Improved Algorithms for the 2-Vertex Disjoint Paths Problem

Torsten Tholey

20.10.08

Abstract

Given distinct vertices s_1, s_2, t_1, and t_2 the 2-vertex-disjoint paths problem (2-VDPP) consists in determining two vertex-disjoint paths p_1, from s_1 to t_1, and p_2, from s_2 to t_2, if such paths exist.

We show that by using some kind of sparsification technique the previously best known time bound of $O(n + m\alpha(m, n))$ can be reduced to $O(m+n\alpha(n, n))$, where α denotes the inverse of the Ackermann function. Moreover, we extend the very practical and simple algorithm of Hagerup for solving the 2-VDPP on 3-connected planar graphs to a simple linear time algorithm for the 2-VDPP on general planar graphs thereby avoiding the computation of planar embeddings or triconnected components.

Copyright

Improved Algorithms for the 2-Vertex Disjoint Paths Problem

Torsten Tholey

Institut für Informatik
Universität Augsburg
D-86135 Augsburg, Germany
tholey@informatik.uni-augsburg.de

Abstract. Given distinct vertices s_1, s_2, t_1, and t_2 the 2-vertex-disjoint paths problem (2-VDPP) consists in determining two vertex-disjoint paths p_1, from s_1 to t_1, and p_2, from s_2 to t_2, if such paths exist.
We show that by using some kind of sparsification technique the previously best known time bound of $O(n + m\alpha(m, n))$ can be reduced to $O(m+n\alpha(n, n))$, where α denotes the inverse of the Ackermann function. Moreover, we extend the very practical and simple algorithm of Hagerup for solving the 2-VDPP on 3-connected planar graphs to a simple linear time algorithm for the 2-VDPP on general planar graphs thereby avoiding the computation of planar embeddings or triconnected components.

1 Introduction

Given a tuple $I = (G, s_1, \dots, s_k, t_1, \dots, t_k)$ with G being a graph and $s_1, \dots, s_k$, $t_1, \dots, t_k$ being pairwise distinct vertices in G, the k-vertex-disjoint paths problem (k-VDPP) consists in determining k pairwise disjoint paths $p_i : s_i \to t_i$ $(i = 1, \dots, k)$. This problem is a fundamental routing problem with several practical applications as for example in the context of VLSI-design and network reliability (see [1] and [8]). Unfortunately, for $k \geq 3$, there are still no efficient algorithms for the k-VDPP. More precisely, Fortune, Hopcroft, and Wyllie [2] proved that even the decision version of the 2-VDPP, where we only want to test the existence of disjoint paths, is $\mathcal{NP}$-complete on directed graphs. For undirected graphs, the decision version of the k-VDPP is solvable in $O(n^3)$ time for any fixed k as shown by Robertson and Seymour [11]—as usual in graph theory we let n and m denote the number of vertices and edges, respectively, of the graph under consideration. Perković and Reed [9] improved the running time to $O(n^2)$. However, the constants hidden in the big-Oh-notation of these algorithms being very large, for $k \geq 3$, there is not yet any known algorithm for the k-VDPP of practical importance. For this reason, research on the k-VDPP has focused on restricted graphs [12, 13], the case $k = 2$ (see below), or both [6, 10]. From now on we will consider only undirected graphs.

Previous results. Perl and Shiloach [10] found the first linear time algorithm for the 2-VDPP on planar triconnected graphs. Woeginger [18] presented a much simpler algorithm but also used a planar embedding of the graph under consideration. Hagerup [3] was able to simplify this algorithm to a very practical algorithm on planar triconnected graphs avoiding the computation of planar embeddings. As already stated by Perl and Shiloach in [10], all these algorithms can be modified to solve the 2-VDPP on general planar graphs using a reduction of Itai. However, Itai's reduction makes use of a graph decomposition into triconnected components. Therefore, combining the algorithm of Hagerup with the reduction of Itai does not seem to result in an efficient practical algorithm.

For general undirected graphs, Ohtsuki [8], Seymour [14], Shiloach [15], and Thomassen [17] independently found the first polynomial time algorithms for the 2-VDPP. Seymour considered the decision version of the problem. The running time of Ohtsuki and Shiloach's algorithm is $O(nm)$. By modifying Shiloach's algorithm, Khuller, Mitchell, and Vazirani [5] could prove a running time of $O(n^2)$ for the 2-VDPP. The author of this paper also following Shiloach's approach reduced the running time to the currently best known time bound $O(n+m\alpha(m,n))$ (see [16]), where α denotes the inverse of the Ackerman function.

New results. In section 2 we sketch the $O(n+m\alpha(m,n))$-time algorithm presented in [16]. In section 3 we show that by using a sparsification technique the running time can be reduced to $O(m+n\alpha(n,n))$. Both algorithms use theoretically efficient but practically inefficient subalgorithms. However, suppose we are given a practical algorithm for finding in $f(m,n)$ time a subgraph isomorphic to a subdivision of the $K_{3,3}$ or K_5 (if such a subgraph exists). Moreover, assume there is a practical algorithm for constructing in $g(m,n)$ time a practical data structure supporting the following operations in $h(m,n)$ time: 1) the insertion of an edge, 2) the test, whether two vertices are 4-connected, and 3) the output of a 3-separator separating two vertices, if such a separator exits. Such algorithms may be in the focus of future research since they are of independent interest. Given such algorithms, a more practical version of the algorithm in [16] runs in $O(n+f(m,n)+g(m,n)+n\cdot h(m,n))$ time. The results of this paper reduce this time to $O(m+f(m,n)+g(n,n)+n\cdot h(n,n))$, which may be substantially smaller. (f, g, and h should be monotonically increasing functions).

