F_2, respectively. Then we can consider the following two cases: $F_1' = F_2'$ or $F_1' \neq F_2'$ (see Fig. 6). Since the case $F_1' = F_2'$ is solvable by a min-cost flow algorithm in G', we may assume that $F_1' \neq F_2'$.

Then, there exist vertices $v_1 \in \partial F_2' \cap J_1$ and $v_2 \in \partial F_1' \cap J_2$. We take a sequence of vertices $v_1^0, v_1^1, \ldots, v_1^{k_1}$ in R_1 such that $v_1^0 = v_1$, $v_1^{k_1} \in C_1^1$ (if C_1^1 consists of one edge, we add a vertex in the middle of the edge, and the added vertex is $v_1^{k_1}$) and v_1^l is on the boundary of the face of $G' - ((\delta(v_1^0) \cup \cdots \cup \delta(v_1^{l-1})) - R_2)$ containing F_2'. Note that we can use $G' - \{v_1^0, \ldots, v_1^{l-1}\}$ instead of $G' - ((\delta(v_1^0) \cup \cdots \cup \delta(v_1^{l-1})) - R_2)$ if $v_1^0, v_1^1, \ldots, v_1^{l-1} \notin J_2$. We take $v_2^0, v_2^1, \ldots, v_2^{k_2}$ in the same way. We also note that such sequences exist, because J_1 and J_2 are simple.

Lemma 10. *Given $0 \leq l_1 \leq k_1$ and $0 \leq l_2 \leq k_2$, we can find in $\mathrm{O}(n \log n)$ time disjoint paths P_1 and P_2 of minimum total length such that P_i passes through $v_i^{l_i}$ but not through $v_i^0, \ldots, v_i^{l_i - 1}$ for each i.*

Proof. We find four paths $P_{s_1}, P_{t_1}, P_{s_2}$, and P_{t_2} in $G_{l_1,l_2} = G' - ((\delta(v_1^0) \cup \cdots \cup \delta(v_1^{l_1-1})) - R_2) - ((\delta(v_2^0) \cup \cdots \cup \delta(v_2^{l_2-1})) - R_1)$ minimizing the total length such that P_{s_i} connects s_i and $v_i^{l_i}$, P_{t_i} connects t_i and $v_i^{l_i}$, and they are mutually vertex-disjoint except for $V(P_{s_i}) \cap V(P_{t_i}) = \{v_i^{l_i}\}$.

In G_{l_1,l_2}, the vertices $v_1^{l_1}, s_2, t_2$ are on the boundary of a common face, and the vertices $v_2^{l_2}, s_1, t_1$ are on the boundary of another face. Thus, using the same argument as in the proof of Lemma 5, we can find in $O(n \log n)$ time four such paths by Theorem 1 with $k = 4$.

Then, $P_1 = P_{s_1} \cup P_{t_1}$ and $P_2 = P_{s_2} \cup P_{t_2}$ are the desired paths. □

We can easily see that if a path P_i between s_i and t_i is contained in R_i, then there exists an integer $0 \leq l \leq k_i$ such that P_i passes through v_i^l but not through $v_i^0, \ldots, v_i^{l-1}$, because P_i passes through at least one of $v_i^0, \ldots, v_i^{k_i}$. Therefore, in order to solve the problem in Lemma 7, it suffices to execute the procedure described in Lemma 10 for every pair (l_1, l_2). Hence, it can be done in $O(n^3 \log n)$ time.

Case 2: Next we consider the case when J_i is not simple. Assume that J_1 passes through a vertex v twice, and $J_1^{[s_1,v]} \cup J_1^{[v,t_1]}$ is a simple path which is shorter than J_1 and in the C_2^1 side of F_2 (see Fig. 5).

