ELSEVIER Discrete Applied Mathematics 85 (1998) 113–138

DISCRETE APPLIED MATHEMATICS

The disjoint shortest paths problem

Tali Eilam-Tzoreff*

Faculty of Commerce and Business and Administration, University of British Columbia, 2053 Main Hall, Vancouver, BC V6T 1Z2, Canada

Received 28 November 1995; revised 17 July 1997; accepted 4 August 1997

Abstract

The disjoint shortest paths problem is defined as follows. Given a graph G and k pairs of distinct vertices (s_i, t_i), $1 \leqslant i \leqslant k$, find whether there exist k pairwise disjoint shortest paths P_i between s_i and t_i for all $1 \leqslant i \leqslant k$. We may consider directed or undirected graphs and the paths may be vertex or edge disjoint. We show that these four problems are NP-complete when k is part of the input even for planar graphs with unit edge-lengths. We give a polynomial algorithm for the two disjoint shortest paths problem (vertex and edge disjoint paths) in undirected graphs with positive edge-lengths. We also consider the following variation of the problem. Given a graph and two distinct pairs of vertices, find whether there exist two disjoint paths P_1, P_2 between them such that P_1 is a shortest path. We show that this problem is NP-complete for undirected graphs with unit edge-lengths. This result is surprising in view of the existence of polynomial algorithms for both the two disjoint paths problem and the two disjoint shortest paths problem for undirected graphs.

1. Introduction

The *k disjoint paths* (kDP) problem is extensively studied. This problem is defined as follows. Given a graph $G=(V,E)$ and k distinct pairs of vertices $(s_1,t_1),\ldots,(s_k,t_k)$. Find whether there exist k pairwise disjoint paths $P_1,\ldots,P_k$ such that P_i is a path connecting s_i and t_i, for each $1 \leqslant i \leqslant k$. Of course, one may consider directed or undirected graphs, vertex-disjoint or edge-disjoint paths.

In this paper we consider disjoint paths problems with some additional constraints on the paths lengths. We consider the kDSP problem which is actually the k disjoint paths problem with the constraint that the paths should be shortest paths. More formally, given a graph $G=(V,E)$ and k pairs of distinct vertices (s_i,t_i) find whether there exist k pairwise disjoint shortest paths P_i between s_i and t_i for all $1 \leqslant i \leqslant k$. We show that all four versions of the kDSP problem (vertex or edge disjoint paths for directed or undirected graphs) for a graph with unit edge-lengths, are NP-complete when k is part of the input even for planar graphs. We give polynomial algorithms for the undirected

* E-mail: eilam@interchg.ubc.ca.

0166-218X/98/$19.00
PII S0166-218X(97)00121-2

2DSP problem for both vertex and edge disjoint paths. These are $O(|V|^8)$ all-quadruples algorithms. We also give an $O(|V|^8)$ algorithm for the weighted 2DSP problem. In this problem we are given an undirected graph and, in addition, to their lengths the edges are assigned weights. (We may assign weights to the vertices as well.) Find a solution to the 2DSP problem of minimal weight.

The 2D1SP problem is another variation of the two disjoint paths problem. The 2D1SP problem is the two disjoint paths problem with the constraint that only one specified path should be a shortest path. Since the 2DSP problem and the two disjoint paths problem in undirected graphs are polynomially solvable, one may expect that this problem is polynomially solvable too but this is not true. We show that this problem is NP-complete for all four versions of the problem for a graph with unit edge-lengths.

Hassin and Megiddo [4] considered the *ideal orientation* problem which is defined as follows. Given an undirected graph G and k pairs of vertices $(s_1,t_1),\dots,(s_k,t_k)$ find whether there exists an orientation G' of G such that the length of the shortest path from s_i to t_i in G is equal to the length of the shortest path from s_i to t_i in G', $1 \leqslant i \leqslant k$. They showed that when k is part of the input the problem is NP-complete, they gave a polynomial algorithm for $k=2$ while the complexity for fixed $k \geqslant 3$ remains an open problem. We show the relation between the two ideal orientation problem and the 2DSP problem. We give another polynomial algorithm to the two ideal orientation problem. It considers all the ideal orientations. Using the weighted 2DSP algorithm we can find an ideal orientation with minimum number of common edges of the two paths. We also give a simple polynomial algorithm to the orientation problem related to the 2D1SP problem. That is, given an undirected graph G and two pairs of vertices (s_1,t_1), (s_2,t_2) find whether there exists a *feasible* orientation G' of G such that the length of the shortest path from s_1 to t_1 in G is equal to the length of the shortest path from s_1 to t_1 in G'. A feasible orientation is an orientation in which there exists a directed path P_i from s_i to t_i.

Directed kDP. Fortune, Hopcroft, and Wyllie [2] considered the *fixed subgraph homeomorphism* problem. For a fixed graph P, given a graph G and a node mapping, does G contain a subgraph homeomorphic to P? They showed that the directed version of the problem (P and G are directed graphs) is NP-complete for all pattern graphs except those whose edges are either incoming edges to one vertex or out-going edges from one vertex. So the directed k vertex-disjoint paths problem is NP-complete for each fixed $k \geqslant 2$. A slight change of their proof gives a proof for directed graphs for which each vertex has either in-degree one or out-degree one. Consequently, we get NP-completeness of the directed k edge-disjoint paths problem for each fixed $k \geqslant 2$ as well.

Undirected kDP. In the undirected case Seymour [22], Shiloach [24], and Ohtsuki [14] gave different polynomial algorithms for the two vertex-disjoint paths problem. Later, Gustedt [3] gave an $O(|E| \log |V|)$ algorithm which improved the $O(|E||V|)$ algorithm of Shiloach [24]. Robertson and Seymour, in a series of papers [16] showed that the

k vertex-disjoint paths problem is in P for any fixed k. In undirected graphs a vertex-disjoint polynomial-time algorithm implies an edge-disjoint polynomial-time algorithm so the k edge-disjoint paths problem is in P for any fixed k as well.

It was shown by Karp [6] that the undirected k vertex-disjoint paths problem is NP-complete when k is part of the input.

Planar undirected kDP. Lynch [12] showed that the undirected k vertex-disjoint paths problem when k is part of the input remains NP-complete for planar graphs. Middendorf and Pfeiffer [13] showed that the planar undirected k disjoint paths problem is NP-complete for both vertex-disjoint and edge-disjoint paths when k is part of the input.

Planar directed kDP. The planar directed k vertex-disjoint paths problem is NP-complete when k is part of the input. (This follows from the NP-completeness of the planar undirected k vertex-disjoint paths problem.) Schrijver [20] showed that the planar directed k vertex-disjoint paths problem is solvable in polynomial time for each fixed k.

The edge-disjoint paths problem is a special case of the multi-commodity integral flow problem. Even, Itai and Shamir [1] showed that the two-commodity integral flow is NP-complete for both the directed and undirected case. Seymour [23] proved that the two-commodity integral flow in planar graphs is in P. This was extended later by Korach [7] for $k=3$ and Sebö [21] for any fixed k.

Itai et al. [5] and Li et al. [9] considered the *min–max k paths* problem. In this problem, we have to find k disjoint paths from s to t such that the maximum of their lengths is minimized. Li et al. [9] showed that all four versions for a graph with unit edge-lengths are NP-complete for fixed $k \geqslant 2$. If instead of finding k disjoint paths from s to t of min–max length, we have to find k such paths between k distinct pairs of vertices $(s_1,t_1),\dots,(s_k,t_k)$, the problem remains NP-complete for fixed $k \geqslant 2$.

The rest of this paper is organized as follows. In Section 2 we show that all four versions of the 2D1SP problem for a graph with unit edge-lengths, are NP-complete. In Section 3 we show that all four versions of the kDSP problem for a planar graph with unit edge-lengths, are NP-complete when k is part of the input. In Section 4.1 we give a polynomial algorithm for the undirected vertex-disjoint 2DSP problem. In Section 4.2 we give a polynomial algorithm for the undirected edge-disjoint 2DSP problem. In Section 4.3 we give a polynomial algorithm for the weighted 2DSP problem. In Section 5 we consider orientation problems related to the 2D1SP problem and the 2DSP problem. In Section 5.1 we give a polynomial algorithm to the orientation problem related to the 2D1SP problem. In Section 5.2 we show the relation between the two ideal orientation problem and the 2DSP problem.

2. The two disjoint one shortest path problem

Given a graph G with positive edge lengths and two pairs of vertices (s_1,t_1), (s_2,t_2) find whether there exist two disjoint paths P_1 from s_1 to t_1 and P_2 from s_2 to t_2

such that P_1 is a shortest path. We denote this problem in short 2D1SP. A specific instance of this problem is denoted by 2D1SP (s_1,t_1), (s_2,t_2) where the path between s_1 and t_1 should be a shortest path. We prove that the four versions of the problem are NP-complete for a graph with unit edge-lengths. We first prove that for an undirected graph the 2D1SP problem is NP-complete. The proof for the directed case is similar.

Claim 1. *Both the vertex and edge-disjoint versions of the 2D1SP problem on an undirected graph are* NP-*complete.*

Proof. The 2D1SP problem clearly belongs to NP. We give a polynomial reduction from 3SAT. For each instance of 3SAT we construct a graph G such that the given expression is satisfiable iff there exists a solution to 2D1SP (s_1,t_1), (s_2,t_2) in G. Let m be the number of clauses and n the number of variables in the expression. For each clause $c_i=(x_i \vee y_i \vee z_i)$, $1 \leqslant i \leqslant m$, we construct a subgraph C_i and for each variable v_j, $1 \leqslant j \leqslant n$, we construct a subgraph V_j as can be seen in Fig. 1. The numbers denote the edge lengths and since they are all positive integers each edge can be replaced by a path with unit length edges. The bold edges in C_i correspond to the literals x_i, y_i, z_i. These edges will be referenced later as e_1, e_2 and e_0, where e_1 is the leftmost and e_0 is rightmost. One path from d_j to d_{j+1} stands for v_j and the other for its complement $\bar{v}_j$. The number of edges in each d_j-d_{j+1} path is twice the maximum between the number of appearances of v_j and the number of appearances of $\bar{v}_j$ in the expression.

We add an edge between d_{n+1} and a_1.