Finally, in section 4 we extend the algorithm of Hagerup [3] for triconnected planar graphs to general planar graphs. This results in a simple algorithm for the 2-VDPP on planar graphs which avoids the computation of both, planar embeddings and triconnected components.

Further in this section we introduce some notations used in this paper. For a graph G, we define $E(G)$ as the edge set of G and $V(G)$ as the vertex set of G. We write $\deg_G(v)$ for the degree of v in G. Throughout this paper, paths will always mean simple paths, i.e., paths visiting each of its vertices at most once. Therefore, for a path p and two vertices a and b of p, we can denote by $p[a,b]$ the subpath of p, or of the reverse path of p, leading from a to b. For a vertex v, an edge e, and a path p, we write $v \in p$ or $e \in p$ if v or e, respectively, is part of p. By $p \circ q$ we denote the concatenation of two paths p and q. A path visiting vertices $v_1, v_2, \ldots, v_k$ in this order is sometimes also denoted by $v_1 - v_2 - \ldots - v_k$. A set

$\mathcal{P}$ of paths is called *internally disjoint* if each vertex v, visited by two different paths $p, q \in \mathcal{P}$, must be an endpoint of both, p and q.

For a graph G, we call a set $S \subseteq V(G)$ a *separator of* G if $G - S$ is not connected. A separator S is a *k-separator* if $|S| = k$. Vertices v and w are *separated* by $S \subseteq V(G)$ if they lie in different connected components of $G - S$. Vertices v and w are *k-connected* if there are k *internally disjoint paths* between v and w. From Menger's theorem it is well-known that non-adjacent vertices v and w are k-connected iff there is no $(k-1)$-separator separating v and w.

2 Solving the 2-VDPP in $O(n + m\alpha(m, n))$ Time

Itai (cf. [15]) has shown that an instance of the 2-VDPP can be reduced in linear time to an instance $I = (G, s_1, s_2, t_1, t_2)$ of the 2-VDPP with G being triconnected. By slightly modifying Shiloachs algorithm such an instance I can be solved in linear time if the following property P holds (see [15, 16]).

(P) All vertices in $G - \{s_1, s_2, t_1, t_2\}$ are 4-connected to x in $G_{x,I}$,

where here and in the following $G_{x,I}$ denotes the graph obtained from G by adding an extra vertex x as well as edges $(x, s_1), (x, s_2), (x, t_1)$, and (x, t_2).

Thus, it remains to reduce an instance of the 2-VDPP on a triconnected graph to an instance on a triconnected graph with property P. For such a reduction, we repeatedly make use of so-called Δ-replacements:

Let S be a 3-separator separating v and x in $G_{x,I}$. By a *Δ-replacement on G by S deleting v* we mean replacing G by the graph G' obtained from G by deleting all vertices of the connected component of $G - S$ containing v with their adjacent edges and by inserting edges between the vertices of S that are not already connected by an edge. Then the following Lemma holds (see [15, 16]).

Lemma 1. *Let $I = (G, s_1, s_2, t_1, t_2)$ be an instance of the 2-VDPP on a triconnected graph G and S be a 3-separator separating a vertex v and x in $G_{x,I}$. Then the graph G' resulting from a Δ-replacement on G by S deleting v is triconnected and contains s_1, s_2, t_1, t_2. Moreover, I has a solution iff the same is true of $I' = (G', s_1, s_2, t_1, t_2)$. Finally the degree of connectivity between x and any vertex w not being deleted by the Δ-replacement can be increased but not decreased by the replacement.*

In order to find vertices not 4-connected to x, the $O(n + m\alpha(m, n))$-time algorithm in [16] uses a data structure D of Kanevsky et al. [4]. This data structure on a triconnected graph supports the insertion of edges and it can answer queries, whether two vertices v and w in the current graph are 4-connected. Finally, if vertices v and w are neither adjacent nor 4-connected, the data structure can output a 3-separator separating v and w. One can use this data structure to find a vertex v in $G_{x,I} - \{s_1, s_2, t_1, t_2\}$ not 4-connected to x and a 3-separator S separating v and x, and then run a Δ-replacement by S deleting v. However,

D does not support edge deletions. Therefore, data structure D is not used for $G_{x,I}$, but for a graph H that only before the first Δ-replacement is equal to $G_{x,I}$. After a Δ-replacement the same edges are added to $G_{x,I}$ and H, but no edge of H is deleted. We can do so since at any step of our algorithm a vertex $v \in V(G)$ is 4-connected to x in $G_{x,I}$ iff it is 4-connected to x in H, and since each 3-separator S separating v from x in H also separates v and x in $G_{x,I}$ (see [16]). A high-level implementation of the reduction described above is shown in Fig. 1.

```
(1) For each $v \in V - \{s_1, s_2, t_1, t_2\}$
(2)     Using $D$ test whether $v$ is 4-vertex-connected to $x$ in $H$.
(3)     If $v$ is not 4-vertex-connected to $x$ in $H$:
(4)         Using $D$ find a 3-separator $S$ separating $v$ from $x$ in $H$.
(5)         Modify $G$ and $G_{x,I}$ by a $\Delta$-replacement by $S$ deleting $v$.
(6)         For each edge $e$ inserted into $G$ by the $\Delta$-replacement
(7)             Insert $e$ into $H$.
(8)             Update the dynamic data structure $D$.
(9) Return G
```

Fig. 1. Reducing an instance of the 2-VDPP to an instance with property P.