By Lemma 9, when we find the shortest disjoint paths (P_1, P_2) satisfying (1) and (2) in Lemma 7, we may assume that P_2 and $J_1^{[s_1,v]} \cup J_1^{[v,t_1]}$ intersect only at v, or do not intersect. For a pair of paths (P_1, P_2) satisfying (1) and (2), if $P_2 \cap (J_1^{[s_1,v]} \cup J_1^{[v,t_1]}) = \emptyset$, (P_1, P_2) is not the optimal solution of the original Min-Sum 2 Disjoint Paths Problem, because $(J_1^{[s_1,v]} \cup J_1^{[v,t_1]}, P_2)$ is shorter than (P_1, P_2).

Hence, we only consider the case when P_2 passes through v. In this case, we find the shortest path J_1' from s_1 to t_1 in $G[[R_1]] - v$, and replace J_1 with J_1'. Then we can execute the same procedure as for Case 1.

This completes the proof of Lemma 7, and Theorem 6 follows. □

3.2 Min-Sum 3 Disjoint Paths Problem

Theorem 11. *Let $G = (V, E)$ be an undirected planar graph and let F be its face. If all six terminals are on ∂F, then the Min-Sum 3 Disjoint Paths Problem in G is solvable in $O(n^4 \log n)$ time.*

Proof. We assume that F is the outer unbounded face. By exchanging the labels of the vertices, it suffices to consider the case when s_1, t_1, s_2, t_2, s_3, and t_3 are clockwise in this order along ∂F. Note that, in the other cases, we can solve the problem by a minimum cost flow algorithm [23], or we can conclude immediately that there is no feasible solution.

Let ∂F_i be the component of $\partial F - \{s_i, t_i\}$ that does not contain any terminals for $i = 1, 2, 3$. For each i, let J_i be the shortest path connecting s_i and t_i in G, and the inside of $J_i \cup \partial F_i$ is denoted by R_i. Then, one can see that $R_i \cap R_j \subseteq J_i \cap J_j$ for distinct i and j. In other words, J_i and J_j do not cross (see Fig. 7).

In the same way as Lemma 9, we have the following lemma.

Lemma 12. *If there exists a feasible solution of the Min-Sum 3 Disjoint Paths Problem, the optimal solution (P_1, P_2, P_3) satisfies that $P_i \subseteq R_i$ for $i = 1, 2, 3$.*

By Lemma 12, it suffices to deal with the graph $G[[R_1 \cup R_2 \cup R_3]]$. For convenience, we add to $G[[R_1 \cup R_2 \cup R_3]]$ three edges t_1s_2, t_2s_3, t_3s_1 such that all terminals are on the boundary of the outer unbounded face. Let G' denote the obtained planar graph.

Then, there exists a bounded face (or a single vertex) Q of G' intersecting with J_1, J_2, and J_3. Take $v_i \in V(J_i) \cap Q$ for $i = 1, 2, 3$ (see Fig. 7).

In a similar way as Case 1 in the proof of Lemma 7, we find a sequence of vertices $v_1^0, v_1^1, \ldots, v_1^{k_1}$ in R_1 such that $v_1^0 = v_1$, $v_1^{k_1} \in \partial F_1$ and v_1^l is on the boundary of the face of

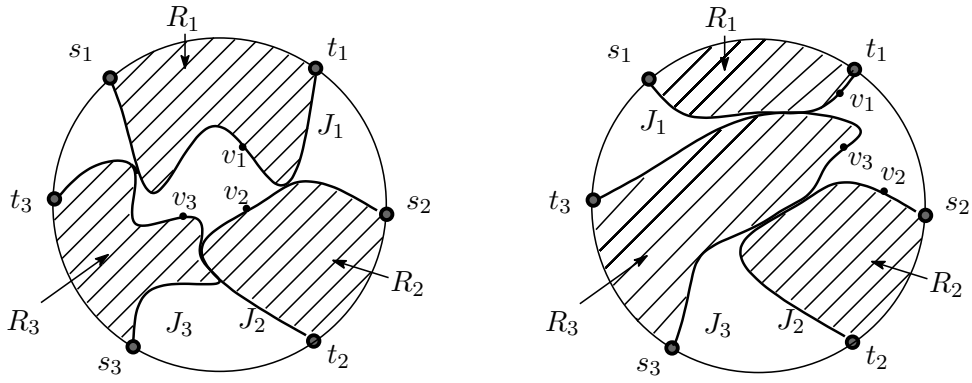

Figure 7: Definitions of J_i and R_i.