We add the vertices $y_1,\ldots,y_{6m}$. We construct paths of length $12m-1$ between y_1 and y_{6m} using $y_1,\ldots,y_{6m}$ (see Fig. 2). The paths consist of subpaths of length four between y_{2i-1} and y_{2i+1}, $1 \leqslant i \leqslant 3m-1$, and a path of length three between y_{6m-1} and y_{6m}. For all $1 \leqslant i \leqslant 3m$, we take the edge $e_{i \bmod 3}$ in $C_{\lceil i/3 \rceil}$ (which stands for some literal, say l) and add two edges from its two endpoints to y_{2i-1} and y_{2i}. Let v_j be the variable which corresponds to that literal l. In the subgraph V_j we select an edge corresponding to l which was not connected yet by an edge to some y_i. We add two edges from its two endpoints to y_{2i-1} and y_{2i}. Note that the paths of length $12m-1$ between y_1 and y_{6m} which use all the y_i's are shortest paths. Fig. 2 shows the graph we get for the expression $(x_1 \vee x_2 \vee \bar{x}_3) \wedge (\bar{x}_1 \vee x_2 \vee \bar{x}_3)$. Claim 2 completes the proof of this claim. □

Claim 2. *The expression is satisfiable iff there exists a solution to 2D1SP* (y_1,y_{6m}), (d_1,a_{m+1}).

Proof. We first show that the existence of a solution implies satisfiability.

If there exists a solution P_1,P_2 to 2D1SP (y_1,y_{6m}), (d_1,a_{m+1}) then P_1 is a shortest path and it uses all the vertices $y_1,\ldots,y_{6m}$. Since P_2 is disjoint (vertex or edge) to P_1,

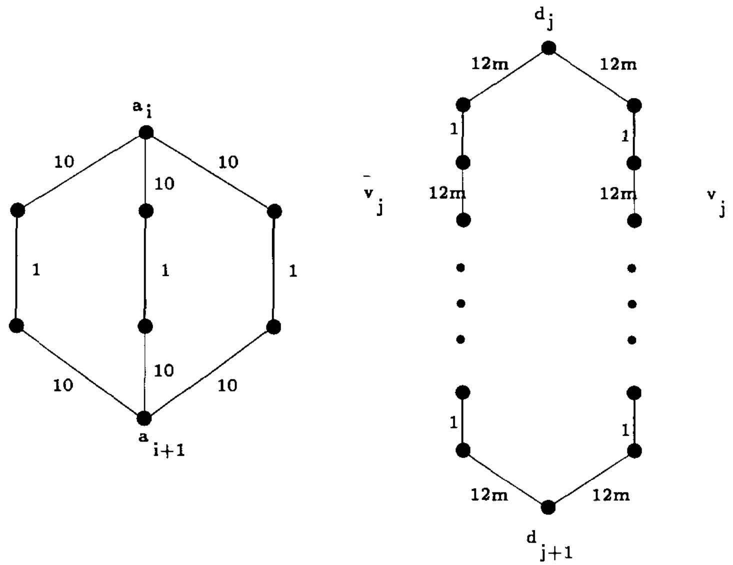

Fig. 1. The subgraphs C_i (left) and V_j (right).

it cannot use these vertices so it passes only through vertices of C_i and V_j. In V_j it traverses one of the two possible paths. If it intersects the path which stands for v_j we assign the variable v_j *false* value. Otherwise, if it chooses the $\bar{v}_j$ path we assign v_j *true* value. We show that this assignment satisfies the expression. Without loss of generality, suppose P_2 used in V_j the v_j path ($v_j \leftarrow false$) then P_1 must traverse all the edges in C_i which correspond to v_j, $1 \leqslant i \leqslant m$. Since P_2 has to pass through all the C_i subgraphs in order to reach a_{m+1}, it cannot use any of these v_j edges. We see that P_2 can use only those edges in C_i which stand for literals with *true* values, $1 \leqslant i \leqslant m$. Since P_2 uses one edge in each C_i there is at least one *true* value literal in each C_i and the expression is satisfiable.

To show that the satisfiability of the expression implies the existence of a solution, we choose the paths P_1, P_2 as follows: P_2 passes in each V_j, $1 \leqslant j \leqslant n$, through the path which corresponds to v_j if v_j is *false* or through $\bar{v}_j$ if v_j is *true*. In each C_i it passes through edges which correspond to *true* value literals. This is possible since there is at least one in each clause. A shortest path P_1 disjoint to P_2 can be chosen as follows: between each of the following pairs of vertices (y_{2i-1}, y_{2i+1}), $1 \leqslant i \leqslant 3m-1$, and (y_{6m-1}, y_{6m}) there are two shortest paths. One crosses some C_i and the other some V_j, both cross in an edge representing the same literal, say l. We have chosen P_2 in such a way that it uses at most one of these two edges, so P_1 uses the other. If l is *false* then P_2 uses the edge in V_j but not in C_i. If l is *true* then P_2 does not use the edge in V_j. P_1 is the concatenation of the shortest paths between y_{2i-1} and y_{2i+1}, $1 \leqslant i \leqslant 3m-1$, and between y_{6m-1} and y_{6m}. □

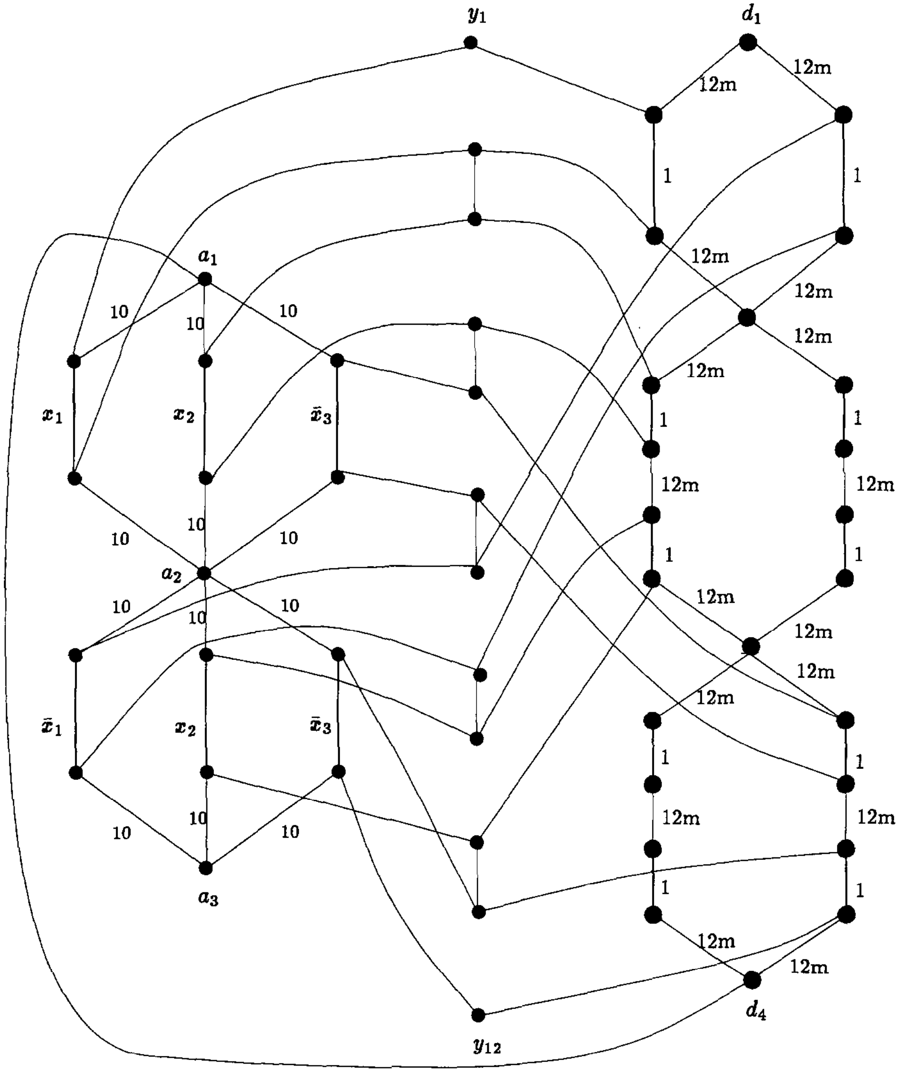

Fig. 2. The graph constructed for $(x_1 \vee x_2 \vee \bar{x}_3) \wedge (\bar{x}_1 \vee x_2 \vee \bar{x}_3)$.

Claim 3. *Both the vertex and edge-disjoint versions of the 2D1SP problem for a directed graph are* NP-*complete.*

Proof. The directed version of the 2D1SP problem clearly belongs to NP as well. We use a similar polynomial reduction from 3SAT but now we build a directed graph G. The underlying graph of G is exactly as in Claim 1. Its edges are directed with accordance to the direction of P_1 from y_1 to y_{6m} and P_2 from d_1 to a_{m+1}. The expression is satisfiable iff there exists a solution to 2D1SP (y_1, y_{6m}), (d_1, a_{m+1}) in this directed graph. $\square$

3. The k disjoint shortest paths problem

Given a graph G and k pairs of distinct vertices (s_i, t_i), find whether there exist k pairwise disjoint shortest paths P_i between s_i and t_i, for all $1 \leqslant i \leqslant k$. A straightforward modification of the reduction given in the proof of Claim 2.1 can be used to show that the four versions of the kDSP problem are NP-complete when k is part of the input. However, we now provide a stronger result. We show that these problems are NP-complete even when we restrict ourselves to *planar* graphs with unit edge-lengths.

Claim 4. *Both the vertex and edge-disjoint versions of the kDSP problem for planar undirected graphs are* NP-*complete when k is part of the input.*

Proof. To prove this for both the vertex-disjoint and edge-disjoint versions we prove NP-completeness of the vertex-disjoint version for planar undirected graphs of maximum degree three. For such graphs the edge-disjoint and vertex-disjoint versions are identical. The problem belongs to NP and we use a reduction from planar 3SAT.

The planar 3SAT is a restriction of 3SAT to expressions y for which the graph $G(y)$ described below is planar. $G(y)$ is a bipartite graph. The vertices in one part stand for the clauses of y and the vertices in the other part stand for the variables occurring in y. There exists an edge in $G(y)$ between the vertices v and C iff in y the variable v occurs in the clause C. Planar 3SAT is NP-complete [11]. Middendorf and Pfeiffer [13] observe that planar 3SAT remains NP-complete even when restricted to expressions in which every variable occurs in exactly three clauses. In such instances of planar 3SAT each clause contains either two or three literals. Clauses with only one literal are not considered. (In such a case the appropriate variable is assigned a value such that the clause is set true, all the clauses that were set true are deleted and from the rest the variable is omitted.) We may also assume that every variable occurs in the expression at least once positively and once negatively. We restrict ourselves to such instances of planar 3SAT. For each such expression y with n variables and m clauses (m_1 with three literals and m_2 with two literals), we build a planar graph $G_1(y)$ which is an instance of the vertex-disjoint $(2m_1 + m_2 + n)$DSP problem.

