Vertex Disjoint Paths for Dispatching in Railways*

Holger Flier[1], Matúš Mihalák[1], Anita Schöbel[2], Peter Widmayer[1], and Anna Zych[1]

1 ETH Zürich, Institute of Theoretical Computer Science, Switzerland
{firstname.lastname}@inf.ethz.ch
2 Georg-August Universität Göttingen, Germany
Institut für Numerische und Angewandte Mathematik
schoebel@math.uni-goettingen.de

Abstract

We study variants of the vertex disjoint paths problem in planar graphs where paths have to be selected from a given set of paths. We study the problem as a decision, maximization, and routing-in-rounds problem. Although all considered variants are NP-hard in planar graphs, restrictions on the location of the terminals, motivated by railway applications, lead to polynomially solvable cases for the decision and maximization versions of the problem, and to a p-approximation algorithm for the routing-in-rounds problem, where p is the maximum number of alternative paths for a terminal pair.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems—Routing and layout, G.2.2 Graph Theory—Path and circuit problems, G.2.3 Applications

Keywords and phrases algorithms, approximation, complexity, graph theory, railways, routing, transportation

Digital Object Identifier 10.4230/OASIcs.ATMOS.2010.61

1 Introduction

We study variants of the vertex disjoint paths problem in planar graphs where for each terminal pair a set of alternative paths is given. Our motivation to study these problems arises from railway applications. During operations, railway dispatchers face the challenging problem of rerouting and rescheduling trains in the presence of delays. Once a train is delayed, it might be in conflict with other trains that are planned to use the same track resources. The dispatcher then has to find a new feasible plan in a very short amount of time. Interestingly enough, these complicated decisions are carried out mostly by humans today, with only basic computer support such as graphical monitoring tools. Nevertheless, the dispatching decisions have a considerable impact on reliability and punctuality as experienced by passengers. Motivated by the importance of the problem and by the lack of methods that can cope with both practical problem sizes and the real-time setting, we study special vertex disjoint paths problems which are abstractions of the dispatching problem.

Typically, a railway station is modeled as a graph with nodes representing points on the tracks, and edges representing track segments that connect such points. We study the case where the resulting graphs are planar, which is the case for many junctions and stations. Considering only the aspect of routing, two trains are in conflict if their routes share a point

* This work was partially funded by the Swiss National Science Foundation (SNF grant no. 200021-125033/1).

on the tracks. Hence, conflict free routes correspond to vertex disjoint paths. Not every route which is physically feasible is desirable in practice, though. Therefore, railway planners allow for each train only a small set of alternative paths for each train. This leads us to various vertex disjoint paths problems where for each terminal pair, corresponding to a train with a given start and target in the railway network, a path has to be chosen from a given set of paths, i.e., the possible set of routes for the train.

1.1 Related Work

We give a brief overview both over literature related to dispatching and results on disjoint paths problems. For a recent survey on railway track allocation problems, see [17]. As noted therein, most of the approaches known so far are impractical in a real time environment.

One line of research aiming at real-time solutions is based on the alternative graph formulation [19], originally used to model job shop variants. The formulation allows to model many constraints, e.g. scheduled stops, rolling stock connections and passenger connections, but does not allow for alternative routes or train speed adaptation [3]. A branch-and-bound algorithm for finding a conflict-free train schedule, minimizing the largest delay, is developed in [4]. In order to solve the compound problem of train sequencing and train routing, where a set of possible routes is given as input, a tabu search is suggested in [2]. For given routes and fixed train speeds, the branch and bound algorithm of [4] is used as a sub-procedure to solve the train sequencing problem. A procedure for handling train speed dynamics that respect signal aspects is presented in [3]. Assuming fixed routes, the coordination of train speeds is performed by iteratively solving the scheduling problem with fixed speeds and updating the train speed profiles if these are not physically realizable.

A complexity study on routing trains through railway stations is given in [15]. There, trains are given a fixed set of inbound and outbound routes to choose from. For each route, all track sections are reserved at once but released section-wise. The problem of deciding whether a feasible schedule exists in which all trains can be scheduled is NP-complete already for 3 possible routes per train, but reduces to 2-SAT for at most 2 possible routes per train. For a fixed number of track sections, a fixed parameter algorithm is provided.

Finally, we give a few pointers to literature on vertex disjoint paths problems where the paths can be chosen arbitrarily. The problem of finding k vertex disjoint paths between k pairs of terminals is NP-complete already for k = 2 in directed *non-planar* graphs [8], and if k is not fixed, even in planar graphs [18]. The vertex disjoint paths problem is solvable in polynomial time in undirected graphs for any fixed k [21], and in directed *planar* graphs for any fixed k [22]. Shortest disjoint paths are treated in [13]. Practically efficient algorithms for special cases of the disjoint paths problem are surveyed in [20].

1.2 Problem Definition

Throughout the paper we study a variety of optimization problems. They share a common input, but differ in objectives and additional assumptions on the input. In what follows, first we define the input, and then we categorize the studied problems.

An input instance for the problems we study is a triple $(G, T, \mathcal{P})$, defined as follows: $G = (V, E)$ is an undirected plane graph, i.e., a planar embedding of a planar graph G. $T \subseteq V$ is a set of k *terminal pairs* $\{s_i, t_i\}$, $i = 1, 2, \dots, k$. Vertices in T are called *terminals.* A path from s_i to t_i is called an s_i-t_i-path. $\mathcal{P} = \{\mathcal{P}_i\}_{i=1\dots k}$ is a collection of sets of paths, where $\mathcal{P}_i$ is a set of s_i-t_i-paths for every $i = 1, \dots, k$. We denote by p the maximum cardinality of a set in $\mathcal{P}$, so $p := \max_{1 \leq i \leq k} |\mathcal{P}_i|$. We denote by $\bigcup \mathcal{P}$ the union of all sets of the collection, thus $\bigcup \mathcal{P} := \bigcup_{i=1\ldots k} \mathcal{P}_i$ is the set of all given paths. The plane embedding of G separates the plane into distinct regions, called *faces*, bordered by graph edges. The unbounded region outside the graph's embedding is called the *outer face*. We study the following algorithmic problems:

Decision Problem: Decide whether there are k vertex disjoint paths $P_1, P_2, \ldots, P_k$, where for each $i = 1, 2, \ldots, k$ path P_i is from $\mathcal{P}_i$.