$G' - E_1^l$ containing Q, where $E_1^l = (\delta(v_1^0) \cup \cdots \cup \delta(v_1^{l-1})) - (R_2 \cup R_3)$. We also find $v_i^0, v_i^1, \ldots, v_i^{k_i}$ and define E_i^l for $i = 2, 3$ in the same way.

Next we show the following lemma.

Lemma 13. *For integers l_1, l_2, l_3 with $0 \leq l_i \leq k_i$, we can find in $O(n \log n)$ time disjoint paths P_1, P_2, P_3 of minimum total length such that P_i passes through $v_i^{l_i}$ but not through $v_i^0, \ldots, v_i^{l_i - 1}$ for each i.*

Proof. We find six paths P_{s_1}, P_{t_1}, P_{s_2}, P_{t_2}, P_{s_3}, and P_{t_3} in $G' - E_1^{l_1} - E_2^{l_2} - E_3^{l_3}$ minimizing the total length such that P_{s_i} connects s_i and $v_i^{l_i}$, P_{t_i} connects t_i and $v_i^{l_i}$, and they are mutually vertex disjoint except for $V(P_{s_i}) \cap V(P_{t_i}) = \{v_i^{l_i}\}$. Since $v_1^{l_1}, v_2^{l_2}$, and $v_3^{l_3}$ are on the boundary of a same face of $G' - E_1^{l_1} - E_2^{l_2} - E_3^{l_3}$, using the same argument as the proof of Lemma 5, we can find in $O(n \log n)$ time six such paths by Theorem 1 with $k = 6$.

Then, P_1, P_2, P_3 defined by $P_i = P_{s_i} \cup P_{t_i}$ are the desired paths. □

If a path P_i between s_i and t_i is contained in R_i, then there exists an integer $0 \leq l \leq k_i$ such that P_i passes through v_i^l but not through $v_i^0, \ldots, v_i^{l-1}$. Thus, in order to solve the original problem, it suffices to execute the procedure described in Lemma 13 for every triple (l_1, l_2, l_3), which can be done in $O(n^4 \log n)$ time. □

4 Min-Max Objective Function

We reprove **NP**-hardness of the Min-Max 2 Disjoint Paths Problem for planar graphs with tree-width 3, using a reduction from the Partition problem. We later learned that the reduction was used independently in almost the same manner in [26] already, without an explicit link to the tree-width and the min-max variant. For graphs with tree-width 2 (including series-parallel graphs and outer-planar graphs), we provide a polynomial-time algorithmfor the Min-Max 2 Disjoint Paths Problem.

4.1 Hardness

In the Partition problem, we are given m items with weights $w_1, w_2 \ldots w_m \in \mathbb{Z}_+$, which are to be split into two subsets of the same weight. The Partition problem is also one of Karp's **NP**-hard problems [11]. Note that the problem is weakly **NP**-hard, that is, it is **NP**-hard when the input size of the problem is $O(m + \log(\max_i w_i + 1))$. We also note that, using dynamic programming, the problem is solvable in time polynomial in m and $\max_i w_i + 1$.

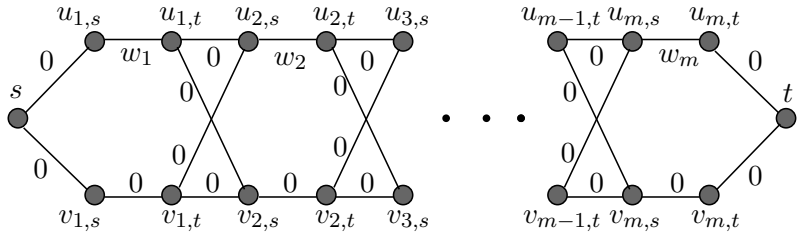

Figure 8: Graph G with tree-width 3.

Observe that the hard graphs for the min-max objective function are trivial for the min-sum objective function.