For each variable v in the expression y which occurs in the clauses A, B and C we build a planar gadget G_v in $G_1(y)$ which is contained in a triangle whose vertices are v_A, v_B, v_C.

For each clause $C = (v \vee w \vee x)$ we build a planar gadget G_C in $G_1(y)$ which is contained in a triangle whose vertices are v_C, w_C, x_C.

For each clause $C = (x \vee w)$ we build a much simpler planar gadget G_C which is contained between two parallel edges connecting the vertices x_C, w_C.

We identify the vertices v_C in the gadgets G_v and in G_C to get the graph $G_1(y)$. $G_1(y)$ is a planar graph. It is, in fact, the line graph of $G(y)$ which is a planar graph of maximal degree three.

In order to get a graph of maximal degree three we replace the vertices v_C by edges (v, C) whose endpoints are of degree three. There is only one way to perform the

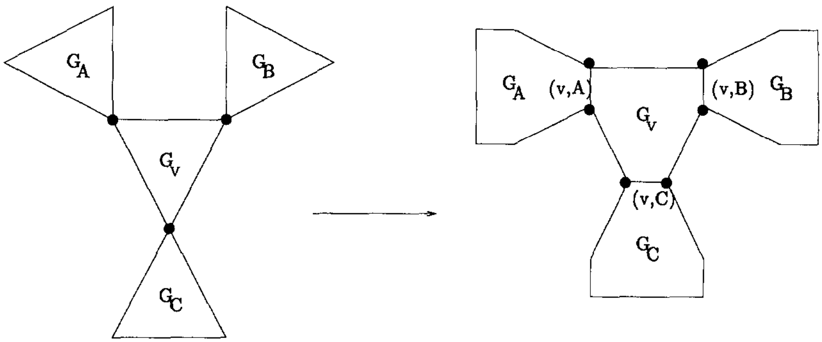

Fig. 3. A scheme of $G_1(y)$.

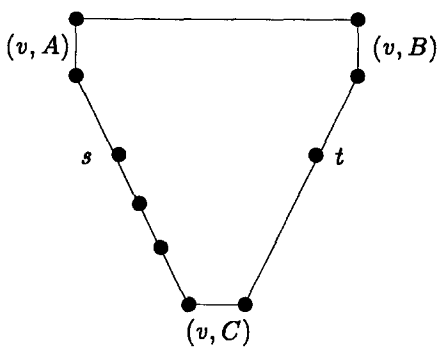

Fig. 4. The gadget G_v.

replacement such that the edge (v, C) belongs to both G_v, G_C and without affecting planarity as illustrated in Fig. 3. G_v is given in Fig. 4 and the gadgets G_C for both cases where the clause C consists of either two or three literals are given in Fig. 5. G_v is constructed so that there exist two shortest paths between s and t. One path passes through the edges (v,A), (v,B) where A and B are clauses in which v occurs positively (without loss of generality, we assume that v occurs twice positively and once negatively). The other path passes through the edge (v, C) where C is the clause in which v occurs negatively. In the case where C consists of three literals there exist two shortest paths of length five in G_C between s and t. The length of the shortest paths between s_1 and t_1 is seven. Note that there exist vertex-disjoint shortest paths $s - t$, $s_1 - t_1$ in G_C. These paths use at least one of the edges (v, C), (w, C), (x, C). Furthermore, two such vertex-disjoint shortest paths exist in G_C even when two of those three edges are not to be used. In the case where C consists of two literals we have only one pair of vertices in G_C. There exists an s–t shortest path in G_C. Here too, such a path uses at least one of the edges (x, C), (w, C) and it exists even when only one of those edges is to be used.

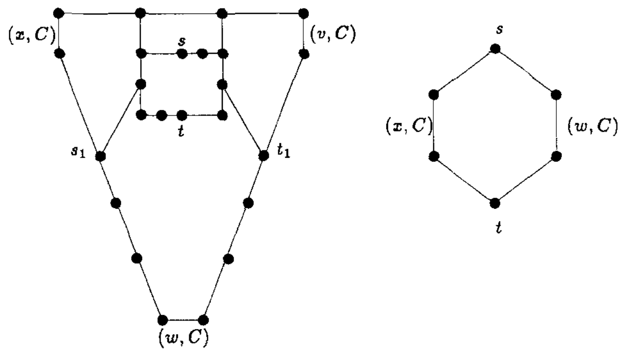

Fig. 5. The gadgets G_C.

$G_1(y)$ is a planar graph of maximal degree three with $(2m_1 + m_2 + n)$ pairs of vertices. Recall that m_1, m_2 are the numbers of the clauses in the expression which consist of three or two literals, respectively. Note that each such pair always occurs in one gadget and all the shortest paths between its endpoints use only edges of the same gadget. Claim 5 completes the proof of the claim. □

Claim 5. *An instance y of planar 3SAT is satisfiable iff there exists a solution to the vertex-disjoint $(2m_1 + m_2 + n)$DSP in $G_1(y)$.*

Proof. For simplification we assume throughout the proof that $m_1 = m$ and $m_2 = 0$. We first show that the existence of a solution to $(2m + n)$DSP in $G_1(y)$ implies satisfiability of y.

Denote such a solution by $P_1, \dots, P_n, Q_1, \dots, Q_{2m}$. P_i is a shortest path between the vertices s and t in the gadget G_{v_i}, $1 \leqslant i \leqslant n$. Q_{2i} and Q_{2i-1} are the shortest paths in the gadget G_{C_i}, $1 \leqslant i \leqslant m$. If P_i passes through an edge (v_i, A) where A is a clause in which v_i occurs positively, we assign v_i false value. If A is a clause in which v_i occurs negatively, we assign v_i true value. In the gadget G_{C_j} there exist two vertex disjoint shortest paths Q_{2j}, Q_{2j-1}. They use at least one of the edges (x, C_j), (w, C_j), (v, C_j). Without loss of generality, assume (v, C_j) is used. So the shortest path in G_v does not use this edge.

If v occurs negatively in C_j then the shortest path in G_v used an edge (v, A) where A is a clause in which v appears positively, v was assigned false value and, therefore, C_j is satisfied.

If v occurs positively in C_j then the shortest path in G_v used an edge (v, A) where A is a clause in which v appears negatively, v was assigned true value and, therefore, C_j is satisfied.

To show that satisfiability implies the existence of a solution to $(2m + n)$DSP in $G_1(y)$ we construct a solution $P_1, \dots, P_n$, $Q_1, \dots, Q_{2m}$ as follows. If the variable v_i has true value, we choose a path P_i in G_{v_i} which uses the edges (v_i, C), where C is a clause in which v_i appears negatively. If v_i has false value, we choose a path P_i in G_{v_i} which uses the edges (v_i, C), where C is a clause in which v_i appears positively. Since this is a truth assignment, for each clause $C = (x \vee w \vee v)$ at least one of the edges (x, C), (w, C), (v, C) in G_C was not used by the paths P_i. As we mentioned above there exist two vertex-disjoint shortest paths in G_C even if they may use only one of these edges. □

We consider now the kDSP problem for planar directed graphs. Given a planar directed graph G and k pairs of distinct vertices (s_i, t_i), find whether there exist k pairwise disjoint directed shortest paths P_i from s_i to t_i, for all $1 \leqslant i \leqslant k$.

This problem belongs to NP for both its vertex and edge-disjoint versions.

Claim 6. *The vertex-disjoint kDSP for planar directed graphs is* NP-*complete.*

Proof. We give a simple reduction from the vertex-disjoint planar undirected kDSP problem. Given an instance of this problem we replace each edge of the given undirected planar graph G by two anti-parallel directed edges. There exists a solution to the vertex-disjoint kDSP problem in the resulting directed planar graph G' iff there exists a solution to the vertex-disjoint kDSP problem in G. (Note that such a reduction is not applicable for the edge-disjoint version.) □

Claim 7. *The edge-disjoint kDSP for planar directed graphs is* NP-*complete.*

Proof. We use a similar reduction from planar 3SAT. Again we restrict ourselves to instances in which every variable occurs in exactly three clauses. For each variable v in the expression y we build a similar gadget G_v. We direct the edges of G_v from s to t. For each clause $C = (v \vee w \vee x)$ we build a planar directed gadget which is contained in a triangle whose vertices are v_C, w_C, x_C. For each clause $C = (w \vee x)$ we build a planar directed gadget which is contained between two parallel edges whose vertices are w_C, x_C. To get the graph $G_1(y)$ we identify the vertices v_C in the gadgets G_v and in G_C. As we did in the proof of Claim 3.1 we replace these vertices v_C by edges (v, C) without affecting planarity. The new edges (v, C) are directed in accordance to the direction in the gadgets G_v. That is, we direct them so that there exist two directed shortest paths from s to t.

Let $C = (v \vee w \vee x)$. The edges (v, C), (w, C), (x, C) which were given direction already may be all directed in the same direction or not. In Fig. 6 we give the gadget G_C for both possibilities.

For the first we specify three pairs of terminal vertices (s, t), (s_1, t_1), (t_1, s_1). There exist two shortest paths of length four from s to t. There exist two shortest paths of length seven from s_1 to t_1. One uses the edge (w, C) and the other uses none of (v, C), (w, C), (x, C). There exist five shortest paths of length seven from t_1 to s_1. One uses the edge (x, C), another uses the edge (v, C), one path uses both (x, C) and (v, C), and two others use none of (v, C), (w, C), (x, C). Note that there exist in G_C

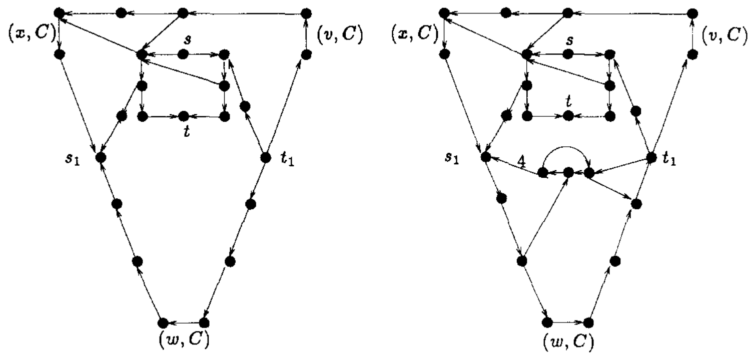

Fig. 6. The directed gadgets G_C.