By Lemma 1, the algorithm shown in Fig. 1 determines a graph $\tilde{G}$ with property P that contains disjoint paths $p_1 : s_1 \to t_1$ and $p_2 : s_2 \to t_2$ iff the same is true for the original graph G. Given such paths in $\tilde{G}$, in some kind of backtracking steps, one can compute paths solving the original instance in $O(n + m)$ time (see [16]). We next analyze the running time of the reduction to $\tilde{G}$. D supports the initialization with a triconnected graph in $O(n + m\alpha(m, n))$ time and up to $O(n)$ queries, computations of 3-separators, and edges insertions in $O(n\alpha(m, n))$ time. Given v and S, the running time of a Δ-replacement by S deleting v is at most linear in the number of edges deleted by the replacement. The whole reduction runs in $O(n + m\alpha(m, n))$ time (compare [16]).

The main idea of our new algorithm is to replace the graph H by a *sparse certificate.* A *sparse certificate for the k-connectivity* of a graph G is a subgraph G_* of G with $V(G_*) = V(G)$ and $O(n)$ edges, where for each pair of vertices v and w of G, v and w are k-connected in G if and only if they are k-connected in G_*. A *sparse certificate for the k-connectivity to a vertex x* of a graph G containing x is a subgraph G_* of G with $V(G_*) = V(G)$ and $O(n)$ edges, where a vertex v is k-connected to x in G if and only if the same is true of G_*.

Unfortunately, updating a sparse certificate for the k-connectivity to a vertex x after a Δ-replacement is not easy. For this reason, we choose the vertices in line 1 of the algorithm in Fig. 1 in a particular order. Sometimes we also modify a vertex v and/or a 3-separator computed by the algorithm of Fig. 1. This should guarantee that H is some kind of sparse certificate of $G_{x,I}$ even after each update of H and $G_{x,I}$.

3 An $O(m + n\alpha(n, n))$-time Algorithm for the 2-VDPP

We present an $O(m+n\alpha(n, n))$-time reduction of an instance $I = (G, s_1, s_2, t_1, t_2)$ of the 2-VDPP on a triconnected graph G to an instance $I' = (G', s_1, s_2, t_1, t_2)$ with G' being a triconnected graph with property P (defined in the previous section). The reduction is divided into two phases.

The first phase dynamically updates $G_{x,I}$ and two graphs H and H_*. We initialize H with a copy of $G_{x,I}$ and H_* with a sparse ceritificate for the k-connectivity of H for all $k \leq 4$. Such a certificate can be constructed in linear time (see [7]). H is used only to simplify the proof of correctness. During each update, all edges inserted into $G_{x,I}$ are also inserted into H and H_*, but no edge will be deleted from H and H_*. We also construct the data structure of Kanevsky et al. [4] described in the previous section. However, we use it for H_* instead for G or H, and we call it D_*. Finally, we maintain a list L of all $v \in V(G)$ with $\deg_G(v) > 3$. Since a Δ-replacement only changes the degree of a constant number of vertices remaining in G—beside restricting L to the remaining vertices—L can be updated in $O(1)$ time after a Δ-replacement.

After initializing H, H_*, D_*, and L, we use D_* to test, for each vertex $v \in L$, whether it is 4-connected to x. If not, again using D_*, we determine a 3-separator S separating x and v in H_*. Similar to the algorithm of Fig. 1 we want to run a Δ-replacement by a 3-separator S', but we choose S' very carefully instead of just setting $S' = S$. Let K be the subgraph of H_* induced by the vertices of S and by the vertices of the connected component of $H_* - S$ containing v. Let S' be the set obtained from S by replacing each vertex $w \in S$ with $\deg_K(w) = 1$ by its unique neighbor $n(w)$ in K. A unique neighbor $n(w)$ of a vertex $w \in S$ with $\deg_K(w) = 1$ is neither equal to another vertex $u \in S$ nor to a unique neighbor $n(u)$ in K of a vertex u with $u \in S$. (Otherwise $S - \{w\}$ or $(S - \{u, w\}) \cup \{n(w)\}$ would be a 2-separator in the triconnected graph H_*). Therefore, we have $|S'| = 3$. We will later show that a 3-separator separating a vertex $w \in V(G)$ from x in H_* also separates x and w in $G_{x,I}$. From $K - S$ containing v and $\deg_G(v) > 3$, we then can conclude: With S separating v and x in $G_{x,I}$, set S' separates x in $G_{x,I}$ from a vertex v' equal to v or to a neighbor of v. Define K' to be the subgraph K' of H_* induced by the vertices of S' and by the vertices of the connected component of $H_* - S'$ containing v'. Then $\deg_{K'}(w) \geq 2$ holds for all $w \in S'$. We now run a Δ-replacement on $G_{x,I}$ by S' deleting v'. If v remains in $G_{x,I}$, we have $v \in S' - S$ and v is a neighbor of exactly one vertex $w \in S$. Hence, after the replacement v has exactly three neighbors, namely w and the vertices in $S' - \{v\}$. Since we have $\deg_{K'}(w) \geq 2$ for all $w \in S'$, the degree of a vertex remaining in $G_{x,I}$ cannot increase after a Δ-replacement by S'. This also being true of v and all following Δ-replacements, v is never reinserted to L in phase 1. Fig. 2 shows a high-level implemenation of phase 1.

