THE MAXIMUM CAPACITY OF A LINE PLAN IS INAPPROXIMABLE

CHRISTINA PUHL AND SEBASTIAN STILLER

Abstract. We consider a basic subproblem which arises in line planning with upper capacities: How much can be routed maximally along all possible lines? The essence of this problem is the *Path Constrained Network Flow* (PCN) problem. We explore the complexity of this problem. In particular we show that it is as hard to approximate as MAX CLIQUE. We also show that the PCN problem is hard for special graph classes, interesting both from a complexity and from a practical perspective. Finally, we present a relevant graph class for which there is a polynomial-time algorithm.

1. Introduction

Motivation. A classical step in the hierarchy of planning scheduled transportation networks is *line planning*. Its task is to determine the lines and their frequencies along which the transportation service shall be offered. Given a network, i.e., a directed graph with upper capacities on the arcs, a *line plan* is a set of lines, i.e., paths in the network, and an integer assigned to each line, its *frequency*. A line plan is *feasible* if it satisfies a given transportation demand and respects the upper arc-capacities of the underlying network. Moreover, the lines that can be used, i.e., the trips a physical train and its crew can serve, are subject to various regulations such as length bounds, limits to the number of terminals and several other, partly very specific requirements. Therefore, line planning often relies on a so-called *line pool*, i.e., an explicitly given set of possible path to which the line plan is restricted.

The objective in line planning is usually some minimization of operation cost or cost of passengers‘ discomfort. Special events, like world cups, fairs, concerts, or temporary exhibitions could create exceptional demand for which special line plans have to be designed. In other cases, construction or maintenance works create exceptionally low upper capacities in the network. Again special line plans are required. Finally, even daily, high utilization of a scarce infrastructure may demand for the construction of a line plan that makes the system work to its capacity. Therefore, it is a natural subproblem of line planning to ask for the maximum demand that can be met given a network, upper arc-capacities and a line pool.

E-mail: puhl@math.tu-berlin.de, stiller@math.tu-berlin.de

This work has partially been supported by the ARRIVAL project, within the 6th Framework Program of the European Commission under contract no. FP6-021235-2.

We examine the complexity of this question, showing that even its single-source, single-sink version is **NP**-complete and in a strong sense inapproximable. Moreover, we study the problem on special graph classes like planar graphs, graphs with bounded treewidth, and a certain class derived from trees, for which we devise a non-trivial, polynomial-time algorithm.

Our complexity results also apply to the closely related Path Constraint Network Flow Problem, which has already been mentioned in [4].

Related Work. Usually an instance of a line planning problem is given by a directed graph $G = (V, A)$, with arc-capacities $c : A \to \mathbb{R}_+$, and projected demands, e.g., how many passengers intend to travel from node x to node y. Moreover, we are given explicitly or implicitly the set of possible lines in G, i.e., the *line pool* denoted by $\mathcal{P}$. A feasible line plan is an assignment of integer frequencies f_p to the lines in $p \in \mathcal{P}$, such that the capacity of each arc a is respected, i.e., $\sum_{p \in \mathcal{P}: a \in p} f_p \leq c(a)$, and the demand can be fulfilled.

Whether the demand, i.e., a certain amount of transportation for each origin-destination pair, can be fulfilled by a *fixed* line plan is itself a non-trivial problem. The passengers have to be routed along the chosen lines respecting on their part the capacities offered by the line plan. In practice, the routing constructed by a central optimization may not coincide with the routing actually chosen by the individual passengers. Hereby, violations of capacities may arise in the real, individual routing even though a feasible, central routing exists.