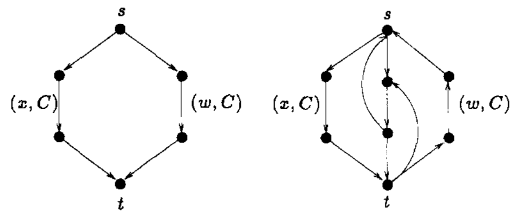

Fig. 7. The directed gadgets G_C.

three edge-disjoint shortest paths between the three pairs of terminals specified above even when two of the three edges (v,C), (w,C), (x,C) are not to be used, but at least one of these edges should be used.

For the latter we specify two pairs of vertices (s,t), (t_1,s_1). The shortest paths between them are identical to those in the right gadget except an additional t_1–s_1 shortest path which uses the edge (w,C). Here too there exist two edge-disjoint shortest paths between the two pairs of terminals even when two of the three edges, (v,C), (w,C), (x,C) are not to be used, but at least one of these edges should be used.

Note that these figures correspond to the case where at least two of the edges (v,C), (w,C), (x,C) are directed counterclockwise. If at least two are directed clockwise we should take the mirror image of the above gadgets. If $C=(w \vee x)$ the two possibilities for the gadget G_C are as in Fig. 7. For the gadget on the right we specify two pairs of vertices (s,t), (t,s). □

4. The two disjoint shortest paths problem

In this section we prove the following theorem.

Theorem 8. *The undirected vertex-disjoint 2DSP problem, the undirected edge-disjoint 2DSP problem, and the weighted 2DSP problem are polynomially solvable.*

In the rest of this section we denote by 2DSP (s_1,t_1), (s_2,t_2) the following problem. Given an undirected graph $G=(V,E)$ with positive edge-lengths and two pairs of distinct vertices (s_1,t_1) and (s_2,t_2) find whether there exist two disjoint shortest paths P_1 between s_1 and t_1, P_2 between s_2 and t_2. We also denote by $L(x,y)$ the subgraph of G consisting of the vertices and edges lying on shortest paths between any two vertices x and y. For convenience, we assume that the endpoints x,y are not in this subgraph. We also denote by $l(x,y)$ the length of an x–y shortest path.

4.1. *The two vertex-disjoint shortest paths problem*

In this subsection we give a polynomial algorithm for the vertex-disjoint 2DSP problem. An important case to which the algorithm refers later is when both $s_1,t_1 \in L(s_2,t_2)$ and both $s_2,t_2 \in L(s_1,t_1)$. That is, $l(s_2,s_1)+l(s_1,t_2)=l(s_2,t_1)+l(t_1,t_2)=l(s_2,t_2)$ and $l(s_1,s_2)+l(s_2,t_1)=l(s_1,t_2)+l(t_2,t_1)=l(s_1,t_1)$. Our first goal in this subsection is to analyze this case. This is done in the four following claims: Claims 9–12. In all four we assume that this case holds.

Claim 9. *There is no s_1–t_1 shortest path that meets both $L(s_1,s_2)$ and $L(s_1,t_2)$ or both $L(t_1,s_2)$ and $L(t_1,t_2)$.*

Proof. Assume that there exists an s_1–t_1 shortest path P_1 that meets both $L(s_1,s_2)$ and $L(s_1,t_2)$. Assume w.l.o.g. that $L(s_1,t_2)$ is first met by P_1 after $L(s_1,s_2)$ was met by P_1. Let a be the last vertex in $L(s_1,s_2)$ which P_1 traverses (from s to t) before it first meets $L(s_1,t_2)$ and let b be the first vertex in $L(s_1,t_2)$ which P_1 meets; see Fig. 8. Note that a and b may be the same vertex but both a and b are not s_1 so $l(s_1,a)$, $l(s_1,b)>0$. Since P_1 is a shortest path $l(s_1,b)=l(s_1,a)+l(a,b)>l(a,b)$ we get that $l(a,b)<l(a,s_1)+l(s_1,b)$. We consider now an s_2–t_2 path, P_2, which consists of the following three subpaths: An s_2–a shortest path, then the (a,b) subpath of P_1 (which is an (a,b) shortest path) and a (b,t_2) shortest path. Then, $l(P_2)=l(s_2,a)+l(a,b)+l(b,t_2)<l(s_2,a)+l(a,s_1)+l(s_1,b)+l(b,t_2)=l(s_2,s_1)+l(s_1,t_2)=l(s_2,t_2)$. We get that the length of P_2 is less then the length of a shortest s_2–t_2 path. This is a contradiction.

The proof for an s_1–t_1 shortest path which meets both $L(t_1,s_2)$ and $L(t_1,t_2)$ follows by symmetry. □

Claim 10. $L(s_1,s_2)$, $L(s_2,t_1)$, $L(t_1,t_2)$, $L(t_2,s_1)$ *are pairwise disjoint.*

Proof. From Claim 9 and a similar claim for an s_2–t_2 shortest path we get that each of the following pairs of subgraphs are disjoint. $L(s_1,s_2)$ and $L(s_2,t_1)$, $L(s_2,t_1)$ and $L(t_1,t_2)$, $L(t_1,t_2)$ and $L(t_2,s_1)$, $L(s_1,s_2)$ and $L(t_2,s_1)$. We prove now that $L(s_1,s_2)$ and $L(t_1,t_2)$ are disjoint as well. If $L(s_1,s_2)$ and $L(t_1,t_2)$ are not disjoint then there exists a vertex b which belongs to both of them (see Fig. 9). An s_1–b subpath of $L(s_1,s_2)$ is an s_1–b shortest path. An s_1–b path which consists of an s_1–t_2 shortest path followed by

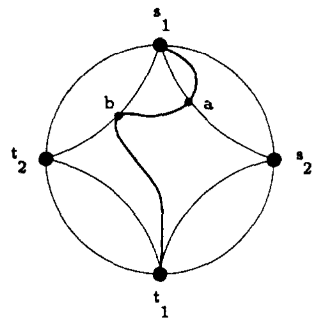

Fig. 8.

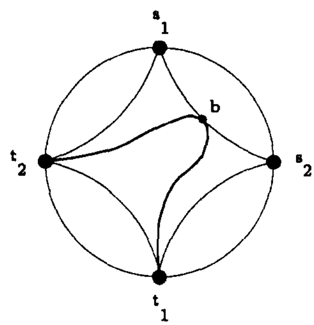

Fig. 9.

a t_2–b subpath of $L(t_2,t_1)$, is also an s_1–b shortest path. So $l(s_1,b)=l(s_1,t_2)+l(t_2,b)>l(t_2,b)$ and this implies that $l(t_2,b)<l(t_2,s_1)+l(s_1,b)$. We get that there exists an s_2–t_2 path, P_2, which consists of a t_2–b subpath of $L(t_1,t_2)$ and a (b,s_2) subpath of $L(s_1,s_2)$. Then,

$$\begin{aligned} l(P_2) &= l(t_2,b)+l(b,s_2)<l(t_2,s_1)+l(s_1,b)+l(b,s_2) \\ &= l(t_2,s_1)+l(s_1,s_2)=l(s_2,t_2). \end{aligned}$$

We get that the length of P_2 is less than the length of a shortest s_2–t_2 path. This is contradiction to our assumption that $L(s_1,s_2)$ and $L(t_1,t_2)$ are not disjoint. The proof of the disjointness of $L(s_2,t_1)$ and $L(t_2,s_1)$ follows by symmetry. □

Claim 11. *An s_1–t_1 shortest path which is not disjoint to $L(s_1,s_2)$ or $L(s_2,t_1)$ is disjoint to t_2. An s_1–t_1 shortest path which is not disjoint to $L(t_1,t_2)$ or $L(t_2,s_1)$ is disjoint to s_2. An s_2–t_2 shortest path which is not disjoint to $L(s_1,s_2)$ or $L(t_2,s_1)$ is disjoint to t_1. An s_2–t_2 shortest path which is not disjoint to $L(s_2,t_1)$ or $L(t_1,t_2)$ is disjoint to s_1.*

Proof. If we have an s_1–t_1 shortest path, P, which uses t_2 then the s_1–t_2 subpath of P belongs to $L(t_2,s_1)$ and the rest of it belongs to $L(t_1,t_2)$. By Claim 9, such a path is disjoint to $L(s_1,s_2)$ and $L(s_2,t_1)$. The proof of the other cases follows by symmetry. □

Claim 12. *Suppose $x \in L(s_1,t_1)$ but $x \notin L(s_1,s_2)$ and $x \notin L(t_2,s_1)$ then an (x,t_1) shortest path is disjoint to both s_2 and t_2.*

Proof. The proof is immediate. □

We say that a quadruple (x,y), (u,v) is *adjacent* to (s_1,t_1), (s_2,t_2) if $x,y \in L(s_1,t_1)$, x and y are adjacent in $L(s_1,t_1)$ to s_1 and t_1, respectively, and $l(s_1,x)+l(x,y)+l(y,t_1)=l(s_1,t_1)$. Similarly, $u,v \in L(s_2,t_2)$, u and v are adjacent in $L(s_2,t_2)$ to s_2 and t_2, respectively, and $l(s_2,u)+l(u,v)+l(v,t_2)=l(s_2,t_2)$. That is, there exist an s_1–t_1 shortest path which uses both edges (s_1,x) and (y,t_1) and an s_2–t_2 shortest path which uses both edges (s_2,u) and (v,t_2).