If the following invariants (1) and (2) hold before and after each Δ-replacement, we can conclude from Lemma 1 and the considerations above that the following holds at the end of phase 1: $G_{x,I}$ is triconnected and (G, s_1, t_1, s_2, t_2) is solvable iff our original instance is solvable. Moreover, a vertex v in G is 4-connected to x in G iff $\deg_{G_{x,I}}(v) \neq 3$.

```
(1) Construct a list $L$ of all vertices $v$ with $\deg_G(v) \geq 4$.
(2) For each $v \in L$
(3)     Using $D_*$ test whether $v$ is 4-vertex-connected to $x$ in $H_*$.
(4)     If $v$ is not 4-vertex-connected to $x$ in $H_*$:
(5)         Using $D_*$ find a 3-separator $S$ separating $v$ and $x$ in $H_*$.
(6)         $K :=$ subgraph of $H_*$ induced by $S$ and by the vertices of
                the connected component of $H_* - S$ containing $v$.
(7)         For each $w \in S$
(8)             If $w$ has only one neighbor $u$ in $K$:
(9)                 $S := (S - \{w\}) \cup \{u\}$   /* replacement of $S$ by $S'$. */
(10)        Let $v'$ be an arbitrary vertex equal to $v$ or to a neighbor
                of $v$ separated by $S$ from $x$ in $G_{x,I}$.
(11)        Modify $G$ and $G_{x,I}$ by a $\Delta$-replacement by $S$ deleting $v'$.
(12)        For each edge $e$ inserted into $G$ by the $\Delta$-replacement
(13)            Insert $e$ into $H$ and $H_*$.
(14)            Update list $L$ and the dynamic data structure $D_*$.
(15) Return G
```

Fig. 2. A high-level implementation of phase 1.

(1) A vertex $v \in V(G)$ is 4-connected to x in $G_{x,I}$ iff the same is true for H_*,
(2) A 3-separator S separating a vertex $v \in V(G)$ and x in H_* also separates v and x in H and $G_{x,I}$.
(3) Two vertices v and w are 4-vertex connected in H iff the same is true of H_*.
(4) $G_{x,I}, H$, and H_* are triconnected.

However, in order to prove property (2), we need the properties (3) and (4):

Lemma 2. *Before or after a Δ-replacement, property (2) follows from the properties (3) and (4).*

Proof. Let us consider a series of Δ-replacements on G. Let F be a graph initialized with $G_{x,I}$ before the first replacement. Assume F is updated after each replacement by inserting the same edges as into G but not deleting any edge. Then, a 3-separator S separating a vertex $v \in V(G)$ from x in F also separates v and x in $G_{x,I}$ (see [16]). It remains to show that a 3-separator S separating a vertex v from x in H_* also separates v and x in H.

Suppose there is a 3-separator S separating a vertex v from x in H_* but not in H. Then there must be an edge $(u, w) \in E(H) \setminus E(H_*)$ with u and w lying in different components of $H_* - S$.

Since H_* is triconnected (property (4)) there are three internally disjoint paths from u to w in H using only edges of H_*. Edge (u, w) defines a fourth path from u to w in H so that we obtain a contradiction to (3). □

Lemma 3. *Properties (1), (3), (4) hold before and after each Δ-replacement.*

Proof. Properties (1), (3), and (4) hold before the first Δ-replacement since at this time H is a copy of $G_{x,I}$ and H_* is a sparse certificate of the k-connectivity for H for every $k \leq 4$.

Let us assume that the properties (1), (3), and (4) hold before a Δ-replacement on G by a 3-separator S' deleting a vertex v'. Let $G_{x,I}, H$, and H_* be the graphs before the Δ-replacement and $G'_{x,I}, H', H'_*$ be the corresponding graphs after the Δ-replacement.

Since no edge of H and H_* is deleted, H' and H'_* are triconnected. By Lemma 1 $G'_{x,I}$ is also triconnected. In [16] the author of this paper has shown that a vertex $v \in V(G'_{x,I})$ is 4-connected to x in $G'_{x,I}$ iff the same is true of H'. Hence, property (1) follows from property (3). It remains to show property (3).

Since $E(H'_*) \subseteq E(H')$, two vertices 4-connected in H'_* are also 4-connected in H'. For the reverse direction, let p_1, p_2, p_3, and p_4 be four internally disjoint paths in H' leading from a vertex u to a vertex w. It remains to show that there are also four internally disjoint paths from u to w in H'_*. Before showing that, let us define $K_{v'}$ to be the connected component of $G_{x,I} - S'$ deleted from $G_{x,I}$. In particular, $v' \in K_{v'}$. Moreover, let K_x be the union of all other components of $G_{x,I} - S'$ remaining in G. For each pair of vertices $y, z \in S'$ and each $C \in \{K_{v'}, K_x\}$, we define $p(y, C, z)$ to be a path from y to z in H_* that beside y and z only visits vertices in C. Such paths exist: There are three internally disjoint paths from v' to the vertices in S' only visiting vertices in $K_{v'} \cup S'$, since S' is a 3-separator in the triconnected graph H_*. We construct $p(y, K_{v'}, z)$ by combining the two of the three paths ending in y and z. The existence of a path $p(y, K_x, z)$ can be shown in a similar way. We now distinguish several cases depending on the cardinality of $S' \cap \{u, w\}$.