There are different approaches in the literature to address this issue. For example, in the work of Bussieck et al. [2] a system split in optimization is used. Prior to constructing the line plan the passengers are routed through the network G according to their origin and destination, and some assumptions on their routing behavior. This entails a (fractional) multi-commodity flow problem. The total resulting flow on an arc a is interpreted as its *lower arc capacity* $\ell(a)$. Then an instance of a line planning problem is a quintuple $(G, c, \ell, \mathcal{P}, \text{cost})$ consisting of a network, upper and lower capacities on each arc, a line pool, and some cost function cost. The optimization problem can be expressed as:

$$\begin{aligned} &\min \ \text{cost}(f) \\ \ell(a) \leq \sum_{p \in \mathcal{P}: a \in p} f_p \leq c(a) \quad &\forall a \in A \\ f \geq 0 \end{aligned}$$

For f restricted to integer vectors, this is known to be NP-hard [2].

A different way to define the demand satisfaction is pursued by [1, 8]. Here the multi-commodity flow problem of routing the passengers is part of the line planning optimization procedure. The articles of Borndörfer and Pfetsch, and of Bussieck et al. contain a comprehensive overview on the literature for their specific approach.

Our question is basic to both approaches, and crucial whenever upper capacities become tight.

The problem we consider is a slight specialization of the Path Constraint Network Flow (PCN) problem defined already in Gary and Johnson ([4], Problem ND34). They stated that the PCN problem is—among some other

complexity features—**NP**-complete. Our hardness and inapproximability results carry over to the original PCN as we consider a specialization.

For the proof of hardness Garey and Johnson refer to a private communication with H.J. Prömel. To our knowlege no proof has ever been published and Prömel himself in a further private communication with the authors kindly stated that he was no longer in possession of one. The proofs we give recover all results stated in [4] and add several complexity results, in particular the *inapproximability* of the optimization problem. Moreover, we add results for special graph classes.

Definition of the PCN Problem. In the classical flow theory, flow is send in a network from source to sink respecting arc capacities. In the 60s Ford and Fulkerson showed that any flow can be decomposed in path flows and circle flows [3]. In the general case the set of paths used for the decomposition is not restricted. In our case we are given a set of paths $\mathcal{P}$. We are now interested in flows that can be decomposed using only the paths in $\mathcal{P}$. We give the formal definition of the PCN problem according to Gary and Johnson [4]:

Definition 1.1 (Path Constrained Network Flow Problem)**.** Let $G = (V, A)$ be a directed graph with capacities $c(a) \in \mathbb{N}$ for each $a \in A$, a source $s \in V$ and a sink $t \in V$. Furthermore a collection $\mathcal{P}$ of directed paths in G and a requirement $K \in \mathbb{N}$ is given. The set of paths $\mathcal{P}_a \subset \mathcal{P}$ is the set of all paths in $\mathcal{P}$ containing the arc a. The question is whether there is a function $y : \mathcal{P} \to \mathbb{N}$ such that for the flow function $f : A \to \mathbb{N}$ defined by $f(a) = \sum_{p \in \mathcal{P}_a}$, the following three conditions hold:

(1) $f(a) \leq c(a)$ for all $a \in A$,
(2) for each $v \in V \backslash \{s, t\}$, flow is conserved at v, and
(3) the net flow into t is at least K.

We will consider two slightly different PCN problems: the (s, t)-PCN by restricting the set of paths $\mathcal{P}$ to a set of (s, t)-paths and the (s, t)-PCN$_1$ problem by additionally demanding unit capacities on the arcs. Note that flow conservation (2) is automatically given in this case. Due to the unit capacities any solution for the (s, t)-PCN$_1$ consists of arc disjoint (s, t)-paths. It is easy to see that any result of hardness for the (s, t)-PCN$_1$ holds for the PCN problems.

As we also consider inapproximability we have to define a PCN optimization problem. Thereby, the objective is to maximize the value of the net flow K. We will refer to both problems as PCN problem, as it will be clear from the context whether the optimization or the decision problem is meant.

Results and Techniques. We give positive and negative results for the (s, t)-PCN problem and the (s, t)-PCN$_1$ problem. Note that the hardness results carry over to the more general cases.