Theorem 14 ([26])**.** *The Min-Max* 2 *Disjoint Paths Problem is (weakly)* **NP***-hard for planar graphs with tree-width at least* 3*.*

Proof. The polynomial-time reduction from the **NP**-hard problem Partition$(w_0, w_1, \ldots, w_{m-1})$ to a graph is shown in Fig. 8.

For each weight w_i we add four vertices (two on each side, *s*ource and *t*arget) and six edges; the graph $G = (V, E)$ is defined by

$$V := \{s, t\} \cup \bigcup_{i=1}^{m} V_i, \qquad E := \{su_{1,s}, sv_{1,s}, u_{m,s}u_{m,t}, v_{m,s}v_{m,t}, u_{m,t}t, v_{m,t}t\} \cup \bigcup_{i=1}^{m-1} E_i,$$

where

$$V_i = \{u_{i,s}, u_{i,t}, v_{i,s}, v_{i,t}\}, \qquad E_i = \{u_{i,s}u_{i,t}, u_{i,t}u_{i+1,s}, u_{i,t}v_{i+1,s}, v_{i,s}v_{i,t}, v_{i,t}v_{i+1,s}, v_{i,t}u_{i+1,s}\}.$$

All edges have weight 0, except for the edges $(u_{i,s}, u_{i,t})$, which have weight w_i. This graph has tree-width 3 as we can put two consecutive vertices from one side and their counterparts on the other side into a subset, which yields a path as tree-decomposition. Note that the graph can be embedded in a plane without edge intersections.

Two optimal paths with respect to the min-max objective function solve the partition problem as follows: if the edge $u_{i,s}u_{i,t}$ is on path P_1, include w_i in set S_1, otherwise include w_i in set S_2. In each 'step' i, only one path may use the 0-edge and the other path is required to take the w_i-edge. The min-max objective function is optimized if and only if the weights are partitioned evenly and both paths have the same length. □

Corollary 15. *The Min-Max* 2 *Disjoint Paths Problem is (weakly)* **NP***-hard for planar graphs.*

4.2 Polynomial-time algorithm for tree-width-2 graphs

We first give a polynomial-time algorithm for outer-planar graphs. Recall that a graph is *outer-planar* if it can be drawn such that every vertex is incident to the outer face and no edges cross. Consider the cyclic order of terminals in clock-wise direction of an outer-planar graph. For $k \geq 2$ terminal pairs, if there is a feasible solution – that is, there are k disjoint paths – it may never happen that *only one* terminal of a pair (s_i, t_i) lies *between* the terminals of another pair (s_j, t_j), since their paths P_i and P_j would intersect. Without loss of generality, we may assume that the cyclic order of the two terminal pairs is (s_1, t_1, t_2, s_2). Note that a minimum cost flow algorithm can not optimize the min-max objective function. We name the sets of nodes between the terminals by $\mathcal{N}$orth, $\mathcal{S}$outh, $\mathcal{E}$ast, and $\mathcal{W}$est as follows: let $\mathcal{W}$ denote the nodes between s_1 and s_2, let $\mathcal{N}$ denote the nodes between s_1 and t_1, let $\mathcal{E}$ denote the nodes between t_1 and t_2,

and let $\mathcal{S}$ denote the nodes between s_2 and t_2. Nodes and edges from and to $\mathcal{N}$ may not be used by P_2 and, analogously, nodes in $\mathcal{S}$ are prohibited for P_2. Of course, every edge may be used by at most one path. Observe that if an edge between a node from $\mathcal{W}$ and a node from $\mathcal{E}$ is part of the optimal solution, then all nodes and edges 'below' this edge may only be part of P_2 and all edges 'above' may only be part of P_1. We split the problem of finding disjoint paths into a polynomial number of two independent shortest path problems. Every pair of nodes from $(\{s_2\} \cup \mathcal{W}) \times (\{t_2\} \cup \mathcal{E})$ defines a partition, which defines two vertex-induced subgraphs. The algorithm solves two independent shortest path problems, one for each subgraph.