Maximization Problem: Find a maximum number of vertex disjoint paths $P_{i_1}, P_{i_2}, \ldots, P_{i_m}$ where every P_{i_j}, $j = 1, \ldots, m$, is from $\mathcal{P}_{i_j}$

Routing-in-Rounds Problem: Find a labeled set of paths $S = \{P_i\}_{i=1\ldots k}$, $P_i \in \mathcal{P}_i$, where each path P_i is assigned a label (often called round) $r_i \in \mathbb{N}$, such that for any $P_i, P_j \in S$ if $P_i \cap P_j \neq \emptyset$ then $r_i \neq r_j$. The objective is to minimize the number of different labels (rounds) that were assigned.

Clearly, the decision problem can be seen both as the maximization problem, where we are to decide whether m equals k, and as the routing-in-rounds problem, where we are to decide whether one round suffices.

We study the problem for the following special cases of the positions of the terminals in input graph G. Besides the general case where the terminals can be any nodes of G, we also study the case where the terminals lie on the outer face of G. For the latter case, we also consider two special sub-cases.

First, we consider the case where the terminals appear in a counterclockwise traversal on the boundary as a sequence $s_1, s_2, \ldots, s_k, t_{\pi(1)}, t_{\pi(2)}, \ldots, t_{\pi(k)}$ for some permutation π. We say that such an instance has a *separating cut*, or that the *terminals can be separated*. See Figure 3 for an example.

Second, we consider a special case of a separating cut, where the terminals appear on the boundary of the outer face in the order $s_1, s_2, \ldots, s_k, t_k, t_{k-1}, \ldots, t_2, t_1$, in which case we say that the terminals are *sorted*.

Depending on the considered optimization goal and assumptions made about the terminals, we obtain a particular computational problem which we refer to as GOAL-VDP-TERMINALS using the following naming convention: GOAL is D, M or R if the problem is a decision problem (D), maximization problem (M), or routing-in-rounds problem (R), respectively; VDP stands for vertex disjoint paths (and appears in every name); TERMINALS is either ANY, OUT, SEP, or SORT, if we assume nothing about the positions of the terminals (ANY), the terminals appear on the outer face (OUT), the terminals can be separated (SEP), or the terminals are sorted (SOR) respectively. Thus, for example, M-VDP-OUT is a computational problem which asks, for a given plane graph G with terminals on the outer face of G, to find a maximum number of vertex disjoint paths.

1.3 Overview of the paper

This paper is structured as follows: We discuss variants of the decision problem in Section 2, of the maximization problem in Section 3, and of the routing-in-rounds problem in Section 4. An overview of our most important complexity results is given in Table 1.

2 D-VDP: Decision Problems

In this section we consider the problem of deciding whether all trains can be dispatched in the same round. An input instance is a triple $(G, T, \mathcal{P})$, where G is a plane graph, T is a set of k terminal pairs $\{s_i, t_i\}$, $i = 1, 2, \cdots, k$, and $\mathcal{P} = \{\mathcal{P}_i\}_{i=1\ldots k}$, where $\mathcal{P}_i$ is a set of

	D-VDP	M-VDP	R-VDP
Any	NP-complete for $p \geq 3$	NP-hard for $p \geq 1$	APX-hard
Out	open, trivial for $p = 1$	open	
Sep	polynomial		p-approximable, APX-complete for $p \geq 2$, polynomial for $p = 1$
Sort			

Table 1 Summary of complexity results

s_i-t_i-paths. The problem is to decide whether there are k vertex disjoint s_i-t_i-paths $P_i \in \mathcal{P}_i$, $i = 1, 2, \cdots, k$.

We show that for planar graphs the general problem D-VDP-Any is NP-complete whenever $p \geq 3$, and solvable in polynomial time otherwise. The special case D-VDP-Sep, where the terminals can be separated, can be solved in polynomial time by reduction to M-VDP-Sep, for which we give a polynomial time algorithm in Section 3.3.

The complexity of D-VDP-Out remains open for $p \geq 3$. We remark that a necessary condition for the existence of k vertex disjoint paths is that they may not cross each other. Therefore, to study the complexity of D-VDP-Out, it suffices to consider instances as follows. We say that the terminals are *nested*, if for no two terminal pairs s_i, t_i and s_j, t_j, $i \neq j$, the terminals occur in the sequence s_i, s_j, t_i, t_j when traversing the boundary of the graph in counterclockwise order. Note that if terminals occur in the sequence s_i, s_j, t_i, t_j, any two paths $P_i \in \mathcal{P}_i$ and $P_j \in \mathcal{P}_j$ intersect.

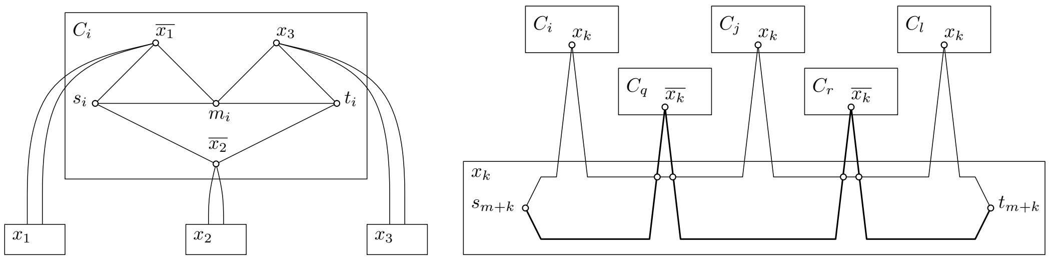
▶ Remark. If there exists a solution for an instance of D-VDP-Out, then the terminals must be nested.