In detail we proof **NP**-hardness for (s, t)-PCN$_1$ problem, the questions whether its integrality gap, respectively its duality gap is zero. We proof that the maximum value of an (s, t)-PCN$_1$ cannot be approximated within a constant factor unless, $\mathbf{P} = \mathbf{NP}$. We extend the inapproximability of the (s, t)-PCN$_1$ problem for graphs with bounded treewidth and planar graphs, and for the (s, t)-PCN problem on grid graphs.

We will present two paradigmatic reductions in detail. All hardness and inapproximability results are shown by reductions similar to those. We consider these reductions paradigmatic because they introduce a new and powerful technique to analyze complexity of line planning problems. Path constraint network flows are a central aspect of line planning.

Our factor preserving reduction from MAX CLIQUE for the inapproximability is reversible. Therefore, any practical algorithm for MAX CLIQUE can easily be turned into an algorithm for the (s,t)-PCN$_1$ problem.

Finally, we devise a polynomial time algorithm for the (s,t)-PCN$_1$ problem on (s,t)-extended outtrees, which we define in this paper. The problem on this graph class is equivalent to the transshipment problem on outtrees. Moreover, (s,t)-extended outtrees are a generalization of trees motivated by transportation systems, which are trees except for the connection to a common depot of the vehicles.

Note that the fractional PCN problem in general is easy as it can be formulated as an LP of polynomial size.

Structure of this Paper. In Section 2 we compile the general complexity results and the paradigmatic proofs. In Section 3 and 4 we give the positive and negative results for special graph classes. In particular, we present the algorithm for (s,t)-extended outtrees.

2. Complexity and Inapproximability of the PCN Problem

In this section we show the **NP**-completeness of the PCN problem. All PCN problems considered are in **NP**: Given a path multiplicity vector $y \in \mathbb{N}^{|\mathcal{P}|}$ we can calculate its value in polynomial time, construct the corresponding flow and check its feasibility. By our reductions for the PCN problem it will become clear that a certificate that would only comprise the flow without the decomposition could not be checked in polynomial time to be decomposable into paths in $\mathcal{P}$ unless $\mathbf{P} = \mathbf{NP}$.

We start by showing that the (s,t)-PCN$_1$ problem is strongly **NP**-complete by using a reduction from 3SAT.

Recall that a decision problem $\mathcal{X}$ is called strongly **NP**-hard, iff there is a polynomial p such that the restriction of $\mathcal{X}$ to the set of instances x where the largest integer in the input of x is bounded from above by $p(\text{inputsize}(x))$ is still **NP**-hard (cf. [6]). (We can assume that the input contains no numbers other than integers.) In other words, even a unary encoding of the numbers in the input, would not allow for algorithms polynomial in the input size, unless $\mathbf{P} = \mathbf{NP}$.

Theorem 2.1. *The (s,t)-PCN$_1$ is strongly* **NP***-complete.*

Proof. We reduce from 3SAT. Let I be an instance of 3SAT with n variables $x_1, ..., x_n$ and m clauses $C_1, ..., C_m \subseteq \{x_1, \overline{x}_1, \ldots, \overline{x}_n\}$ with $|C_j| = 3$.

We start with the construction of G and $\mathcal{P}$. The set $\mathcal{P}$ contains for every variable x_i a path p_{x_i} and a path $p_{\overline{x}_i}$ both crossing the arc e_i. For every clause C_j there are three paths $\tilde{p}_{j,1}, \tilde{p}_{j,2}$ and $\tilde{p}_{j,3}$ in $\mathcal{P}$ traversing the arc e_{n+j} and three arcs $c_{j,1}, c_{j,2}$ and $c_{j,3}$. Each arc $c_{j,i}$ represents exactly one literal $z_{j,i}$ of C_j. The path $\tilde{p}_{j,i}$ traverses arc $c_{j,i}$. Any other path p_z with $z \in \{x_1, \overline{x}_1, ..., x_n, \overline{x}_n\}$ contains arc $c_{j,i}$ if and only if $z_{j,i} = \overline{z}$ (Fig. 1). To complete the construction of an (s,t)-PCN$_1$ instance we set K to $n + m$.