The running time can be improved by a linear factor if, instead of node pairs, we consider edges between $\mathcal{W}$ and $\mathcal{E}$.

Algorithm 1. Input: an undirected outer-planar graph $G = (V, E)$, terminals $s_1, s_2, t_1, t_2 \in V$, w.l.o.g. in cyclic order (s_1, t_1, t_2, s_2).

Let $\mathcal{W}$ denote the nodes between s_1 and s_2, let $\mathcal{N}$ denote the nodes between s_1 and t_1, let $\mathcal{E}$ denote the nodes between t_1 and t_2, and let $\mathcal{S}$ denote the nodes between s_2 and t_2.

1. For each pair $(w, e) \in (\{s_2\} \cup \mathcal{W}) \times (\{t_2\} \cup \mathcal{E})$ such that $we \in E$ or $(w, e) = (s_2, t_2)$
   - Partition $\mathcal{W}$ into $\mathcal{W}_1$ containing the nodes between w and s_1 and $\mathcal{W}_2 = \mathcal{W} \setminus \mathcal{W}_1$
   - Partition $\mathcal{E}$ into $\mathcal{E}_1$ containing the nodes between e and t_1 and $\mathcal{E}_2 = \mathcal{E} \setminus \mathcal{E}_2$
   - Find a shortest path P_1 between s_1 and t_1 in $G_1 = G[\{s_1, t_1\} \cup \mathcal{N} \cup \mathcal{W}_1 \cup \mathcal{E}_1]$
   - Find a shortest path P_2 between s_2 and t_2 in $G_2 = G[\{s_2, t_2\} \cup \mathcal{S} \cup \mathcal{W}_2 \cup \mathcal{E}_2]$
   - Update the previous optimum (SP_1, SP_2) if (P_1, P_2) is a better solution
2. Return the optimal solution (SP_1, SP_2).

Theorem 16. *Algorithm 1 optimally solves the Min-Max* 2 *Disjoint Paths Problem for outer-planar graphs in time* $\mathrm{O}(n^2)$.

Proof. Time complexity: The number of edges between $\mathcal{W}$ and $\mathcal{E}$ is at most $\mathrm{O}(n)$. For each pair of endpoints the algorithm solves two independent shortest path problems in time $\mathrm{O}(n)$ [9].

Correctness: Each edge may be used by at most one path. If an edge between a node from $\mathcal{W}$ and a node from $\mathcal{E}$ is part of the optimal solution, then all nodes and edges 'below' this edge may only be part of P_2 and all edges 'above' may only be part of P_1. Therefore, by computing the solution for all possible partitions, the algorithm finds the optimal solution. □

In what follows, we reduce the Min-Max 2 Disjoint Paths Problem in graphs with tree-width 2 to the problem in outer-planar graphs.

Definition 2. A graph has *contracted tree-width 2* if it has tree-width 2 and the corresponding tree-decomposition $(T, \mathcal{V})$ satisfies that no pair of vertices is contained in more than two vertex sets of $\mathcal{V}$, that is $|V_{t_1} \cap V_{t_2} \cap V_{t_3}| \leq 1$ for any $t_1, t_2, t_3 \in V(T)$.

Lemma 17. *Graphs with contracted tree-width* 2 *are outer-planar.*

Proof. Proof is by induction. A graph consisting of three vertices is obviously outer-planar. Let $(T, \mathcal{V})$ be the tree-decomposition of $G = (V, E)$ with contracted tree-width 2. Take a leaf $l \in V(T)$ of T and let $V' = \bigcup_{t \neq l} V_t$. Then, by induction hypothesis, $G[V']$ is outer-planar. If $|V_l \cap V'| = 1$ then G is obviously outer-planar, and so we may assume that $V_l \cap V' = \{u, v\}$. If there exists an edge connecting u and v on the boundary of G', then G is outer-planar. Otherwise, a pair $\{u, v\}$ is part of two sets of $\mathcal{V} \setminus V_l$, which contradicts the definition of contracted tree-width. This shows that G remains outer-planar. □

We reduce the problem for a general graph with tree-width at most 2 to equivalent problems in a graph with contracted tree-width 2 (which is also outer-planar), which we then solve using Algorithm 1.