If $\{u, w\} \subseteq S'$, define y as the vertex in $S' \setminus \{u, w\}$. Then H'_* contains four internally disjoint paths from u to w in H'_*: $u - w$, $u - y - w$, $p(u, K_{v'}, w)$, and $p(u, K_x, w)$.

Let us next consider the case, where $\{u, w\} \not\subseteq S'$ and at most one path p out of p_1, p_2, p_3, and p_4 visits edges with either at least one endpoint in $K_{v'}$ or both endpoints in S'. Let a be the first and b be the last vertex of S' visited by p. By replacing $p[a, b]$ by $p(a, K_{v'}, b)$ we obtain four internally disjoint paths from u to w in H instead of H'. Since (3) holds before the Δ-replacement, there are also 4 internally disjoint paths from u to w in H_*. The same must be true of H'_* because of $(E(H_*) \subseteq E(H'_*))$.

We now examine the case $\{u, w\} \not\subseteq S'$ with two paths p, q out of the four internally disjoint paths p_1, p_2, p_3, p_4 from u to w visiting edges with at least one endpoint in $K_{v'}$ or both endpoints in S'. This can only be the case if $|\{u, w\} \cap S'| = 1$. Say, w.l.o.g., $w \in S'$. Let a and b be the first vertices of S' visited by p or q, respectively. Suppose that we can show that the subpaths $p[a, w]$ and $q[b, w]$ can be replaced by two internally disjoint paths p', from a to w, and q', from b to w in $H - K_x$. Then, similar to the previously considered case, we obtain four internally disjoint paths from u to w in H and hence also in H_* and H'_*.

Guaranteeing the existence of p' and q' is the reason for dividing our algorithm into two phases. Let K be the subgraph K of H induced by the vertices of $K_{v'}$ and S'. We have already shown on page 5 that by our choice of v' and S' we must have $\deg_K(x) \geq 2$ for all $x \in S'$. (More precisely, we considered a subgraph K of H_* instead of a subgraph K of H). If (a, w) is an edge in H, we choose $p' := (a, w)$ and $q' = p(b, K_{v'}, w)$. Similarly, if (b, w) is an edge in H, we choose $p' := p(a, K_{v'}, w)$ and $q' := (b, w)$.

If neither the edge (w, a) nor the edge (w, b) exists in H, w is connected to two distinct vertices c, d in $K - S'$. In K there are three internally disjoint paths $p_{a,c}, p_{b,c}, p_{w,c}$ leading from a, b, or w, respectively, to c as well as three internally disjoint paths $p_{a,d}, p_{b,d}, p_{w,d}$ leading from a, b, or w to d. Choose $\tilde{p}$ as one of the two paths $p_{a,d}$ and $p_{b,d}$ such that $\tilde{p}$ does not visit c. Say w.l.o.g. $\tilde{p} = p_{a,d}$. If $\tilde{p}$ is internally disjoint to both, $p_{a,c}$ and $p_{b,c}$, we define $p' := \tilde{p} \circ (d, w)$ and $q' := p_{b,c} \circ (c, w)$. Otherwise, let y be the last vertex of $p_{a,d}$ which is also visited by $p_{a,c}$ or $p_{b,c}$. If $y \in p_{a,c}$, we set $p' := p_{a,c}[a, y] \circ \tilde{p}[y, d] \circ (d, w)$ and $q' := p_{b,c} \circ (c, w)$. If $y \in p_{b,c}$, we set $p' := p_{a,c} \circ (c, w)$ and $q' := p_{b,c}[b, y] \circ \tilde{p}[y, d] \circ (d, w)$. □

At the end of phase 1 a vertex v in $G_{x,I}$ is 4-connected to x iff $\deg_{G_{x,I}}(v) \geq 4$. We start phase 2 similar to phase 1 with constructing a copy H of $G_{x,I}$ and a sparse certificate H_* for the k-connectivity of H for all $k \leq 4$. Immediately after initializing H and H_*, we color all vertices $v \in V(G_{x,I})$ with $\deg_{G_{x,I}}(v) \geq 4$ black and all $w \in V(G_{x,I})$ with $\deg_{G_{x,I}}(w) = 3$ red. The color of a vertex will never be updated in phase 2. According to the triconnectivity of G the vertices s_1, s_2, t_1, and t_2 are 4-connected to x in $G_{x,I}$ and hence are colored black.

We proceed as follows for deleting the remaining vertices of $G_{x,I}$ that are not 4-connected to x: We run the algorithm of Fig. 1, but we replace each occurrence of H in this algorithm by H_*. Additionally—for the sake of analysis—in line 7 we insert e into both, H and H_*, instead only into H_*. This works if the following invariant (2) holds. We prove that (2) holds by showing that both invariants, (1) and (2), hold before and after each Δ-replacement.

(1) Each edge of $G_{x,I}$ adjacent to a red colored vertex is also contained in H_*.
(2) A vertex $v \in V(G)$ is 4-connected to x in H_* if and only if the same is true of $G_{x,I}$. Moreover, every 3-separator separating a vertex $v \in V(G)$ from x in H_* also separates v and x in H and $G_{x,I}$.