Note the following properties of the construction:

(1) Any feasible function y must meet $y(p_{x_i}) + y(p_{\overline{x}_i}) \leq 1$ and $y(\tilde{p}_{j,1}) + y(\tilde{p}_{j,2}) + y(\tilde{p}_{j,3}) \leq 1$ for capacity reasons on the arcs e_ℓ.
(2) Any feasible function y with net flow $n + m$ must satisfy $y(p_{x_i}) + y(p_{\overline{x}_i}) = 1$ and $y(\tilde{p}_{j,1})+y(\tilde{p}_{j,2})+y(\tilde{p}_{j,3}) = 1$ due to the cut $\{e_1, \ldots, e_{n+m}\}$.

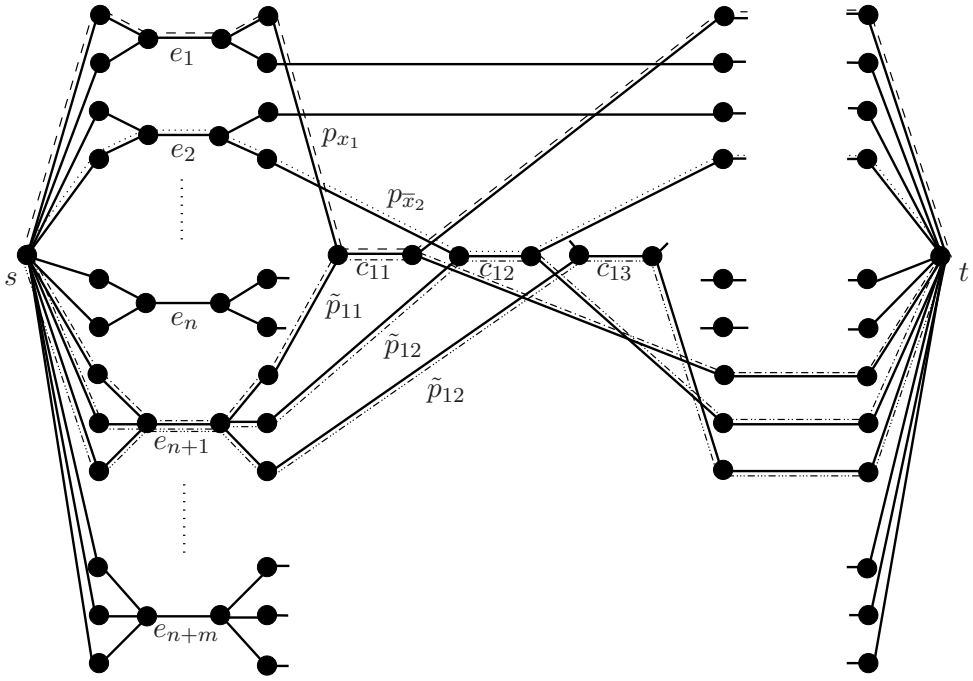

FIGURE 1. The set of paths $\mathcal{P}$ contains $2n + 3m$ s, t- paths.

Now we show, that there exists a function $y : \mathcal{P} \to \mathbb{N}$ with value $n + m$ and $\sum_{p \in \mathcal{P}_a} y(p) \leq 1$ for all arcs $a \in A$ if and only if the instance I can be satisfied.

Let $y : \mathcal{P} \to \mathbb{N}$ be a function with $\sum_{p \in \mathcal{P}_a} y(p) \leq 1$ and $\sum_{p \in \mathcal{P}} y(p) = n+m$. Define the vector x by

$$x_i := \begin{cases} 1 & \text{if } y(p_{x_i}) = 1 \\ 0 & \text{if } y(p_{\overline{x}_i}) = 1. \end{cases}$$

Due to the obeservation (2) x is well defined.