As the tree-width is at most 2, every set $V_t \in \mathcal{V}$ has cardinality at most three. The graph induced by V_t and the edges of the original graph may consist of one or two edges or it may form a triangle. The overlap with another set $V_{t'}$ consists of at most two vertices, $|V_t \cap V_{t'}| \leq 2$ and all these small graphs are linked forming a tree. In the tree decomposition $(T, \mathcal{V})$, we update the length of all edges uv that lie in more than two sets in $\mathcal{V}$ by deleting a subgraph or by renaming the terminals.

Algorithm 2. Input: a graph $G = (V, E)$ with tree-width at most 2, terminals s_1, s_2, t_1, t_2.

1. Compute a tree-decomposition $(T, \mathcal{V})$ of G (see [2]). We may assume that $V_i \neq V_j$ for distinct $i, j \in V(T)$.

2. While there exist pairs $\{u, v\}$ with $\exists V_{t_1}, V_{t_2}, V_{t_3}, t_1 \neq t_2 \neq t_3 \neq t_1, \{u, v\} \subseteq V_{t_1} \cap V_{t_2} \cap V_{t_3}$ compute $G - \{u, v\}$ in which there are at least three distinct components $G_{i_1}, G_{i_2}, \ldots, G_{i_p}$.

   - If some component G_i contains no terminal, remove G_i, add an edge uv (if uv does not exist), and update $l(uv) \leftarrow \min\{l(uv), d_{G_i+u+v}(u, v)\}$.
   - If all $G_{i_1}, G_{i_2}, \ldots, G_{i_p}$ contain terminals, then $p = 3, 4$ and two subgraphs, say G_{i_1}, G_{i_2}, contain one terminal each. Rename the terminals such that the terminal in G_{i_1} is s_1. Compute $d_{G_{i_1}+u}(s_1, u)$ and $d_{G_{i_1}+v}(s_1, v)$ and do the same for the terminal in G_{i_2}. Remove G_{i_1} and G_{i_2} and create new instances as follows.

     If the terminals 'match', meaning that the terminal in G_{i_2} is t_1, recursively create four instances of the problem ($u = s_1, v = t_1$; $u = t_1, v = s_1$; $v = s_1 = t_1$; and $u = s_1 = t_1$) and return the minimal solution. Otherwise, if the terminals do not match, recursively create two instances of the problem and return the minimal solution.

3. The resulting graph is outer-planar. Solve the problem using Algorithm 1.

Theorem 18. *Algorithm 2 optimally solves the Min-Max* 2 *Disjoint Paths Problem for graphs with tree-width at most* 2 *in time* $\mathrm{O}(n^3)$.

Proof. Time complexity (very scarce estimates): There will be at most $\mathrm{O}(n)$ reductions as there are at most $\mathrm{O}(n)$ edges, and the second reduction in Step 2 occurs at most twice. Each reduction affects at most $\mathrm{O}(n)$ components. Finding the shortest path in each component takes time at most $\mathrm{O}(n)$ [9]. Therefore, all reductions take time at most $\mathrm{O}(n^3)$. The recursive call occurs at most twice with at most four instances each. This yields at most $4 \cdot 4 = \mathrm{O}(1)$ disjoint shortest path problems in an outer-planar graph, each of which can be solved in time $\mathrm{O}(n^3)$.

Correctness: Removing $\{u, v\}$ splits the graph into at least three component graphs, since T contains no cycle.

- If one component, say G_i, does not contain a single terminal s_1, s_2, t_1, t_2, update the edge length of uv by $\min\{l(uv), d_{G_i+u+v}(u, v)\}$. This works since only one path may pass through G_i.

- If G_i contains exactly one terminal, say s_1, remove G_i from G and solve the disjoint shortest path problem for two separate instances $u = s_1$ and $v = s_1$, add $d_{G_i+u}(s_1, u)$ and $d_{G_i+v}(s_1, v)$, respectively, and return the minimum solution. This works since P_2 cannot enter and leave G_i without interfering with P_1.