Claim 13. *There exists a solution P_1, P_2 to 2DSP (s_1,t_1), (s_2,t_2) in G iff there exists a solution Q_1, Q_2 to 2DSP (x,y), (u,v) for a quadruple (x,y), (u,v) adjacent to (s_1,t_1), (s_2,t_2), such that $s_2,t_2 \notin Q_1$, $s_1,t_1 \notin Q_2$.*

Proof. The proof is immediate. □

Claim 13 suggests a recursive algorithm for 2DSP (s_1,t_1), (s_2,t_2). If $s_1 \notin L(s_2,t_2)$ for every vertex $x \in L(s_1,t_1)$ adjacent in $L(s_1,t_1)$ to s_1, check whether there exists a solution Q_1,Q_2 to the 2DSP (x,t_1), (s_2,t_2). If for such a vertex x there exists a solution Q_1,Q_2 then it can be extended to a solution to the 2DSP (s_1,t_1), (s_2,t_2) by adding the edge (s_1,x) to Q_1. If for all such x there does not exist a solution to 2DSP (x,t_1), (s_2,t_2) then there does not exist a solution to 2DSP (s_1,t_1), (s_2,t_2). If $s_1 \in L(s_2,t_2)$ but $t_1 \notin L(s_2,t_2)$ or $t_2 \notin L(s_1,t_1)$ or $s_2 \notin L(s_1,t_1)$ we perform similar checks. This is done in $O(|V|)$. Otherwise both s_1 and t_1 are vertices of $L(s_2,t_2)$ and both s_2 and t_2 are vertices of $L(s_1,t_1)$. In this case, the existence of a solution Q_1,Q_2 to 2DSP (x,y), (u,v) for an adjacent quadruple (x,y), (u,v), is not sufficient. We may not be able to extend it to a solution to 2DSP (s_1,t_1), (s_2,t_2) since Q_1 may use s_2 or t_2 and Q_2 may use s_1 or t_1. So in order to decide whether there exists a solution to 2DSP (s_1,t_1), (s_2,t_2) we should be able to decide for each adjacent quadruple (x,y), (u,v) whether there exists a solution Q_1, Q_2 to 2DSP (x,y), (u,v) such that $s_2,t_2 \notin Q_1$ and $s_1,t_1 \notin Q_2$. We divide the adjacent quadruples to groups and for each group we show how this is verified. We have 3^4 groups of adjacent quadruples:

$$x \in L(s_1,s_2) \quad \text{or} \quad x \in L(t_2,s_1) \quad \text{or} \quad x \in L(s_1,t_1) \setminus (L(s_1,s_2) \cup L(t_2,s_1)).$$
$$y \in L(s_2,t_1) \quad \text{or} \quad y \in L(t_1,t_2) \quad \text{or} \quad y \in L(s_1,t_1) \setminus (L(s_2,t_1) \cup L(t_1,t_2)).$$
$$u \in L(s_1,s_2) \quad \text{or} \quad u \in L(s_2,t_1) \quad \text{or} \quad u \in L(s_2,t_2) \setminus (L(s_1,s_2) \cup L(s_2,t_1)).$$
$$v \in L(t_1,t_2) \quad \text{or} \quad v \in L(t_2,s_1) \quad \text{or} \quad v \in L(s_2,t_2) \setminus (L(t_1,t_2) \cup L(t_2,s_1)).$$

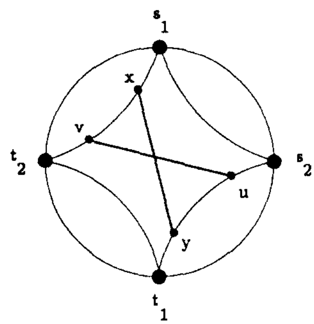

Fig. 10. Case 1.

Since some of these groups are symmetric (the quadruples are symmetric) we will actually have consider just 11 groups of adjacent quadruples for which we will show how to check the existence of a solution to 2DSP (x,y), (u,v) that can be extended to a solution to 2DSP (s_1,t_1), (s_2,t_2). The elaboration of these cases follows.

1. For adjacent quadruples $(x,y),(u,v)$ such that $x\in L(t_2,s_1)$, $y\in L(s_2,t_1)$, $u\in L(s_2,t_1), v\in L(t_2,s_1)$, (see Fig. 10), check whether there exists a solution Q_1, Q_2 to 2DSP (x,y), (u,v). By Claim 11, $s_2,t_2\notin Q_1$ and $s_1,t_1\notin Q_2$.

   Note that instead of the check above, we can check the existence of a solution to 2DSP (s_1,y), (s_2,v) for all $y\in L(s_2,t_1)$, $v\in L(t_2,s_1)$. (This check 'catches' the 9 possibilities for x to be adjacent to s_1 and u adjacent to s_2. See cases 3, 4, 9 and 11 below). If for fixed such y and v there exists a solution Q_1,Q_2 to 2DSP (s_1,y), (s_2,v), Claim 11 assures that $t_2\notin Q_1$ and $t_1\notin Q_2$. Each such solution can be extended to a solution to 2DSP (s_1,t_1), (s_2,t_2) by adding the edge (y,t_1) to Q_1 and the edge (v,t_2) to Q_2. If for all such y and v there does not exist a solution to 2DSP (s_1,y), (s_2,v), then for any adjacent quadruple (x,y), (u,v) satisfying the conditions of Case 1 there does not exist a solution to 2DSP (x,y), (u,v) that can be extended to solution to 2DSP (s_1,t_1), (s_2,t_2). The case where $x\in L(s_1,s_2)$, $y\in L(t_1,t_2)$, $u\in L(s_1,s_2)$, $v\in L(t_1,t_2)$, is symmetric.

2. For adjacent quadruples $(x,y),(u,v)$ such that $x\in L(t_2,s_1)$, $y\in L(s_2,t_1)$, $u\in (s_1,s_2), v\in L(t_1,t_2)$, check whether there exists a solution Q_1,Q_2 to 2DSP (x,y), (u,v). By Claim 11, $s_2,t_2\notin Q_1$ and $s_1,t_1\notin Q_2$.

   In this case we cannot extend a solution Q_1,Q_2 to 2DSP (s_1,y), (s_2,v), to a solution to 2DSP (s_1,t_1), (s_2,t_2) because Q_2 may use t_1.

   A solution Q_1,Q_2 to 2DSP (s_1,y), (u,t_2) cannot be extended to a solution to 2DSP (s_1,t_1), (s_2,t_2) because Q_1 may use s_2.

   A solution Q_1,Q_2 to 2DSP (x,t_1), (s_2,v) cannot be extended to a solution to 2DSP (s_1,t_1), (s_2,t_2) because Q_1 may use t_2.

   A solution Q_1,Q_2 to 2DSP (x,t_1), (u,t_2) cannot be extended to a solution to 2DSP (s_1,t_1), (s_2,t_2) because Q_2 may use s_1.

   The case where $x\in L(s_1,s_2)$, $y\in L(t_1,t_2)$, $u\in L(s_2,t_1)$, $v\in L(t_2,s_1)$, is symmetric.

2. 3.

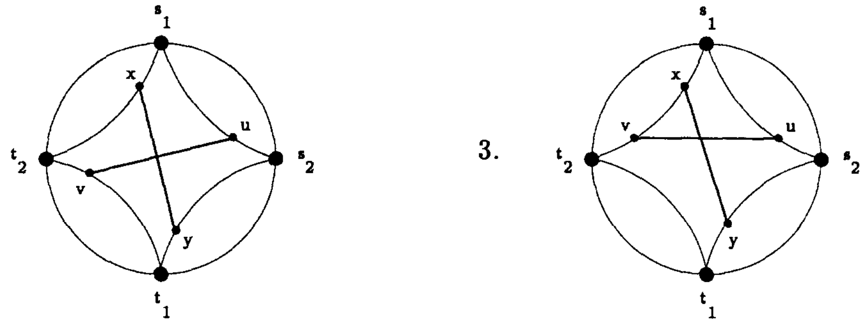

Fig. 11. Cases 2 and 3.

3. For adjacent quadruples (x,y), (u,v) such that $x \in L(t_2,s_1)$, $y \in L(s_2,t_1)$, $u \in L(s_1,s_2)$, $v \in L(t_2,s_1)$, check whether there exists a solution Q_1,Q_2 to 2DSP (s_1,y), (s_2,v). By Claim 11, $t_2 \notin Q_1$ and $t_1 \notin Q_2$.

   Note that in this case, we cannot extend a solution Q_1,Q_2 to 2DSP (x,y), (u,v) to a solution to 2DSP (s_1,t_1), (s_2,t_2) because Q_2 may use s_1. But, if there exists a solution such that Q_2 does not use s_1 then there exists a solution to 2DSP (s_1,y), (s_2,v).

   There are 7 more symmetric cases (see Fig. 11).
4. For adjacent quadruples (x,y), (u,v) such that $x \in L(s_1,s_2)$, $y \in L(s_2,t_1)$, $u \in L(s_1,s_2)$, $v \in L(t_2,s_1)$, check whether there exists a solution Q_1,Q_2 to 2DSP (s_1,y), (s_2,v). By Claim 11, $t_2 \notin Q_1$ and $t_1 \notin Q_2$.

   Note that in this case too, we cannot extend a solution to 2DSP (x,y), (u,v), to a solution to 2DSP (s_1,t_1), (s_2,t_2) because Q_2 may use s_1 and Q_1 may use s_2.

   There are 3 more symmetric cases.

   We denote $L(s_1,s_2) \cup L(s_2,t_1) \cup L(t_1,t_2) \cup L(t_2,s_1)$ by C.
5. For adjacent quadruples (x,y), (u,v) such that $x \in L(s_1,t_1) \backslash C$, $u \in L(s_2,t_2) \backslash C$, check whether there exists a solution Q_1,Q_2 to 2DSP (x,y), (u,v). By Claim 12, $s_2,t_2 \notin Q_1$ and $s_1,t_1 \notin Q_2$.

   Note that instead of the check above, we can check the existence of a solution to 2DSP (x,t_1), (u,t_2).

   If $y \in C$ and $v \in C$ there are 16 symmetric cases. If only one of y and v belongs to C there are 8 symmetric cases. If both y and v do not belong to C there exists only one case (see Fig. 12).
6. For adjacent quadruples (x,y), (u,v) such that $x \in L(s_1,s_2)$, $y \in L(s_2,t_1)$, $u,v \in L(s_2,t_2) \backslash C$, check whether there exists a solution Q_1,Q_2 to 2DSP (s_1,y), (s_2,v). By Claim 11, $t_2 \notin Q_1$. By Claim 12, $t_1 \notin Q_2$.

   Note that in this case, we cannot extend a solution to 2DSP (x,y), (u,v), to a solution to 2DSP (s_1,t_1), (s_2,t_2) because Q_1 may use s_2.

   There are 3 more symmetric cases.

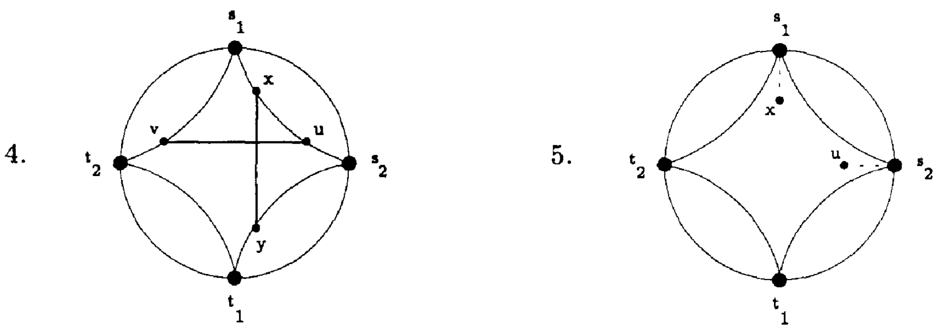

Fig. 12. Cases 4 and 5.