Next, we prove NP-completeness of D-VDP-Any for $p \geq 3$ by reduction from Planar3SAT, which is defined as follows. Let $\phi = (X, C)$ be an instance of 3SAT, with variable set $X = \{x_1, \ldots x_n\}$ and clauses $C = \{C_1 \ldots C_m\}$ such that each clause consists of exactly 3 literals. Define a formula graph $G_\phi = (V, E)$ with vertex set $V = X \cup C$, and edges $E = \{(x_k, C_i) : x_k \in C_i \text{ or } \overline{x_k} \in C_i\}$. Planar3SAT is 3SAT restricted to instances ϕ for which G_ϕ is planar, and was proved NP-complete in [16].

▶ **Theorem 1.** D-VDP-Any *is NP-Complete for* $p \geq 3$.

Proof. Let ϕ be an instance of Planar3SAT. To construct an instance of a graph $G_p = (V_p, E_p)$ for D-VDP-Any, we start with $G_\phi = (V, E)$. We substitute each node $C_i \in V$ by a corresponding clause gadget, and each node x_i by a corresponding variable gadget, as described in the following.

A clause gadget as shown in Figure a is created for each clause $C_i \in \phi$. It consists of 6 nodes. Let $C_i = \{l_1^i, l_2^i, l_3^i\}$, where l_j^i are the literals of C_i. Three nodes of the gadget correspond to these literals. They are connected to a path (s_i, m_i, t_i) in a way that depends on a plane drawing of G_ϕ. Let e_1, e_2, e_3 be the edges in a counterclockwise order connecting vertex $C_i \in V$ in G_ϕ with vertices $x_1, x_2, x_3 \in V$, where $l_1^i \in \{x_1, \overline{x_1}\}, l_2^i \in \{x_2, \overline{x_2}\}, l_3^i \in \{x_3, \overline{x_3}\}$. We add (l_1^i, s_i), (l_1^i, m_i), (l_2^i, m_i), (l_2^i, t_i), (l_3^i, s_i), (l_3^i, t_i) to E_p, as shown in Figure a. This gadget is planar. Moreover, if we substitute node $C_i \in G_\phi$ with its clause gadget, literal nodes of the gadget can be connected with corresponding variable nodes preserving the planarity of G_ϕ. We set $\{s_i, t_i\}$ as a terminal pair in G_p. We let $\mathcal{P}_i$ be the following set of fixed paths: $\{(s_i, l_1^i, m_i, t_i), (s_i, m_i, l_2^i, t_i), (s_i, l_3^i, t_i)\}$.



(a) Clause gadget with terminal pair. **(b)** Variable gadget with terminal pair.

Figure 1 Transformation from PLANAR3SAT to D-VDP-ANY.

Now we construct a gadget for each vertex $x_k \in G_\phi$. It consists of two terminal vertices $\{s_{m+k}, t_{m+k}\}$ and two fixed paths between them: $P_{m+k}, \overline{P_{m+k}} \in \mathcal{P}_{m+k}$. Path P_{m+k} contains all the literals $\overline{x_k}$ in the clause gadgets. We want to enforce, that if the solution contains path P_{m+k}, then no other path containing literal $\overline{x_k}$ can be chosen. Intuitively, choosing P_{m+k} corresponds to setting x_k to true, and choosing a path with x_k on it for a terminal pair of a clause gadget corresponds to satisfying the clause with literal x_k. Similarly, path $\overline{P_{m+k}}$ contains all the literals x_k in the clause gadgets. In order to draw path P_{m+k}, we substitute the edges that connect x_k with clause gadgets containing $\overline{x_k}$ by peaks on the path from s_{m+k} to t_{m+k}. Thus, each peak reaches the corresponding clause gadget. We proceed analogically to draw $\overline{P_{m+k}}$. Obviously, P_{m+k} can intersect $\overline{P_{m+k}}$, but in that case we add a vertex at the spot of intersection to make G_p planar. The variable gadget is shown in Figure b.

We are asking for a choice of paths that would select one of the paths for each terminal pair such that all selected paths are vertex disjoint. It remains to show that the initial formula has a satisfying assignment if and only if such a choice exists.

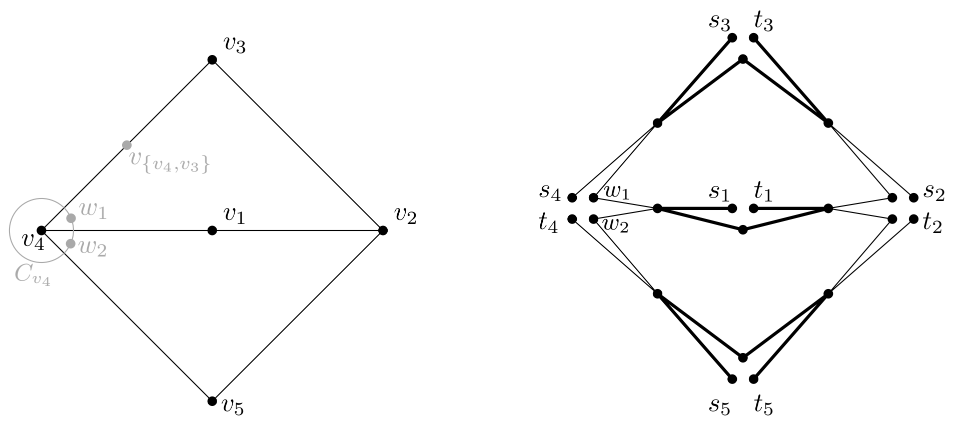
Assume that $m + n$ disjoint paths, one for each terminal pair, can be chosen. To obtain a satisfying assignment for ϕ, set x_k to true if and only if P_{m+k} was chosen for terminal pair $\{s_{m+k}, t_{m+k}\}$. To see that each clause C_i is satisfied by that assignment, let $P \in \mathcal{P}_i$ be the path chosen for a terminal pair of the corresponding clause gadget, and let l^i_j be a literal of C_i lying on P. Assume w.l.o.g. that l^i_j is a non-negated variable x_j. In that case $\overline{P_{m+j}}$ could not have been chosen, and therefore x_j must have been set to true. Thus, clause C_i is satisfied by x_j.