We now show that x satisfies all clauses. Suppose there exists a clause C_j which is not satisfied. Then $y(p_{z_1}) = y(p_{z_2}) = y(p_{z_3}) = 1$ for $z_i = \overline{z}_{j,i}$ and due to capacity no other path passing arc $c_{j,1}, c_{j,2}$ or $c_{j,3}$ contains any flow. Therefore $y(\tilde{p}_{j,1}) + y(\tilde{p}_{j,2}) + y(\tilde{p}_{j,3}) = 0$, a contradiction to observation (2).

Let now x be a vector that satisfies all clauses. We compose a function $y : \mathcal{P} \to \mathbb{N}$ with $\sum_{a \in \mathcal{P}_a} y(p) \leq 1$ by

$$y(p_{x_i}) = \begin{cases} 1 & \text{if } x_i = 1 \\ 0 & \text{else} \end{cases} \text{ and } y(p_{\overline{x}_i}) = \begin{cases} 1 & \text{if } x_i = 0 \\ 0 & \text{else.} \end{cases}$$

For any clause there exists a literal $z_{j,i^*} = 1$. Therefore no flow is passing through arc c_{j,i^*}. Define $y(\tilde{p}_{j,i^*}) = 1$ and $y(\tilde{p}_{j,i}) = 0$ for $i \neq i^*$. The function y therefore fulfills all capacity restrictions and has the value $m + n$.

□

The constuction in this proof leads to two further reductions concerning the dual gap and the LP- and IP-relation of the PCN optimization problems.

The (s,t)-PCN (or (s,t)-PCN$_1$) problem with non-integral flow is equivalent to linear programming and therefore polynomially solvable. But the question whether the best rational flow fails to exceed the best integral flow is **NP**-complete. With the construction of G and $\mathcal{P}$ in Theorem 2.1 this is easy to see. A rational flow $y^* : \mathcal{P} \to \mathbb{Q}^{|\mathcal{P}|}$ with value $n + m$ and satisfying all capacities is always given by $y^*(p_z) = \frac{1}{2}$ for all $z \in \{x_1, \overline{x}_1, \ldots, x_n, \overline{x}_n\}$ and $y^*(\tilde{p}_{j_i}) = \frac{1}{3}$ for all $j = 1, \ldots, m$ and $i = 1, 2, 3$.

Theorem 2.2. *Given an (s,t)-PCN$_1$ instance, the decision whether the best rational flow fails to exceed the best integral flow is* **NP***-complete.*

Furthermore the decision problem whether for the (s,t)-PCN$_1$ optimization problem the dual gap equals zero is **NP**-complete. The dual optimization problem asks for a subset of arcs, A', with minimum cardinality $|A'|$ such that every path $p \in \mathcal{P}$ traverses at least one arc of A'. For the (s,t)-PCN$_1$ instance in Theorem 2.1 an optimal dual solution A' consists of the arcs $\{e_1, \ldots, e_{n+m}\}$ with the value $n + m$. This observation is sufficient to show the following theorem:

Theorem 2.3. *Given an (s,t)-PCN$_1$ instance, the decision whether the dual gap is greater than zero is* **NP***-complete.*

We are now interested in the approximability of the PCN optimization problems. A certain type of reduction, called L-reduction, preserves approximability (whereas in general polynomial tranformations do not). Recall that to establish an L-reduction (cf. [6]) from an optimization problem $\mathcal{X}$ to an optimization problem $\mathcal{X}'$ we have to devise a pair of functions f and g, both computable in polynomial time, and two constants $\alpha, \beta > 0$ such that for any instance x of $\mathcal{X}$:

- $f(x)$ is an instance of $\mathcal{P}'$ with $\mathrm{OPT}(f(x)) \leq \alpha \mathrm{OPT}(x)$;
- For any feasible solution y' of $f(x)$, $g(x, y')$ is a feasible solution of x such that $|c_x(g(x, y')) - \mathrm{OPT}(x)| \leq \beta |c_{f(x)}(y') - \mathrm{OPT}(f(x))|$,

where c_x is the cost function of the instance x.