Lemma 4. *Invariant (1) holds before and after each Δ-replacement.*

Proof. H_* being initialized with a sparse certificate for the 3-connectivity of the original graph $G_{x,I}$, the current graph H_* contains all edges of the original graph $G_{x,I}$ adjacent to a red colored vertex. All other edges adjacent to a red colored vertex in the current graph $G_{x,I}$ have been inserted by a Δ-replacement. By construction, they are also inserted into H_*. □

Corollary 5. *Invariant (2) holds before and after each Δ-replacement.*

Proof. As shown in [16], a vertex is 4-connected to x in $G_{x,I}$ iff the same is true for H. Therefore, a vertex $v \in V(G)$ 4-connected to x in H_* is also 4-connected to x in H and $G_{x,I}$. Assume now that there is a 3-separator S separating a vertex $v \in V(G)$ and x in H_* but not in $G_{x,I}$ (or not in H). Since during the Δ-replacements the degree of connectivity between two vertices cannot decrease in H_*, v is a red vertex. Let p be a path from v to x in $G_{x,I} - S$ (or in $H - S$) and let b be the first black vertex on this path. Vertex b exists since s_1, s_2, t_1, and t_2 are black. According to invariant (1) $p[v, b]$ is contained in H_*. For b as a black vertex, there must exist four internally disjoint paths from b to x in H_* with one of them not visiting any vertex of S. Hence, there is a path from v to x in $H_* \setminus S$. This is a contradiction to the fact that S separates v and x in H_*. $\square$

At the end of phase 2 we obtain a graph $G_{x,I}$ with property P. This follows from the correctness of the algorithm of Fig. 1 and from invariant (2). The analysis of the running time is nearly the same as in [16]. The difference is that the term $O(m\alpha(m, n))$ caused by the usage of the dynamic data structure of Kanevsky et al. can be reduced to $O(n\alpha(n, n))$ since D_* is used for graphs with $O(n)$ edges. Hence the algorithm runs in $O(m + n\alpha(n, n))$ time.

4 Solving the 2-VDPP on Planar Graphs

In order to solve the 2-VDDP on planar graphs, we show how an instance $I = (G, s_1, s_2, t_1, t_2)$ of the 2-VDPP on a planar graph can be reduced to an instance $I' = (G', s'_1, s'_2, t'_1, t'_2)$ on a planar graph G' with three internally disjoint paths between the vertices s'_1 and t'_1 as well as three internally disjoint paths between s'_2 and t'_2. We then apply the algorithm of Hagerup which indeed also runs correctly on this weaker class of graphs (without changing any proof in [3]).

Say w.l.o.g. that G is connected. If there are less than three internally disjoint paths between s_1 and t_1, we will split our graph into smaller subgraphs: For a graph H and a separator S separating vertices v and w in H, let $H_S(v)$ be the graph obtained from the subgraph of H induced by the vertices of S and by the vertices of the connected component C of $H - S$ containing v if we insert edges between the vertices of S that are not already connected. Similarly, $H_S(\neg v)$ should be the graph obtained from the subgraph of H induced by the vertices of $V(H) \setminus V(C)$ if we insert edges between all vertices of S that are not already connected. In order to split an instance into smaller instances, we will apply the following lemma:

Lemma 6. *Let an arbitrary but fixed integer $k \geq 1$ and two non-adjacent vertices s and t in a graph G be given. Then one can find either $(k+1)$ internally disjoint paths from s to t in $O(m)$ time or a k-separator S with the following property in $O(|E(G_S(s))|)$ time: S separates s and t, but in $G_S(s)$ no vertex $w \in S$ is separated by a k-separator from s.*

By splitting instances of the 2-VDPP according to Lemma 6, we obtain the following lemmas. Here we will only prove the more complicated Lemma 8.

Lemma 7. *Suppose that the 2-VDPP can be solved in linear time on instances (G, s_1, s_2, t_1, t_2), where G is a planar graph with two internally disjoints paths between s_1 and t_1 as well as two internally disjoint paths between s_2 and t_2. Then the 2-VDPP on general planar graphs can also be solved in linear time.*

Lemma 8. *Suppose that the 2-VDPP can be solved in linear time on instances (G, s_1, s_2, t_1, t_2), where G is a planar graph with three internally disjoint paths between s_1 and t_1 as well as two internally disjoint paths between s_2 and t_2. Then the 2-VDPP on general planar graphs can also be solved in linear time.*

Proof. According to Lemma 7 we focus on an instance $I := (G, s_1, s_2, t_1, t_2)$ on a planar graph G with two internally disjoint paths between s_1 and t_1 as well as two internally disjoint paths between s_2 and t_2. W.l.o.g. I is *non-simple*, i.e., $s_1 \neq t_1$ and $s_2 \neq t_2$. If also three internally disjoint paths p_1, p_2, and p_3 from s_1 to t_1 exist, they can be constructed with the classical algorithm of Ford and Fulkerson. Afterwards, by our assumption the 2-VDPP can be solved in linear time. If three internally disjoint paths do not exist, we construct only two internally disjoint paths p_1 and p_2 from s_1 to t_1 and recursively split the original instance in several rounds into smaller instances as follows.

Starting with the original instance I, during our algorithm we will always encounter either exactly one instance $I_1 = (H, s'_1, s'_2, t_1, t'_2)$ or two instances $I_1 = (H, s'_1, s'_2, t_1, t'_2)$ and $I_2 = (H, s''_1, s'_2, t_1, t'_2)$. Immediately after the initialization, we have $H = G$ and $s'_1 = s_1$. After the first split, we always have $H = G_T(\neg s_1)$, where T is a 2-separator separating s_1 and t_1 computed in a previous *splitting round*. We will also have $s'_1 \in T$, and if I_2 exists, $T = \{s'_1, s''_1\}$. In each splitting round, applying Lemma 6 we will search for a 2-separator S separating s'_1 and t_1 and hence also separating s_1 and t_1. We use S to split instance I_1 as well as I_2, if the later exists. In particular, this means we try to find solutions for I_1 and I_2 simultaneously. Let us assume that we are faced with one or two instances I_1 or I_1 and I_2 as described above and that (except in Case 4) by applying Lemma 6 we found a set $S = \{u, w\}$ separating s'_1 and t_1. Choose i, j such that $u \in p_i$, $w \in p_j$, and $\{i, j\} = \{1, 2\}$.