- If G_i contains two terminals, then reduce another subgraph G_j using the rules from above. At most two subgraphs may have this property and the resulting graph will have the desired property for the pair $\{u, v\}$.
- The remaining cases are symmetric.

After Step 2, no edge uv is in more than two sets V_i, V_j of the tree-decomposition, which is exactly the definition of contracted tree-width 2. The resulting graph is outer-planar by Lemma 17. By computing the solution for all possible pairs, the algorithm finds the optimal solution. □

4.3 Pseudo-polynomial-time algorithm for bounded tree-width graphs

As shown in Theorem 14, the Min-Max k Disjoint Paths Problem is NP-hard even if $k = 2$ and the tree-width of the input graph is at most three, whereas the Min-Sum k Disjoint Paths Problem can be solved in polynomial time in bounded tree-width graphs [18]. In this subsection, for fixed k, we give a pseudo-polynomial-time algorithm for the Min-Max k Disjoint Paths Problem in bounded tree-width graphs. Note that this technique also works for the weighted versions introduced in [26, 27].

Theorem 19. *Let $G = (V, E)$ be a graph whose tree-width is bounded by a fixed constant, and let $\ell : E \to \mathbb{Z}_+$ be an integer-valued length function. Then, for fixed k, the Min-Max k Disjoint Paths Problem can be solved in time polynomial in $|V|$ and $\ell(E)$.*

Proof. We introduce a new problem called the *weighted folio*, whose unweighted version is introduced in [15]. Let $G = (V, E)$ be a graph, let $\ell : E \to \mathbb{Z}_+$ be an integer-valued length function, and let $X \subseteq V$ be a vertex set. A pair $(\mathcal{X}, \vec{z})$ of a partition $\mathcal{X} = \{X_1, X_2, \ldots, X_p\}$ of X and an integer vector $\vec{z} = (z_1, z_2, \ldots, z_p) \in \mathbb{Z}_+^p$ is *realizable* if there are disjoint trees $T_1, T_2, \ldots, T_p$ in G such that $X_i \subseteq V(T_i)$ and $\sum_{e \in E(T_i)} \ell(e) = z_i$ for $i = 1, \ldots, p$. The weighted folio is the problem to enumerate all realizable pairs $(\mathcal{X}, \vec{z})$ in G. One can see that the solution of the Min-Max k Disjoint Paths Problem is immediately derived from that of the weighted folio in which $X = \{s_1, \ldots, s_k, t_1, \ldots, t_k\}$. That is, for a partition $\mathcal{X} = \{\{s_1, t_1\}, \ldots, \{s_k, t_k\}\}$ of X, it suffices to find a realizable pair $(\mathcal{X}, \vec{z})$ minimizing $\max_i(z_i)$. Note that the number of realizable pairs is at most $(|X|\ell(E))^{|X|}$, which is polynomial size of $\ell(E)$ if $|X|$ is fixed.

If the tree-width of the input graph is bounded by w, the weighted folio can be solved for each bag using the standard dynamic programming technique, which takes time polynomial in $|V|, (w + |X|)^{w+|X|}$, and $\ell(E)^{w+|X|}$ (see [1, 15]). This completes the proof. □

References

[1] Stefan Arnborg and Andrzej Proskurowski. Linear time algorithms for NP-hard problems restricted to partial k-trees. *Discrete Applied Mathematics*, 23(1):11–24, 1989.

[2] Hans L. Bodlaender. A linear-time algorithm for finding tree-decompositions of small treewidth. *SIAM Journal on Computing*, 25(6):1305–1317, 1996.

[3] Éric Colin de Verdière and Alexander Schrijver. Shortest vertex-disjoint two-face paths in planar graphs. In *STACS*, pages 181–192, 2008.

[4] Edsger Wybe Dijkstra. A note on two problems in connection with graphs. *Numerische Math.*, 1:269–271, 1959.