Now assume there is a satisfying assignment for ϕ. For each x_j, choose path P_{m+j} if x_j is set to true, and $\overline{P_{m+j}}$ otherwise. For each clause C_i, choose a path containing a literal that is set to true. ◀

In the following, we prove that we can solve instances having at most two paths per train in polynomial time by reduction to 2-SAT, which is solvable in polynomial time, see e.g. [9].

▶ **Lemma 2.** D-VDP-ANY *can be solved in polynomial time if* $p \leq 2$.

Proof. For an instance I of D-VDP-ANY we create a 2-SAT formula $\phi(I)$ which admits a satisfying assignment if and only if I has a solution. For each set $\mathcal{P}_i = \{P^1_i, P^2_i\} \in \mathcal{P}$ we create variables x^1_i, x^2_i, and add a clause $\{x^1_i, x^2_i\}$ to $\phi(I)$. In order to satisfy these clauses, one of the paths for each terminal pair has to be chosen, i.e., the corresponding variable has to be set to true. Whenever two paths P^k_j and P^l_i intersect, we add a clause $\{\overline{x^k_j}, \overline{x^l_i}\}$. These clauses forbid to choose two intersecting paths, i.e., rule out any assignment in which both corresponding variables are set to true. ◀



(a) Planar independent set instance. **(b)** Transformation to M-VDP-Any

Figure 2 Transformation from PlanarIndependentSet to M-VDP-Any. Every vertex v_i is transformed into a terminal pair $\{s_i, t_i\}$, and every edge e into an additional vertex v_e. For each terminal pair $\{s_i, t_i\}$, create an s_i-t_i-path (using auxiliary vertices and edges) that traverses (besides the auxiliary vertices) exactly every vertex v_e corresponding to an edge e adjacent to v_i. Part of the transformation is depicted in gray. (a) The maximum independent set is $\{v_1, v_3, v_5\}$. (b) The corresponding vertex disjoint paths are shown in bold.

3 M-VDP: Maximization Problems

In this section we consider variants of the maximization problem. An input instance is a triple $(G, T, \mathcal{P})$, where G is a plane graph, T is a set of k terminal pairs $\{s_i, t_i\}$, $i = 1, 2, \cdots, k$, and $\mathcal{P} = \{\mathcal{P}_i\}_{i=1\ldots k}$, where $\mathcal{P}_i$ is a set of s_i-t_i-paths. An output solution is a set $S \subseteq \bigcup \mathcal{P}$ of maximum cardinality, such that the paths of S are vertex disjoint (thus $S \cap \mathcal{P}_i \leq 1$ for all $i = 1, \ldots, k$).

We first show that M-VDP-Any is NP-complete. We leave the complexity of M-VDP-Out open but show polynomial time solvability for the special case where $p = 1$ and paths in $\bigcup \mathcal{P}$ have a certain monotonicity property. Finally, we consider M-VDP-Sep and show that it can be solved in polynomial time.

3.1 M-VDP-Any: Terminals anywhere

We show that M-VDP-Any is NP-hard already for the case $p = 1$ (i.e., when there is one fixed path per terminal pair) by a reduction from the NP-complete problem PlanarIndependentSet which is the problem of deciding whether a given planar graph contains, for a given ℓ, an independent set of size ℓ [9, GT20, p.194].

▶ **Theorem 3.** M-VDP-Any *is NP-hard already for* $p = 1$.

Proof. The reduction is illustrated in Fig. 2. Consider an instance of PlanarIndependentSet given by a planar graph G and an integer ℓ. For every node v of G we construct a terminal pair $\{s_v, t_v\}$. We further create for every edge $e = \{u, v\}$ in G a vertex v_e. We now construct an $s_v - t_v$-path P_v for every vertex v in G such that two nodes u and v from G are adjacent if and only if the paths P_u and P_v intersect. Our construction will result into a planar graph which shows that the decision variant of M-VDP-any is an NP-complete problem (and thus M-VDP-any is NP-hard). To explain how the paths P_v, $v \in V$, look like, consider a planar embedding of G. Thus, vertices of G are points of the plane, and edges of G are lines connecting the corresponding points. Place s_v and t_v close to each other on the position of v. Place vertex v_e into the middle of the line corresponding to edge e. Let d denote the degree of vertex v in G. Consider the neighbors of v in a cyclic order induced naturally by the cyclic order of the lines connecting v with its neighbors (the lines correspond to the edges in G). If vertex v is adjacent to vertices $v_1, \cdots, v_d$ (in that order) we construct a path from s_v to t_v that goes via vertices $v_{e_1}, v_{e_2}, \cdots, v_{e_d}$, where $e_i = \{v, v_i\}$, $i = 1, \cdots, d$. For vertex v we introduce auxiliary vertices $w_1, \cdots, w_{d-1}$ and let the path P_v be $s_v, v_{e_1}, w_1, v_{e_2}, w_2, \cdots, w_{d-1}, v_{e_d}, t_v$, and we also create the necessary edges for the path P_v. We note that there are no other edges in the construction than that from the paths P_v, $v \in V(G)$. It is easy to see that the resulting graph is planar (if G does not contain a minor of K_5 or $K_{3,3}$, then neither our modified instance does). Figure 2 suggests a planar embedding of our construction that resembles in shape the embedding of G. Consider a small-enough circle C_v centered in v that intersects only the lines corresponding to edges adjacent to v, and every such line exactly once. Place s_v and t_v inside C_v. We place the vertices w_i on the circle C_v (i.e., close enough to v) and between the intersections of C_v with the two lines connecting v with v_i and v_{i+1}, and draw the two lines connecting w_i, $v_{e_{i+1}}$ and w_{i+1} very closely to the original line between v and $v_{e_{i+1}}$. It is also easy to see that two vertices u, v from G are adjacent if and only if the two paths P_u and P_v intersect (at vertex v_e, $e = \{u, v\}$). ◀

We note that PlanarIndependentSet admits a PTAS [1] and thus our reduction, although approximation-preserving, does not show any hardness of approximation. This remains an interesting open problem. We also note that in contrary to the maximization problem, D-VDP-Any with $p = 1$ is trivial to solve.