Case 1 $H_S(\neg s'_1) = G_S(\neg s_1)$ contains both, s'_2 and t'_2. In this case we will always have $s'_2 = s_2$ and $t'_2 = t_2$. We reduce our problem to the instances $I_3 = (G_S(\neg s'_1), u, s'_2, t_1, t'_2)$ and $I_4 = (G_S(\neg s'_1), w, s'_2, t_1, t'_2)$. Assume we are given two paths p and q solving the 2-VDPP on one of these instances with p leading from u or w, respectively, to t_1. Then, depending on the exact subcase, p can by extended by $p_i[s'_1, u]$, $(s'_1, s''_1) \circ p_i[s''_1, u]$, $p_j[s'_1, w]$, or $(s'_1, s''_1) \circ p_j[s''_1, w]$ to a solution of I_1. More precisely, since we may have $(u, w) \notin E(G)$, p should not visit (u, w). However, if p visits both, u and w, p can be shortend by removing $p[u, w]$ or $p[w, u]$, respectively; after the removal p and q still define a solution to one of the instances I_3 and I_4. Because of edge (s'_1, s''_1) we can also find two paths solving I_2 with none of these paths visiting both, u and w. Conversely, if neither I_3 nor I_4 has a solution, neither I_1 nor I_2 can have a solution.

Case 2 $H_S(\neg s'_1)$ contains exactly one of s'_2 and t'_2, say w.l.o.g. t'_2. For two paths p and q solving the 2-VDPP on I_1 (or on I_2), the parts of the paths p and q contained in $H_S(s'_1)$ must solve the 2-VDPP on either $J_1 = (H_S(s'_1), s'_1, s'_2, u, w)$ or $J_2 = (H_S(s'_1), s'_1, s'_2, w, u)$ (or on either $J_3 = (H_S(s'_1), s''_1, s'_2, u, w)$ or $J_4 = (H_S(s'_1), s''_1, s'_2, w, u)$). Let $(H_S(s'_1), \tilde{s}_1, \tilde{s}_2, \tilde{t}_1, \tilde{t}_2)$ be the instance J_x for an $x \in \{1, \ldots, 4\}$. We next show that solving J_x is easier than solving I_1 or I_2: Solving J_x is trivial, if $(\tilde{s_1}, \tilde{t_1})$ or $(\tilde{s_2}, \tilde{t_2})$ is an edge of $H_S(s'_1)$. Otherwise, by construction, there is no 2-separator separating s'_1 from $\tilde{t}_1$ (Lemma 6). In particular, there must be three internally disjoint paths between s'_1 and $\tilde{t}_1$ and hence J_1 and J_2 can be solved in linear time. For the instances J_3 and J_4, we know that a 2-separator S' separating $\tilde{s}_1 = s''_1$ and $\tilde{t}_1$ must contain s'_1. Assume, there is such a 2-separator $S' = \{s'_1, w'\}$. Then J_3 and J_4 have to be solved recursively by our reduction algorithm. However, when splitting J_x into instances on the graphs $H_{S'}(s''_1)$ and $H_{S'}(\neg s''_1)$ we will show that we only have to search for a solution of an instance of the 2-VDPP on $H_{S'}(s''_1)$ with the path starting in s''_1 leading to w' and not to s'_1. Therefore, on $H_{S'}(\neg s''_1)$ we have to search for a solution with the path ending in $\tilde{t}_1$ starting in w' and not in s'_1. This means that for solving J_x we need not to solve two but only one instance on $H_{S'}(s''_1)$ as well as on $H_{S'}(\neg s''_1)$. Note that the instance on $H_{S'}(s''_1)$ can be solved in linear time since there is no 2-separator separating s''_1 and w'. The instance on $H_{S'}(\neg s''_1)$ can be recursively handeld in the same way as J_x. Hence, our reduction algorithm splits J_x only in a linear number of instances of the 2-VDPP on subgraphs that beside the constructed 2-separators are distinct. Therefore, a solution for J_3 as well as for J_4 can be found in linear time. It remains to show that on the subgraph $H_{S'}(s''_1)$ we do not need to consider solutions of the 2-VDPP with s''_1 leading to s'_1. This is obvious if s'_1 is equal to either s'_2 or t'_2. Otherwise, because of s'_2 and t'_2 lying in $H - \{s'_1, s''_1\}$ and because of $T := \{s'_1, s''_1\}$ being a 2-separator determined in a previous step, in the step finding the 2-separator T, we must have been in Case 1. In Case 1 we must have reduced our original problem to the problem of solving one of our instances $I_1 = (G_T(\neg s_1), s'_1, s'_2, t_1, t'_2)$ and $I_2 = (G_T(\neg s_1), s''_1, s'_2, t_1, t'_2)$. It is easy to see that with a solution to I_1 or I_2 there also exist paths solving I_1 or I_2 with none of these paths visiting both, s'_1 and s''_1.