[5] Herbert Edelsbrunner and Ernst Peter Mücke. Simulation of simplicity: a technique to cope with degenerate cases in geometric algorithms. *ACM Transactions on Graphics*, 9(1):66–104, 1990.

[6] Steven Fortune, John E. Hopcroft, and James Wyllie. The directed subgraph homeomorphism problem. *Theoretical Computer Science*, 10:111–121, 1980.

[7] András Frank. *Paths, Flows, and VLSI-Layout*, chapter Packing paths, cuts and circuits – a survey, pages 49–100. Springer-Verlag, 1990.

[8] Rudolf Halin. S-functions for graphs. *Journal of Geometry*, 8(1-2):171–186, 1976.

[9] Monika Rauch Henzinger, Philip N. Klein, Satish Rao, and Sairam Subramanian. Faster shortest-path algorithms for planar graphs. *Journal of Computer and System Sciences*, 55(1):3–23, 1997.

[10] Alon Itai, Yehoshua Perl, and Yossi Shiloach. The complexity of finding maximum disjoint paths with length constraints. *Networks*, 12:277–286, 1981.

[11] Richard M. Karp. On the computational complexity of combinatorial problems. *Networks*, 5:45–68, 1975.

[12] Chung-Lun Li, S. Thomas McCormick, and David Simchi-Levi. The complexity of finding two disjoint paths with min-max objective function. *Discrete Applied Mathematics*, 26(1):105–115, 1990.

[13] James F. Lynch. The equivalence of theorem proving and the interconnection problem. *SIGDA Newsletter*, 5(3):31–36, 1975.

[14] Richard G. Ogier, Vladislav Rutenburg, and Nachum Shacham. Distributed algorithms for computing shortest pairs of disjoint paths. *IEEE Transactions on Information Theory*, 39(2):443–455, 1993.

[15] Neil Robertson and Paul D. Seymour. Graph minors. II. Algorithmic aspects of tree-width. *Journal of Algorithms*, 7:309–322, 1986.

[16] Neil Robertson and Paul D. Seymour. *Paths, Flows, and VLSI-Layout*, chapter An outline of a disjoint paths algorithm, pages 267–292. Springer-Verlag, 1990.

[17] Neil Robertson and Paul D. Seymour. Graph minors. XIII. The disjoint paths problem. *Journal of Combinatorial Theory, Series B*, 63(1):65–110, 1995.

[18] Petra Scheffler. A practical linear time algorithm for disjoint paths in graphs with bounded tree-width. Technical Report 396, Technische Universität Berlin, 1994.

[19] Alexander Schrijver. Finding k disjoint paths in a directed planar graph. *SIAM Journal on Computing*, 23(4):780–788, 1994.

[20] Paul D. Seymour. Disjoint paths in graphs. *Discrete Mathematics*, 29:293–309, 1980.

[21] Yossi Shiloach. A polynomial solution to the undirected two paths problem. *Journal of the ACM*, 27(3):445–456, 1980.

[22] Anand Srinivas and Eytan Modiano. Finding minimum energy disjoint paths in wireless ad-hoc networks. *Wireless Networks*, 11(4):401–417, 2005.

[23] Éva Tardos. A strongly polynomial minimum cost circulation algorithm. *Combinatorica*, 5(3):247–256, 1985.

[24] Carsten Thomassen. 2-linked graphs. *European Journal of Combinatorics*, 1:371–378, 1980.

[25] Bing Yang, Si-Qing Zheng, and Suresh Katukam. Finding two disjoint paths in a network with min-min objective function. In *IASTED International Conference on Parallel and Distributed Computing and Systems*, pages 75–80, 2003.

[26] Bing Yang, Si-Qing Zheng, and Enyue Lu. Finding two disjoint paths in a network with normalized α^--min-sum objective function. In *IASTED International Conference on Parallel and Distributed Computing and Systems*, pages 342–348, 2005.

[27] Bing Yang, Si-Qing Zheng, and Enyue Lu. Finding two disjoint paths in a network with normalized α^+-min-sum objective function. In *ISAAC*, pages 954–963, 2005.