3.2 M-VDP-Out: Terminals on the outer face

We do not know the complexity of M-VDP-Out in general, but point to a similar open problem in graph theory, namely the complexity of finding a maximum independent set in outerstring graphs, e.g., see [14]. It is easy to see that the class of outerstring graphs and the class of graphs considered in M-VDP-Out with $p = 1$ are equivalent: strings can be represented by paths and vice versa.

There is, however, a polynomially solvable special case of M-VDP-Out. Consider the special case of M-VDP-Out with $p = 1$ where any two paths intersect in at most one vertex, and if they intersect, they cross each other. We call such paths *monotone*.

▶ Remark. M-VDP-Out with monotone paths and $p = 1$ can be solved in polynomial time.

Proof. By reduction to maximum independent set in circle graphs, for which a polynomial time algorithm is given in [10]. A circle graph is the intersection graph of a family of chords in a circle. Considering that the paths of the instance of M-VDP-Out are monotone and have their ends on the outer face of the graph, it is easy to see that there is a family of chords in a circle where two chords cross iff their corresponding paths cross. Further, because $p = 1$, a maximum independent set in the corresponding circle graph corresponds to an optimal solution of the considered instance of M-VDP-Out. ◀

3.3 M-VDP-Sep: Separating cut

In this section we consider instances with a separating cut, i.e., where the terminals appear in a counterclockwise traversal of the outer face in the sequence $s_1, s_2, \ldots, s_k, t_{\pi(1)}, t_{\pi(2)}, \ldots, t_{\pi(k)}$ for some permutation π of the numbers $1, 2, \ldots, k$. See Figure 3 for an example.

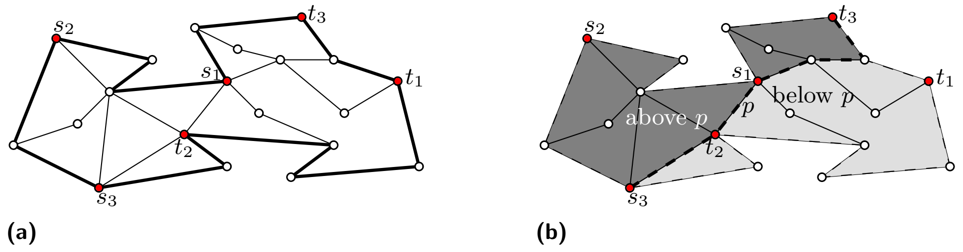


Figure 3 An instance of M-VDP-SEP, i.e., all terminals lie on the outer face of G and there is a separating cut in G. (a) The outer face is depicted in bold. (b) An $s_3 - t_3$ path p (dashed bold line), the part above p (dark shaded area), and the part below p (light shaded area).

The setting has the following important property. Every path $P \in \mathcal{P}_i$ separates the plane embedding of the graph into two parts. The part *above* P is the set in the plane enclosed by the curve formed by the boundary of the outer face between t_i and s_i (in counterclockwise order) and by path P (from s_i to t_i). The part *below* P is the set in the plane enclosed by the curve formed by the boundary of the outer face between s_i and t_i (in counterclockwise order) and by path P (from t_i to s_i). In the following we say that a point/path/vertex/etc. *lies above* P if it lies in the part above P. We similarly define to *lie below* P. Observe that both sets are compact and closed. They share only path P and otherwise are disjoint. Therefore any path P' that lies above P and is disjoint to P is also disjoint to any path P'' that lies below P. Observe also that if for some j the terminal s_j lies above P and terminal t_j lies below P then there is no $s_j - t_j$ path in G disjoint to P.

In the following, we show that the problem for such instances can be solved in polynomial time. Precisely, we present an algorithm based on dynamic programming, that computes an optimum solution and runs in time in $O(k^2 \cdot p^2)$. Thus, if p is polynomial in k, our algorithm is also polynomial in k. In any case our algorithm is polynomial in the input size (as the input for the problem lists explicitly all s_i-t_i-paths).

The algorithm computes a table $T[i, P]$ for every $i = 1, \ldots, k$ and every $P \in \mathcal{P}_i$. The entry $T[i, P]$ is the size of an optimum solution of the subproblem in which path $P \in P_i$ is chosen, and all other paths can only be chosen from sets $\mathcal{P}_1, \ldots, \mathcal{P}_{i-1}$. Initially, $T[1, P] = 1$ for all $P \in \mathcal{P}_1$. We now show how to inductively compute the whole table. Assume that the table has been filled for all values of i smaller than j. We now show how to compute the table entry $T[j, P]$ for any $P \in \mathcal{P}_j$. We set

$$T[j, P] = 1 + \max_{\substack{1 \le l < j \\ P' \in \mathcal{P}_l; P \cap P' = \emptyset}} T[l, P'].$$

The actual solutions (sets of paths) can be found using standard bookkeeping techniques. The algorithm outputs

$$\max_{\substack{1 \le i \le k \\ P \in \mathcal{P}_i}} T[i, P]$$

as the maximum number of disjoint paths.