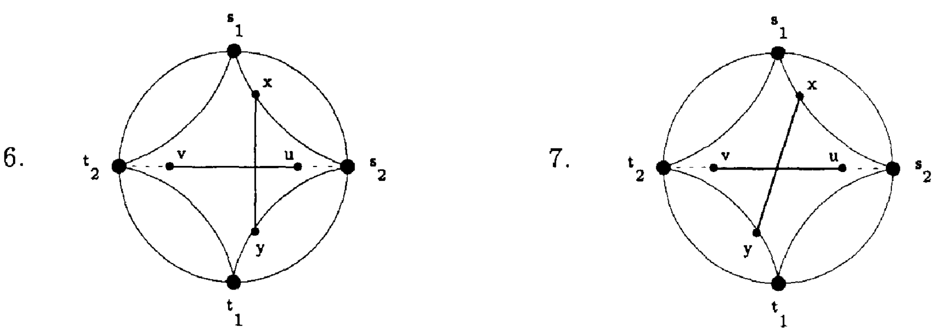

Fig. 13. Cases 6 and 7.

7. For adjacent quadruples $(x,y),(u,v)$ such that $x \in L(s_1,s_2)$, $y \in L(t_1,t_2)$, $u,v \in L(s_2,t_2)\backslash C$, check whether there exists a solution Q_1,Q_2 to 2DSP (x,y), (u,v). By Claim 12, $s_1,t_1 \notin Q_2$. By Claim 11, $s_2,t_2 \notin Q_1$.

   Note that instead of the check above, we can check the existence of a solution to 2DSP (s_1,y), (u,t_2).

   There are 3 more symmetric cases (see Fig. 13).
8. For adjacent quadruples (x,y), (u,v) such that $x \in L(s_1,t_1)\backslash C$, $y \in L(s_2,t_1)$, $u \in L(s_2,t_1)$, $v \in L(t_1,t_2)$, check whether there exists a solution Q_1,Q_2 to 2DSP (x,t_1), (u,t_2). By Claim 11, $s_1 \notin Q_2$. By Claim 12, $s_2,t_2 \notin Q_1$.

   Note that in this case, we cannot extend a solution to 2DSP (x,y), (u,v), to a solution to 2DSP (s_1,t_1), (s_2,t_2) because Q_2 may use t_1.

   There are 7 more symmetric cases.
9. For adjacent quadruples (x,y), (u,v) such that $x \in L(s_1,t_1)\backslash C$, $y \in L(s_2,t_1)$, $u \in L(s_1,s_2)$, $v \in L(t_2,s_1)$, check whether there exists a solution Q_1,Q_2 to 2DSP (s_1,y), (s_2,v). By Claim 11, $t_1 \notin Q_2$ and $t_2 \notin Q_1$.

   Note that in this case, we cannot extend a solution to 2DSP (x,y), (u,v), to a solution to 2DSP (s_1,t_1), (s_2,t_2) because Q_2 may use s_1.

   There are 7 more symmetric cases (see Fig. 14).

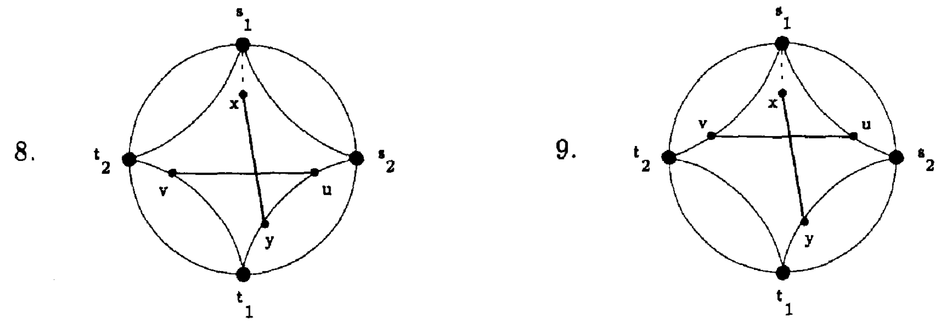

Fig. 14. Cases 8 and 9.

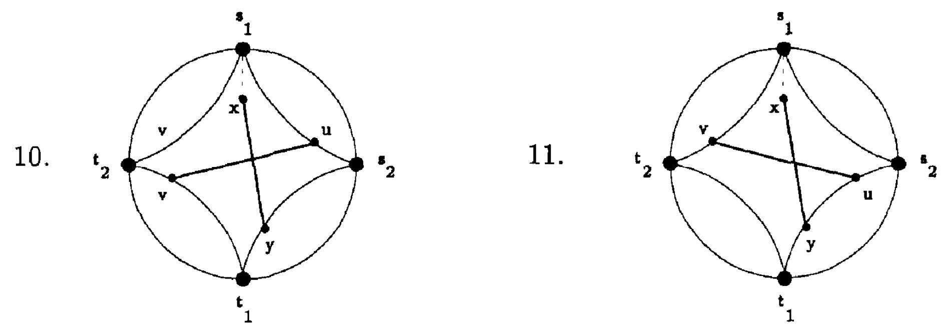

Fig. 15. Cases 10 and 11.

10. For adjacent quadruples (x,y), (u,v) such that $x \in L(s_1,t_1)\backslash C$, $y \in L(s_2,t_1)$, $u \in L(s_1,s_2)$, $v \in L(t_1,t_2)$, check whether there exists a solution Q_1,Q_2 to 2DSP (x,y), (u,v). By Claim 11, $s_1,t_1 \notin Q_2$. By Claim 12, $s_2,t_2 \notin Q_1$.

    Note that instead of the check above, we can check the existence of a solution to 2DSP (x,t_1), (s_2,v).

    There are 7 more symmetric cases.
11. For adjacent quadruples (x,y), (u,v) such that $x \in L(s_1,t_1)\backslash C$, $y \in L(s_2,t_1)$, $u \in L(s_2,t_1)$, $v \in L(t_2,s_1)$, check whether there exists a solution Q_1,Q_2 to 2DSP (x,y), (u,v). By Claim 11, $s_1,t_1 \notin Q_2$. By Claim 12, $s_2,t_2 \notin Q_1$.

    Note that instead of the check above, we can check the existence of a solution to 2DSP (x,t_1), (u,t_2) or the existence of a solution to 2DSP (s_1,y), (s_2,v).

    There are 7 more symmetric cases (see Fig. 15).

We give now the 2DSP algorithm which rises from the discussion above. This is a bottom-up algorithm which is implemented using dynamic programming.

The 2DSP algorithm

1. If $s_1 \notin L(s_2,t_2)$ check for every vertex $x \in L(s_1,t_1)$ adjacent to s_1, whether there exists a solution Q_1,Q_2 to 2DSP (x,t_1), (s_2,t_2). There exists a solution to 2DSP

(s_1,t_1), (s_2,t_2) iff there exists a solution to at least one of these problems. If $s_1 \in L(s_2,t_2)$ but $t_1 \notin L(s_2,t_2)$ or $t_2 \notin L(s_1,t_1)$ or $s_2 \notin L(s_1,t_1)$ we perform similar checks.

2. If $s_1,t_1 \in L(s_2,t_2)$ and $s_2,t_2 \in L(s_1,t_1)$ check whether there exists a solution to one of the following 2DSP problems. There exists a solution to 2DSP (s_1,t_1), (s_2,t_2) iff there exists a solution to at least one of these problems. Note that $x,y \in L(s_1,t_1)$ are adjacent to s_1 and t_1, respectively, and $u,v \in L(s_2,t_2)$ are adjacent to s_2 and t_2, respectively.

- 2DSP (s_1,y), (s_2,v) for all $y \in L(s_2,t_1)$, $v \in L(t_2,s_1)$.
- 2DSP (x,t_1), (s_2,v) for all $x \in L(s_1,s_2)$, $v \in L(t_1,t_2)$. B
- 2DSP (s_1,y), (u,t_2) for all $y \in L(t_1,t_2)$, $u \in L(s_1,s_2)$.
- 2DSP (x,t_1), (u,t_2) for all $x \in L(t_2,s_1)$, $u \in L(s_2,t_1)$.
- 2DSP (s_1,y), (s_2,v) for all $y,v \notin C$.
- 2DSP (x,t_1), (s_2,v) for all $x,v \notin C$.
- 2DSP (s_1,y), (u,t_2) for all $y,u \notin C$.
- 2DSP (x,t_1), (u,t_2) for all $x,u \notin C$.
- 2DSP (s_1,y), (s_2,v) for all $y \in L(s_2,t_1)$, $v \notin C$.
- 2DSP (s_1,y), (u,t_2) for all $y \in L(t_1,t_2)$, $u \notin C$.
- 2DSP (x,t_1), (s_2,v) for all $x \notin C$, $v \in L(t_1,t_2)$.
- 2DSP (s_1,y), (s_2,v) for all $y \notin C$, $v \in L(t_2,s_1)$.
- 2DSP (s_1,y), (u,t_2) for all $y \notin C$, $u \in L(s_1,s_2)$.
- 2DSP (x,t_1), (u,t_2) for all $x \in L(t_2,s_1)$, $u \notin C$.
- 2DSP (x,t_1), (u,t_2) for all $x \notin C$, $u \in L(s_2,t_1)$.
- 2DSP (x,t_1), (s_2,v) for all $x \in L(s_1,s_2)$, $v \notin C$.
- 2DSP (x,y), (u,v) for all the adjacent quadruples satisfying either
  $x \in L(t_2,s_1)$, $y \in L(s_2,t_1)$, $u \in L(s_1,s_2)$, $v \in L(t_1,t_2)$ or
  $x \in L(s_1,s_2)$, $y \in L(t_1,t_2)$, $u \in L(s_2,t_1)$, $v \in L(t_2,s_1)$.

These checks are done in $O(|V|^4)$ time for each quadruple $(s_1,t_1),(s_2,t_2)$ and take a total of $O(|V|^8)$ time. Note that except for the last group of checks the checks are done in $O(|V|^2)$. As for the last group of checks we could not check even in $O(|V|^3)$. That is, it is not sufficient to check the existence of a solution to the 2DSP (x,y), (u,v) when only one of the following holds $x=s_1$ or $y=t_1$ or $u=s_1$ or $v=t_1$.

4.2. The two edge-disjoint shortest paths problem

The edge-disjoint version of the 2DSP problem in an undirected graph $G=(V,E)$ is solvable in polynomial time too. We give two possible algorithms. Both of them make use of the algorithm for the vertex-disjoint 2DSP problem.