If, for $x \in \{1, 3\}$, there are paths p'_1 and q'_1 solving J_x, but no two paths p'_2 and q'_2 solving J_{x+1}, it is easy to see that $I_{\lceil \frac{x}{2} \rceil}$ can be reduced to $J'_1 := (H_S(\neg s'_1), u, w, t_1, t'_2)$. Accordingly, for $x \in \{2, 4\}$, if there are paths p'_1 and q'_1 solving J_x, but no two paths p'_2 and q'_2 solving J_{x-1}, instance $I_{\frac{x}{2}}$ can be reduced to $J'_2 := (H_S(\neg s'_1), w, u, t_1, t'_2)$. Finally, if, for $x \in \{2, 4\}$, we find paths solving J_x and paths solving J_{x-1}, the only remaining problem is to find two disjoint paths from the vertices w and u to the vertices t_1 and t'_2 in $H_S(\neg s'_1)$. This problem can be solved in $O(|E(H_S(\neg s'_1))|)$ time with network flow algorithms.

Case 3 $H_S(\neg s'_1)$ contains neither s'_2 nor t'_2. By symmetry Case 3 can be solved very similar to Case 1.

Case 4 Finally, assume that there is no 2-separator separating s'_1 and t_1. Then I_1 can be solved in linear time. However, as in Case 2, for solving I_2, we should split I_2 recursively in a linear number of instances on graphs that beside the nodes of a 2-separator are pairwise disjoint. □

Theorem 9. *The 2-VDPP on a planar graph can be solved in linear time without computing a combinatorial embedding of G or the triconnected components of G.*

Proof. The reduction algorithm used in our proof of Lemma 8 is applied twice in order to guarantee the existence of three internally disjoint paths between s_1

and t_1 (in the first run) as well as between s_2 and t_2 (in the second run). It is easy to show that the second run maintains the existence of three internally disjoint paths between s_1 and t_1 if $\{s_1, t_1\}$ does not separate s_2 and t_2. However, in the latter case it is clear that paths solving the 2-VDPP do not exist. $\square$

References

1. A. Aggarwal, J. Kleinberg, and D. P. Williamson, Node-disjoint paths on the mesh and a new trade-off in VLSI layout, *SIAM J. Comput.* **29** (2000), pp. 1321–1333.
2. S. Fortune, J. Hopcroft, and J. Wyllie, The directed subgraph homeomorphism problem, *Theoret. Comput. Sci.* **10** (1980), pp. 111–121.
3. T. Hagerup, A very practical algorithm for the two-paths problem in 3-connected planar graphs, Proc. 33rd International Workshop on Graph-Theoretic Concepts in Computer Science (WG 2007), Lecture Notes in Computer Science Vol. 4769, Springer-Verlag, Berlin, 2007, pp. 145–150.
4. A. Kanevsky, R. Tamassia, G. Di Battista, and J. Chen, On-line maintenance of the four-connected components of a graph, Proc. 32nd Annual IEEE Symposium on Foundations of Computer Science (FOCS 1991), pp. 793–801.
5. S. Khuller, S. G. Mitchell, and V. V. Vazirani, Processor efficient parallel algorithms for the two disjoint paths problem and for finding a Kuratowski homeomorph, *SIAM J. Comput.* **21** (1992), pp. 486–506.
6. C. L. Lucchesi and M. C. M. T. Giglio, On the irrelevance of edge orientations on the acyclic directed two disjoint paths problem, IC Technical Report DCC-92-03, Universidade Estadual de Campinas, Instituto de Computação, 1992.
7. H. Nagamochi and T. Ibaraki, Linear time algorithms for finding a sparse k-connected spanning subgraph of a k-connected graph. *Algorithmica* **7** (1992), pp. 583–596.
8. T. Ohtsuki, The two disjoint path problem and wire routing design, Proc. Symposium on Graph Theory and Algorithms, Lecture Notes in Computer Science, Vol. 108, Springer, Berlin, 1981, pp. 207–216.
9. L. Perković and B. Reed, An improved algorithm for finding tree decompositions of small width, *International Journal of Foundations of Computer Science (IJFCS)* **11** (2000), pp. 365–371.
10. Y. Perl and Y. Shiloach, Finding two disjoint paths between two pairs of vertices in a graph, *J. ACM* **25** (1978), pp. 1–9.
11. N. Robertson and P. D. Seymour, Graph minors. XIII. The disjoint paths problem, *J. Comb. Theory, Ser. B*, **63** (1995), pp. 65–110.
12. P. Scheffler, A practical linear time algorithm for disjoint paths in graphs with bounded tree-width, Report No. 396/1994, TU Berlin, FB Mathematik, 1994.
13. A. Schrijver, Finding k disjoint paths in a directed planar graph, *SIAM J. Comput.* **23** (1994), pp. 780–788.
14. P. D. Seymour, Disjoint paths in graphs, *Discrete Math.* **29** (1980), pp. 293–309.
15. Y. Shiloach, A polynomial solution to the undirected two paths problem, *J. ACM* **27** (1980), pp. 445–456.
16. T. Tholey Solving the 2-disjoint paths problem in nearly linear time. *Theory Comput. Systems* **39** (2006), pp. 51-78.
17. C. Thomassen, 2-linked graphs, *Europ. J. Combinatorics* **1** (1980), pp. 371–378.
18. G. Woeginger, A simple solution to the two paths problem in planar graphs, *Inform. Process. Lett.* **36** (1990), pp. 191–192.