▶ **Theorem 4.** M-VDP-SEP *is solvable in time in $O(k^2 \cdot p^2)$, where k is the number of terminal pairs and p is maximum number of paths per terminal pair.*

Proof. The number of entries in T that the algorithm has to fill is $\sum_{i=1}^{k} |\mathcal{P}_i|$, which is at most $k \cdot p$, where $p := \max_{1 \le i \le k} |\mathcal{P}_i|$. Computing an entry $T[i, P]$ requires time linear in the number of existing (i.e., so far computed) entries, i.e., at most $O(k \cdot p)$. Thus, the total running time of the algorithm is $O(k^2 \cdot p^2)$.

To finish the proof of the theorem, we are left to prove that: $(*)$ the value stored in $T[i, P]$ indeed represents the size of an optimum solution of the subproblem in which path $P \in \mathcal{P}_i$ is chosen, and all other paths can only be chosen from sets $\mathcal{P}_1, \dots, \mathcal{P}_{i-1}$. Once we have this, it is then easy to see that the algorithm returns the maximum number of disjoint paths.

To prove $(*)$ we proceed inductively on i. Trivially, the claim holds for all entries $T[1, P]$, $P \in \mathcal{P}_1$. Assume now that the claim holds for $i < j$. Let $P \in \mathcal{P}_j$. We show that the claim holds for $T[j, P]$. Let OPT be an optimum solution for the subproblem in which path $P \in \mathcal{P}_j$ is chosen, and all other paths can only be chosen from sets $\mathcal{P}_1, \dots, \mathcal{P}_{j-1}$. Consider the set $OPT' := OPT \setminus \{P\}$. Let a be the largest index such that there is a path P_a from $\mathcal{P}_a$ in OPT'. Thus, in OPT' there are only paths from $\mathcal{P}_1, \dots, \mathcal{P}_a$. By induction hypothesis, $T[a, P_a] \geq |OPT'|$. Clearly, P and P_a are disjoint. Let us denote the set of paths corresponding to $T[a, P]$ by $\mathcal{P}(T[a, P])$. In order to see that P is also disjoint to every path in $\mathcal{P}(T[a, P_a])$, consider the fact that every path $P' \in \mathcal{P}(T[a, P_a])$ different from P_a lies above P_a, and the path P lies below P_a. Thus the paths $\mathcal{P}(T[a, P_a])$ plus the path P are disjoint and we have $|\{P\} \cup \mathcal{P}(T[a, P_a])| = 1 + T[a, P_a] \geq 1 + |OPT'| = |OPT|$, which shows the claim. ◀

4 R-VDP : Routing in rounds

In this section we consider variants of the routing-in-rounds problem. An input instance is a triple $(G, T, \mathcal{P})$, where G is a plane graph, T is a set of k terminal pairs $\{s_i, t_i\}$, $i = 1, 2, \cdots, k$, and $\mathcal{P} = \{\mathcal{P}_i\}_{i=1\dots k}$, where $\mathcal{P}_i$ is a set of s_i-t_i-paths. The solution S is a labeled set of paths $S = \{P_i\}_{i=1\dots k}$, $P_i \in \mathcal{P}_i$. Each path $P_i \in S$ is assigned a label $r_i \in \mathbb{N}$, such that for any $P_i, P_j \in S$ if $P_i \cap P_j \neq \emptyset$ then $r_i \neq r_j$. Intuitively, the labels correspond to rounds, which represent a rudimentary notion of time. If paths P_i and P_j are disjoint, the corresponding trains can run at the same time (in the same round). Otherwise, to avoid collisions, we need to schedule them in different rounds. We show that already R-VDP-Sor (terminals sorted on the outer face) is APX-complete for any $p \geq 2$. Further we show that R-VDP-Sep (there is a separating cut) with $p = 1$ can be solved efficiently, and present a p-approximation algorithm for the case of $p \geq 2$.

4.1 R-VDP-Sor: Terminals sorted on the outer face

▶ **Theorem 5.** R-VDP-Sor *for any* $p \geq 2$ *is APX-complete.*

Proof. By reduction from SetCover, which defined as follows. Given a collection $\mathcal{C}$ of subsets of a ground set U, the SetCover problem asks for a collection $\mathcal{C}' \subseteq \mathcal{C}$, such that each $u_i \in U$ belong to at least one member of $\mathcal{C}'$ and $|\mathcal{C}'|$ is minimized, see [9, SP5].

In the reduction, as illustrated in Figure 4, we transform any instance of SetCover as follows. Every element $u_i \in U$ corresponds to one terminal pair $\{s_i, t_i\}$. We let these terminal pairs be drawn one below another in the plane graph we construct, the order is arbitrary. Each occurrence of u_i in a set $C_j \in \mathcal{C}$ corresponds to one s_i-t_i-path. An s_i-t_i-path follows a straight line from s_i to t_i, however contains one peak up. The position of the peak denotes the set C_j in which u_i occurs for that particular occurrence. For two different occurrences in the same set $u_i, u_j \in C_l$, we let the corresponding paths be non-intersecting, by aligning their peaks together in C_l position. If two elements occur in two different sets,

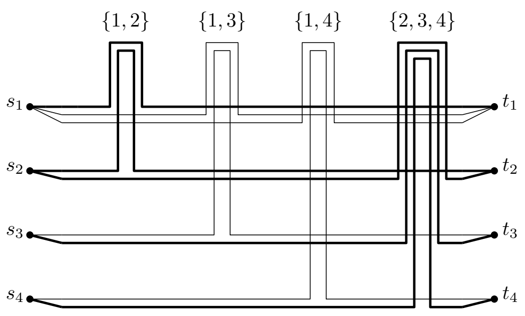


Figure 4 Reduction from set cover. Every element $a_i \in S$ is transformed into a terminal pair $\{s_i, t_i\}$. Each occurrence of an element a_i in a subset $C_j \in C$ is transformed into an s_i-t_i-path, such that two paths are disjoint if and only if they represent elements of the same set. Example with $S = \{1, 2, 3, 4\}$, and $C = \{\{1, 2\}, \{1, 3\}, \{1, 4\}, \{2, 3, 4\}\}$. The chosen paths are drawn bold and correspond to sets $\{1, 2\}$ and $\{2, 3, 4\}$. Note that the terminal pair $\{s_2, t_2\}$ is covered twice.

the corresponding paths intersect because their peaks are not aligned. The peaks follow the shape shown in Figure 4. By this construction, two paths of different terminal pairs can be scheduled in the same round if and only if the corresponding elements belong to the same set in $\mathcal{C}$. The minimum number of rounds needed to schedule all terminal pairs equals $|C'|$.