1. We build $L(G)$ the line graph of G. Each vertex of $L(G)$ corresponds to an edge in E. There is an edge (u,v) in $L(G)$ iff the edges corresponding to u and v are adjacent in G. The length of the edge (u,v) in $L(G)$ equals $[l_G(u)+l_G(v)]/2$. We

add four vertices s_1', t_1', s_2' and t_2' and edges from s_1' to all the vertices in $L(G)$ which correspond to the edges adjacent to s_1 in G, from t_1' to all the vertices which correspond to the edges adjacent to t_1 in G, etc. The length of the edge (s_1', v) in $L(G)$ equals $l_G(v)/2$. In $L(G)$ we look for two vertex-disjoint shortest paths s_1'–t_1' and s_2'–t_2'. There exist a solution to the vertex-disjoint 2DSP (s_1', t_1'), (s_2', t_2') in $L(G)$ iff there exist a solution to the edge-disjoint 2DSP (s_1, t_1), (s_2, t_2) in G.
2. Look for two vertex-disjoint shortest paths. If such paths exist clearly they are edge-disjoint. If no two such vertex-disjoint shortest paths exist but there exist two edge-disjoint shortest paths then these paths intersect in at least one vertex. For each vertex $v \in V$ check whether there exist two edge-disjoint shortest paths which intersect in v. This is done by checking for each vertex v whether there exist four edge-disjoint shortest paths from v to s_1, t_1, s_2 and t_2 as follows: add V a vertex t and four edges from s_1, t_1, s_2 and t_2 to t, assign these edges lengths $L - l(v, s_1)$, $L - l(v, t_1)$, $L - l(v, s_2)$, $L - l(v, t_2)$, respectively, where $L > \max\{l(v, s_1), l(v, t_1), l(v, s_2), l(v, t_2)\}$. In the resulting graph look for the maximum number of edge-disjoint shortest paths between v and t. This is done by orienting the edges according their orientation in the graph of shortest paths from v to t. The capacity of each edge is 1. In this acyclic network we look for the maximum integer flow. It is at most four and if it is exactly four then the answer to edge-disjoint 2DSP problem is positive. Otherwise, the answer is negative.

4.3. *A compact representation of all solutions to all 2DSP problems in G*

The 2DSP algorithm enables us to build a directed graph D which is a compact representation of all the solutions to all 2DSP problems in G. In order to simplify the description we assume that G is a graph with unit edge lengths. The vertices of D are one special vertex s and a vertex for every quadruple of distinct vertices in G. The vertex which stands for a quadruple (x, y, u, v) corresponds to 2DSP (x, y), (u, v) problem. The vertices of D are arranged in levels. The first level consists only of s. The second level consists of all the quadruples (x_1, y_1, x_2, y_2) such that $l(x_1, y_1) = l(x_2, y_2) = 1$. The nth level consists of all the quadruples (x_1, y_1, x_2, y_2) such that $l(x_1, y_1) + l(x_2, y_2) = n$, $l(x_1, y_1)$, $l(x_2, y_2) \geqslant 1$. There are no arcs between vertices in the same level. The arcs are always directed from a vertex in the lower level to a vertex in a higher level. There are arcs from s to all the vertices on the second level of D. The rest of the arcs in D are added along with the execution of the 2DSP algorithm. Whenever the algorithm deduces the existence of a solution to 2DSP (s_1, t_1), (s_2, t_2) from the existence of a solution to 2DSP (x, y), (u, v) we add to D an arc from (x, y, u, v) to (s_1, t_1, s_2, t_2). This arc represents the vertices or edges in G whose addition to the solution to 2DSP (x, y), (u, v) form a solution to 2DSP (s_1, t_1), (s_2, t_2).

Note that when $x \notin L(s_2, t_2)$ we get an arc in D which connects vertices in two consecutive levels and it represents the vertex s_1 or the edge (s_1, x) in G. When both s_1 and $t_1 \in L(s_2, t_2)$ and both s_2 and $t_2 \in L(s_1, t_1)$ if (x, y), (u, v) falls in case 2 of the 2DSP algorithm we get an arc in D which connects vertices in levels whose difference is four. If (x, y), (u, v) falls in any other case except case 2 we get an arc in D which connects vertices in levels whose difference is two.

Claim 14. *Every solution to a 2DSP (s_1, t_1), (s_2, t_2) is represented by at least one path from s to (s_1, t_1, s_2, t_2) in D and each path in D from s to (s_1, t_1, s_2, t_2) stands for a solution to 2DSP (s_1, t_1), (s_2, t_2).*

Proof. We show that each path from s to (s_1, t_1, s_2, t_2) in D corresponds to a solution to 2DSP (s_1, t_1), (s_2, t_2), by induction on n – the level in which (s_1, t_1, s_2, t_2) occurs in D. It is, of course, true for quadruples on the second level. Suppose it holds for quadruples in levels less then n and show that it holds for quadruples (s_1, t_1, s_2, t_2) in the nth level.

If there is an arc in D from (x, t_1, s_2, t_2) in level $n-1$ to (s_1, t_1, s_2, t_2) then $s_1 \notin L(s_2, t_2)$ and x is adjacent to s_1 in $L(s_1, t_1)$. By the induction hypothesis we know that every path from s to (x, t_1, s_2, t_2) stands for a solution to 2DSP (x, t_1), (s_2, t_2). We know that each such solution Q_1, Q_2 can be extended to a solution to 2DSP (s_1, t_1), (s_2, t_2) by adding the edge (s_1, x) to Q_1. So every path from s to (s_1, t_1, s_2, t_2) which passes through (x, t_1, s_2, t_2) corresponds to a solution. If there is an arc from either (s_1, y, s_2, t_2) or (s_1, t_1, u, t_2) or (s_1, t_1, s_2, v) to (s_1, t_1, s_2, t_2) it is shown similarly that every path from s to (s_1, t_1, s_2, t_2) which passes through these quadruple corresponds to a solution.

If there is no arc entering (s_1, t_1, s_2, t_2) from a vertex in level $n-1$ then both s_1 and $t_1 \in L(s_2, t_2)$ and both s_2 and $t_2 \in L(s_1, t_1)$. There may be either arcs from quadruples (x, y, u, v) in level $n-4$ to (s_1, t_1, s_2, t_2) and then (x, y), (u, v) falls in case 2 of the 2DSP algorithm or arcs from quadruples in level $n-2$ to (s_1, t_1, s_2, t_2) and then (x, y), (u, v) falls in any of the other cases except case 2. By the induction hypothesis we know that every path from s to (x, y, u, v) stands for a solution to 2DSP (x, y), (u, v). Note that in each of these cases (1 - 11) the 2DSP algorithm deduces that there exists a solution P_1, P_2 to 2DSP (s_1, t_1), (s_2, t_2) from the existence of a solution Q_1, Q_2 to 2DSP (x, y), (u, v) only when we can assure that **every** such solution Q_1, Q_2 can be extended to a solution to the 2DSP (s_1, t_1), (s_2, t_2). So every path from s to (x, y, u, v) in level $n-2$ or $n-4$ plus the arc from (x, y, u, v) to (s_1, t_1, s_2, t_2) corresponds to a solution to 2DSP (s_1, t_1), (s_2, t_2).

We show now that each solution to a 2DSP (s_1, t_1), (s_2, t_2) problem is represented by a path from s to (s_1, t_1, s_2, t_2) by induction on n - the level of (s_1, t_1, s_2, t_2) in D. It is easily seen that this holds for to all the quadruples in the second level. Suppose it holds for quadruples (s_1, t_1, s_2, t_2) in level less then n and show that it holds for quadruples in the nth level. Let $P_1 = (s_1 = x_0, \ldots, x_k = t_1)$ and $P_2 = (s_2 = u_0, \ldots, u_{n-k} = t_2)$ be a solution. By the induction hypothesis any solution P_1', P_2' to 2DSP (x_i, x_j), (u_k, u_l) is represented by a path from s to (x_i, x_j, u_k, u_l).

If $s_1 \notin L(s_2, t_2)$ the 2DSP algorithm added an arc from (x_1, t_1, s_2, t_2) to (s_1, t_1, s_2, t_2). This arc and the path representing the solution $P_1' = (x_1, \ldots, x_k = t_1)$, $P_2' = (s_2 = u_0, \ldots, u_{n-k} = t_2)$ form the desired path from s to (s_1, t_1, s_2, t_2).

If both s_1, $t_1 \in L(s_2,t_2)$ and both s_2, $t_2 \in L(s_1,t_1)$ then the 2DSP algorithm added an arc according to the case into which falls (x_1,x_{k-1}), (u_1,u_{n-k-1}). The arc was added either from $(x_1,x_{k-1},u_1,u_{n-k-1})$ or from $(s_1,x_{k-1},s_2,u_{n-k-1})$ or from (s_1,x_{k-1},u_1,t_2) or from (x_1,t_1,u_1,t_2) or from (x_1,t_1,s_2,u_{n-k-1}) to (s_1,t_1,s_2,t_2). Again, this arc plus the path representing the appropriate subsolution form a path representing P_1, P_2. □

The structure of D enables us to deal with the following problems:

- Does there exist a solution to 2DSP (s_1,t_1), (s_2,t_2) which does not use a given subset of the vertices of G?

  Delete from D all the arcs which correspond to the forbidden set. In the resulting graph, look for a path from s to (s_1,t_1,s_2,t_2).
- The vertices or edges of G are assigned weights. Find a solution to 2DSP (s_1,t_1), (s_2,t_2) of minimal weight.

  Construct the graph D as follows: Assign an arc e in D length – the sum of weights of its corresponding edges in G. Then, compute the shortest path from s to (s_1,t_1,s_2,t_2) in D.

5. Orientation problems

In this section we deal the orientation problems related to the 2D1SP problem and the 2DSP problem.

5.1. The orientation problem related to the 2D1SP problem

Hassin and Megiddo [4] considered the feasibility of orientations, i.e. given an undirected graph G and k pairs of vertices (s_i,t_i) find an orientation of G in which there exists a directed path P_i from s_i to t_i. As they mentioned there, the existence of a feasible orientation can be decided as follows. Find all the bridges of G. Choose an arbitrary path from s_i to t_i, for all $1 \leqslant i \leqslant k$. Orient the the bridges of G which belong to these paths, with accordance to their orientation on these paths. If you get a contradiction, that is, a bridge was oriented in two opposite directions, then there does not exist a feasible orientation. If we do not get a contradiction then there exists a feasible solution. In the rest of the graph each 2-connected component is oriented in a strongly connected way.

We consider the following related problem. Given an undirected graph G and 2 pairs of vertices (s_1,t_1), (s_2,t_2), find a feasible orientation of G such that, in addition to the feasibility we demand that the length of the path from s_1 to t_1 in the directed graph equals the length of a shortest path between them in G. This is the orientation problem related to the 2D1SP problem. The following algorithm solves this problem.