It is easy to notice that the above reduction is approximation preserving. The SetCover problem is APX-complete when the number of occurrences of an element in sets of $\mathcal{C}$ is bounded by any constant $b \geq 2$ [6]. Hence, the claim of the theorem follows. ◀

4.2 R-VDP-Sep: Separating cut

In this section we consider instances with a separating cut, i.e., where the terminals appear in a counterclockwise traversal of the outer face in the sequence $s_1, s_2, \ldots, s_k, t_{\pi(1)}, t_{\pi(2)}, \ldots, t_{\pi(k)}$ for some permutation π. The separating cut imposes an order structure on the set of all fixed paths $\bigcup \mathcal{P} = \bigcup_{i=1\ldots k} \mathcal{P}_i$. For any two paths $P_i \in \mathcal{P}_i$, $P_j \in \mathcal{P}_j$ we say that $P_i < P_j$ if P_i does not intersect with P_j and $i < j$. Let us denote this order by $(\bigcup \mathcal{P}, <_{\bigcup \mathcal{P}})$. It is easy to see that $<_{\bigcup \mathcal{P}}$ is transitive due to the separating cut, and that $(\bigcup \mathcal{P}, <_{\bigcup \mathcal{P}})$ is a partially ordered set, in short *poset*. We define the *compatibility graph* H for an instance of R-VDP-Sep as follows. There is a vertex in H for each path $P \in \bigcup \mathcal{P}$ and an edge between two vertices if and only if the corresponding paths are disjoint.

▶ **Theorem 6.** *The compatibility graph H of any instance of* R-VDP-Sep *is a comparability graph.*

Proof. Since the edges of H correspond to the order relation of the poset $(\bigcup \mathcal{P}, <_{\bigcup \mathcal{P}})$, H is a comparability graph. ◀

It is well known that comparability graphs and their complements, called co-comparability graphs, are perfect graphs. Many NP-hard problems are polynomial for perfect graphs, among others the coloring problem and the problem of finding a clique of maximum weight in a graph with weights on nodes, see e.g. [11]. Also, for a perfect graph, the size of a maximum clique is equal to the chromatic number of the graph.

A consequence of Theorem 6 is that once a path is selected for each terminal pair, the assignment of paths to a minimum number of rounds is solvable in polynomial time, as the problem is equivalent to the coloring problem of the co-comparability graph $\overline{H}$. It follows that R-VDP-Sep with $p = 1$ is solvable in polynomial time (as in this setting there is no choice for selecting a path for each terminal). Alternatively, we can see R-VDP-Sep with $p = 1$ as the problem of covering a poset with a minimum number of *chains*, for which efficient polynomial time algorithms are known. A *chain* of a poset $(\bigcup \mathcal{P}, <_{\bigcup \mathcal{P}})$ is a subset of $\bigcup P$ of totally ordered elements. A *chain cover* of the poset is a set of chains such that every element of $\bigcup P$ is in (at least) one chain. Thus, a chain of the poset $(\bigcup \mathcal{P}, <_{\bigcup \mathcal{P}})$ corresponds to paths that can be scheduled in the same round. Since $p = 1$, all paths need to be scheduled, and a chain cover of minimum cardinality corresponds to routing of the paths in minimum number of rounds. The problem of covering a poset with a minimum number of chains has been well studied (see for example the characterization of solutions known as the Dilworth's theorem [5]), and can be solved in polynomial time by computing a maximum matching in a related bipartite graph [7].

▶ **Corollary 7.** R-VDP-Sep *with $p = 1$ can be solved in polynomial time.*

4.3 R-VDP-Sep: Separating cut, $p \geq 2$

Since R-VDP-Sep is APX-complete, the question arises how well one can approximate variants of the routing-in-rounds problem. SetCover cannot be approximated within $O(1 - \varepsilon) \ln |S|$, see [6]. There is, however, a B-approximation algorithm for SetCover if each element is covered by at most $B \geq 2$ sets, see [12]. In the following, we give a p-approximation algorithm for R-VDP-Sep. Let H be the compatibility graph of an input instance of R-VDP-Sep as defined in Section 4.2. Let $\overline{H}$ be the complement graph of H. Theorem 6 implies that $\overline{H}$ is a co-comparability graph, and thus a perfect graph. Consider the following integer program to calculate the minimum number of rounds r needed to schedule all terminal pairs of R-VDP-Sep instance:

$$\text{(IP)} \quad \min \quad r \tag{1a}$$

$$\text{s.t.}$$

$$\sum_{P_j \in \mathcal{P}_i} x_{ij} = 1 \qquad \forall \{s_i, t_i\} \in T \tag{1b}$$

$$\sum_{x_{ij} \in C} x_{ij} \leq r \qquad \forall \text{ clique } C \in \overline{H} \tag{1c}$$

$$x_{ij} \in \{0, 1\} \tag{1d}$$

The binary variables x_{ij} denote whether the j'th path from set $\mathcal{P}_i$ is selected. Constraints (1b) require that for each terminal pair $\{s_i, t_i\}$, $i = 1, \ldots, k$, exactly one path in the corresponding set $\mathcal{P}_i$ is chosen. Further, there are exponentially many Constraints (1c) that require that no more than r paths from each (maximal) clique C in $\overline{H}$ are chosen.

▶ **Lemma 8.** *The value r^* of an optimal solution to (IP) equals the minimum number of rounds R needed to schedule a corresponding instance of* R-VDP-Sep.

Proof. Consider an optimal solution of (IP). Variables x_{ij} represent the choice of paths. Consider the subgraph I of $\overline{H}$ induced by the nodes corresponding to the chosen paths. Graph I is a conflict graph for an instance, where there is just one path given per terminal pair (the chosen path). Recall, that when there is just one path per terminal pair, assigning the minimum number of rounds is equivalent to coloring I with minimum number of colors.

The minimum number of colors needed to color I is in turn equal to the size of maximum clique in I, because I is a perfect graph (as it is a subgraph of a perfect graph and in perfect graphs the chromatic number is equal to the clique number). In an optimal solution of (IP), the paths are chosen in a way such that the clique number r^* in I is minimal. Thus, the minimal number of rounds needed in I is equal to $R = r^*$. Hence, the lemma follows. ◀

We note that I in the proof above corresponds to an instance of R-VDP-Sep with $p = 1$, so the actual labels r_i for each chosen path $P_i \in \mathcal{P}_i$ can be found in polynomial time by Corollary 7.

Denote by (LP) the linear relaxation of (IP), and by (LP') the linear program (1a)-(1b). Note that constraints (1c) can be separated in polynomial time. That is, given a feasible solution to (LP'), we can find a violated constraint of (1c), if there is one, by finding a maximum weighted clique in $\overline{H}$, with node weights given by the values of the variables x_{ij}. If this clique does not violate (1c), no other clique does. By the polynomial equivalence of optimization and separation, see [11], (LP) can be solved in polynomial time. We obtain the desired approximation by rounding any fractional values of the x_{ij}. For each $\{s_i, t_i\}$ pair, we choose an x_{ij} with maximum value, denoted by $\overline{x}_i$, and round it to 1. We round the remaining x_{ik}, $k \neq i$ to 0.

▶ **Theorem 9.** R-VDP-Sep *with at most* p *fixed paths per terminal pair can be approximated within a factor of* p.

Proof. Let x^* be an optimal solution to (LP) with objective value r^*. Denote by R the value of an optimal solution to R-VDP-Sep. Clearly, $R \geq r^*$. The rounded values are feasible with respect to equations (1b). Given that there are at most p paths per terminal pair, we have $\overline{x}_i \geq 1/p$ for all terminal pairs $\{s_i, t_i\} \in T$. Hence, each x_{ij} is rounded up by a factor at most p. Therefore, equations (1c) are satisfied for a right hand side of $r^* \cdot p$. Hence, the objective value of the returned solution is at most $p \cdot R$. ◀

References

1 Brenda S. Baker. Approximation algorithms for NP-complete problems on planar graphs. *Journal of the ACM*, 41(1):153–180, 1994.

2 Francesco Corman, Andrea D'Ariano, Dario Pacciarelli, and Marco Pranzo. A tabu search algorithm for rerouting trains during rail operations. *Transportation Research Part B: Methodological*, 44(1):175–192, 2010.

3 Andrea D'Ariano. *Improving Real-Time Dispatching: Models, Algorithms and Applications.* PhD thesis, TRAIL Research School, The Netherlands, 2008.

4 Andrea D'Ariano, Dario Pacciarelli, and Marco Pranzo. A branch and bound algorithm for scheduling trains in a railway network. *European Journal of Operational Research*, 183(2):643–657, 2007.

5 Robert P. Dilworth. A decomposition theorem for partially ordered sets. *The Annals of Mathematics*, 51(1):161–166, 1950.

6 Uriel Feige. A threshold of ln n for approximating set cover. *J. ACM*, 45(4):634–652, 1998.

7 Lester R. Ford and Delbert R. Fulkerson. *Flows in Networks.* Princeton Univ. Press, Princeton, N.J., 1962.

8 Steven Fortune, John E. Hopcroft, and James Wyllie. The directed subgraph homeomorphism problem. *Theor. Comput. Sci.*, 10:111–121, 1980.

9 Michael R. Garey and David S.Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman, 1979.

10 Fanica Gavril. Algorithms for a maximum clique and a maximum independent set of a circle graph. *Networks*, 3(3):261–273, 1973.

11 Martin Grötschel, László Lovász, and Alexander Schrijver. *Geometric Algorithms and Combinatorial Optimization*, volume 2 of *Algorithms and Combinatorics*. Springer, 1993.

12 Dorit S. Hochbaum. Approximation algorithms for the set covering and vertex cover problems. *SIAM Journal on Computing*, 11(3):555–556, 1982.

13 Yusuke Kobayashi and Christian Sommer. On shortest disjoint paths in planar graphs. *Discrete Optimization*, 7:234–245, 2010. Announced at ISAAC 2009.

14 Jan Kratochvíl. String graphs. II. Recognizing string graphs is NP-hard. *Journal of Combinatorial Theory, Series B*, 52(1):67–78, 1991.

15 Leo G. Kroon, H. Edwin Romeijn, and Peter J. Zwaneveld. Routing trains through railway stations: complexity issues. *European Journal of Operational Research*, 98(3):485–498, 1997.

16 David Lichtenstein. Planar formulae and their uses. *SIAM Journal on Computing*, 11(2):329–343, 1982.

17 Richard Lusby, Jesper Larsen, Matthias Ehrgott, and David Ryan. Railway track allocation: models and methods. *OR Spectrum*. To appear.

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

19 Alessandro Mascis and Dario Pacciarelli. Job-shop scheduling with blocking and no-wait constraints. *European Journal of Operational Research*, 143(3):498–517, December 2002.

20 Heike Ripphausen-Lipa, Dorothea Wagner, and Karsten Weihe. *Combinatorial optimization : papers from the DIMACS Special Year*, volume 20 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, chapter Efficient Algorithms for Disjoint Paths in Planar Graphs, pages 295–354. 1995.

21 Neil Robertson and Paul D. Seymour. Graph minors XIII. The disjoint paths problem. *J. Comb. Theory, Ser. B*, 63(1):65–110, 1995.

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