Choose an arbitrary path from s_1 to t_1 and orient the bridges of G which belong to this path with accordance to their orientation on this path. (Note that the bridges belong to any path between s_1 and t_1, including the shortest paths.) Repeat the same

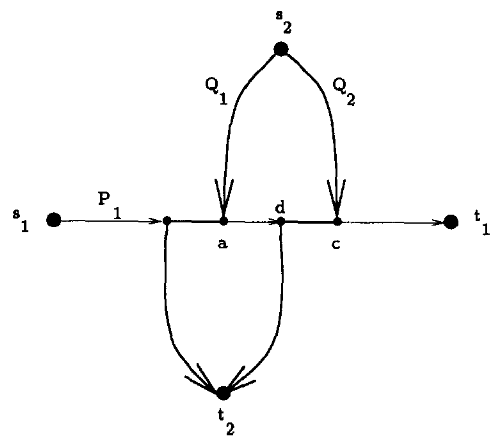

Fig. 16.

for s_2, t_2. If we get a contradiction, then the desired orientation does not exist. If we do not get a contradiction then the problem reduces to 2D1SP orientation problems in the 2-connected components of the graph. We show that in a 2-connected component there always exists such an orientation as follows. Orient the edges of an arbitrary s_1–t_1 shortest path P_1, from s_1 to t_1. Let Q_1, Q_2 be two vertex-disjoint s_2–t_2 paths. Orient the edges of $Q_1 \backslash P_1$ and $Q_2 \backslash P_1$ from s_2 to t_2.

Claim 15. *The above orientation contains a directed path P_2 from s_2 to t_2.*

Proof. Suppose that both Q_1, Q_2 are not disjoint to P_1 and and orienting them from s_2 to t_2 would contradict the orientation of P_1. Let a be the first vertex in P_1 which Q_1 meets and c the first vertex in P_1 which Q_2 meets. Let b be the last vertex in P_1 which Q_1 meets and d the last vertex in P_1 which Q_2 meets. Given two vertices $x, y \in P_1$ we denote by $x < y$ that x is closer to s_1 than y. Assume w.l.o.g. that $a < c$ and $b < d$.

- If $a < b$ or $a < d$, (see Fig. 16), then P_2 consists of the s_2–a subpath of Q_1 followed by the a–b or a–d subpath of P_1 and then it continues on the appropriate Q to t_2. Similarly, if $c < b$ or $c < d$ we get a directed s_2–t_2 path.
- Suppose now that $d < a$ (i.e. $b < d < a < c$). Let $u_1, \ldots, u_i$ be the sequence of vertices in which Q_1 alternately enters and leaves P_1 and $v_1, \ldots, v_j$ be the sequence of vertices in which Q_2 alternately enters and leaves P_1.

  Consider the directed subgraph which consists of the following subpaths:

  (i) The u_k–u_{k+1} subpaths of Q_1 where u_k leaves P_1 and u_{k+1} enters P_1, $u_k > u_{k+1}$, and either $d < u_k < a$ or $d < u_{k+1} < a$.

  (ii) The v_k–v_{k+1} subpaths of Q_2 where v_k leaves P_1 and v_{k+1} enters P_1, $v_k > v_{k+1}$, and either $d < v_k < a$ or $d < v_{k+1} < a$.

(iii) The x–y subpath of P_1 where x is the maximum between a and the leaving points of the above subpaths from P_1 and y is the minimum between d and the entering points of the above subpaths to P_1.

We show that this is a strongly connected component, that is, every edge belongs to a directed cycle. It is obvious for the edges not on P_1 and those edges $(u,v) \in P_1$ enclosed by a subpath of Q_1 or Q_2. That is, there exists a subpath from (i) or (ii) such that $u_{k+1} \leqslant u < v \leqslant u_k$ or $v_{k+1} \leqslant u < v \leqslant v_k$. Suppose there exists an edge $e \in P_1$ which is not enclosed by such a subpath of Q_1. Then, when Q_1 goes from s_2 to t_2 it has to traverse e in an opposite direction to the orientation given to e. Q_2 is vertex disjoint to Q_1 so it does not use e and there should be a subpath of Q_2 in this subgraph which encloses e. So in this strongly connected subgraph we have a directed path from a to d. This directed a–d path preceded by the s_2–a subpath of Q_1 and followed by the d–t_2 subpath of Q_2, forms a directed s_2–t_2 path. □

5.2. The two ideal orientation problem

Consider the two ideal orientation problem raised by Hassin and Megiddo [4]. Given an undirected graph G and four vertices s_1, t_1, s_2, t_2. We want to orient the edges of G so that there exist two directed paths P_1, P_2 from s_1 to t_1 and from s_2 to t_2, respectively, and the length of P_i is equal to the length of the s_i, t_i shortest path in G. The algorithm given by them is a bottom-up algorithm similar to the one we gave in Section 4.1. There too, they considered the case where both s_1 and $t_1 \in L(s_2,t_2)$ and both s_2 and $t_2 \in L(s_1,t_1)$. For this case they gave the following orientations. Orientations along a shortest paths from s_1 to s_2, from s_2 to t_1 and from t_1 to t_2. By Claim 10, these three shortest paths are disjoint except for their ends. The edges of the shortest path from s_2 to t_1 belong to both P_1 and P_2. In this case, any other ideal orientation cannot have more common edges and of course there may exist two disjoint shortest paths, that is, an ideal orientation with no common edges. If there do not exist two disjoint shortest paths we may be interested in finding those orientations of minimum common edges.

The algorithm for the 2DSP problem suggests another solution for the two ideal orientation problem which is more general in the sense that it takes into account all possible orientations. It enables us to find an orientation of minimum common edges.

Given a graph G we perform the following reduction from the ideal orientation problem to the 2DSP problem. If $e \in L(s_1,t_1), L(s_2,t_2)$ and has the same direction in $L(s_1,t_1)$ and $L(s_2,t_2)$ then it can belong to both shortest paths. We replace each such edge in G by two parallel edges. If $e \in L(s_1,t_1), L(s_2,t_2)$ and has opposite directions then it can belong to at most one of the shortest paths. In this case e is left unchanged. In the resulted undirected graph G' we look for two edge-disjoint shortest paths $P_1(s_1,t_1)$, $P_2(s_2,t_2)$. There exists an ideal orientation in G iff there exists a solution to the edge-disjoint 2DSP (s_1,t_1), (s_2,t_2) in G'. We assign the edges in G' weights as follows. When there exist two parallel edges we assign one of them weight one and the other weight zero. All the other edges get weight zero. We look for two edge-disjoint shortest paths in G' of minimal weight. A solution of minimal weight would use as few as possible pairs of parallel edges from G' and the minimal weight equals to the number of common edges.

Open problems. In this paper we investigated variations of the disjoint shortest paths problem. We proved hardness results in some cases and provided polynomial-time algorithms in other cases. However, the complexity of the undirected kDSP problem for fixed $k \geqslant 3$ and the directed kDSP problem for fixed $k \geqslant 2$ is left open. Also, the complexity status of finding two disjoint paths with minimum sum of lengths is not known.

References

[1] S. Even, A. Itai, A. Shamir, On the complexity of timetable and multi-commodity flow problems, SIAM J. Comput. 5 (1976) 691–703.
[2] S. Fortune, J. Hopcroft, J. Wyllie, The directed subgraph homeomorphism problem, Theoret. Comput. Sci. 10 (1980) 111–121.
[3] J. Gustedt, The general two-path problem in time $O(m \log n)$, Technical Report 394, TU Berlin.
[4] R. Hassin, N. Megiddo, On orientations and shortest paths, Linear Algebra Appl. 114/115 (1989) 589–602.
[5] A. Itai, Y. Perl, Y. Shiloach, The complexity of finding maximum disjoint paths with length constraints, Networks 12 (1982) 277–286.
[6] R.M. Karp, On the complexity of combinatorial problems, Networks 5 (1975) 45–68.
[7] E. Korach, Packings of T-cuts and other aspects of dual integrality, Ph.D. thesis, Waterloo University 1982.
[8] P. Klien, S. Rao, M. Rauch, S. Subraminian, Faster shortest-path algorithms for planar graphs, Proceedings of STOC 94.
[9] C.L. Li, S.T. McCormick, D. Simchi-Levi, The complexity of finding two disjoint paths with min–max objective functions, Discrete Appl. Math. 26 (1990) 105–115.
[10] C.L. Li, S.T. McCormick, D. Simchi-Levi, Finding disjoint paths with different path-costs: complexity and algorithms, Networks 22 (1992) 653–667.
[11] D. Lichtenstein, Planar formulae and their uses, SIAM J. Comput. 11 (1982) 329–343.
[12] J.F. Lynch, The equivalence of theorem proving and the interconnection problem, ACM SIGDA Newslett. 5 (1975) 31–38.
[13] M. Middendorf, F. Pfeiffer, On the complexity of the disjoint paths problem, Combinatorica 13 (1993) 97–107.
[14] T. Ohtsuki, The two disjoint path problem and wire routing design, Graph Theory and Algorithms, Sendai, Japan, October 1980, Proceedings.
[15] Y. Perl, Y. Shiloach, Finding two disjoint paths between two pairs of vertices in a graph, J. ACM 25 (1978) 1–9.
[16] N. Robertson, P.D. Seymour, Disjoint paths-a survey, SIAM J. Algebraic Discrete Methods 6 (1985) 300–305.
[17] N. Robertson, P.D. Seymour, Graph minors VI. Disjoint paths across a disc, J. Combin. Theory Ser. B 41 (1986) 115–138.
[18] N. Robertson, P.D. Seymour, Graph minors VII. Disjoint paths on a surface, J. Combin. Theory Ser. B 45 (1988) 212–254.
[19] N. Robertson, P.D. Seymour, Graph minors IX. Disjoint crossed paths, J. Combin. Theory Ser. B 49 (1990) 40–47.
[20] A. Schrijver, Finding k disjoint paths in a directed planar graph, SIAM J. Comput. 23 (1994) 780–788.

[21] A. Sebő, Integer plane multiflows with a fixed number of demands, J. Combin. Theory Ser. B 59 (1993) 163–171.

[22] P.D. Seymour, Disjoint paths in graphs, Discrete Math. Comput. 29 (1980) 293–309.

[23] P.D. Seymour, On odd cuts and plane multi-commodity flows, Proc. London Math. Soc. 42 (1981) 178–192.

[24] Y. Shiloach, A polynomial solution to the undirected two paths problem, J. ACM 27 (1980) 445–456.