In the following theorem we introduce an L-reduction from MAX CLIQUE to the (s,t)-PCN$_1$ problem. In Section 3 this reduction from MAX CLIQUE will be transfered to other PCN instances with graphs from special graph classes.

Theorem 2.4. *The* MAX CLIQUE *problem is L-reducible to the (s,t)-PCN$_1$ problem with constants $\alpha = \beta = 1$ and vice versa.*

Proof. We define an L-reduction from MAX CLIQUE to (s,t)-PCN$_1$. Given a graph $G = (V, E)$ as a MAX CLIQUE instance I we want to construct an (s,t)-PCN$_1$ instance I' such that for any clique $C \subseteq V$ in G one easily

obtains a set of arc disjoint (s,t)-paths with the same cardinality $|C|$ and vice versa.

To this end, the instance I' shall consist of a graph $G' = (V', E')$ and a set of (s,t)-paths $\mathcal{P}'$ such that for any $v \in V$ there exists an (s,t)-path $p_v \in \mathcal{P}'$, and we have: p_v and p_u are arc disjoint if and only if the arc (u,v) is in E. We will call a graph G' together with a set of paths $\mathcal{P}'$ that fulfills those requirements *clique-reducing*.

There is an easy way to construct a clique-reducing pair $(G', \mathcal{P}')$ for a MAX CLIQUE instance G with at most n^2 nodes and at most $6n^2 + n$ arcs in G': Assume V to be indexed by $\{1, \dots, n\}$. We start with n-parallel (disjoint) paths from s to t. We change the graph and the paths successivly from 1 to n. For each $n \geq i > j \geq 1$ and $\{v_i, v_j\} \notin E$ we deviate the path p_{v_i} corresponding to node v_i such that it intersects on an exclusive arc of path p_{v_j}. Caerfully inserting necessary arcs for this construction we can end up with a path of at most $5n + 2$.

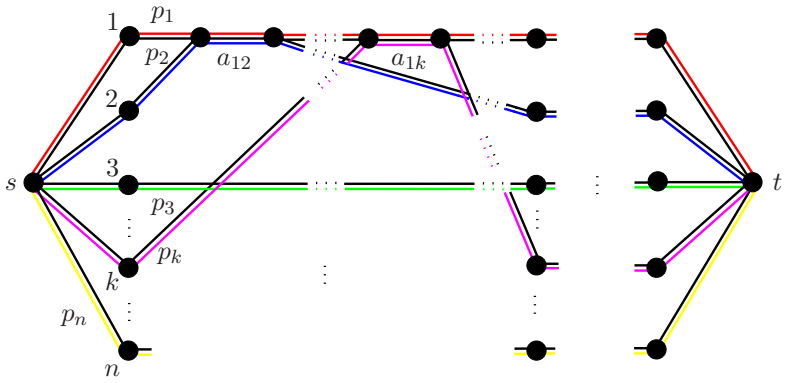

FIGURE 2. The arc $(1,2)$ and $(1,k)$ are not in E, but the arc $(1,3)$. Therefore the (s,t)-paths p_1 and p_2, and p_1 and p_k of $\mathcal{P}'$ have a common arc, whereas p_1 and p_3 are disjoint.

Let $\overline{\mathcal{P}}$ be a subset of $\mathcal{P}'$ and $\overline{C} := \{v \in V | p_v \in \overline{\mathcal{P}}\}$. By construction $\overline{C}$ is a clique if and only if all paths in $\overline{\mathcal{P}}$ are arc disjoint. Therefore any optimal solution of the (s,t)-PCN$_1$ instance yields an optimal solution for the MAX CLIQUE instance with the same optimal value.

In a similar way an L-reduction from (s,t)-PCN$_1$ to MAX CLIQUE can be constructed. □

The complexity of MAX CLIQUE is well studied and belongs to the highest class, Class IV, of approximation problems as classified in [5]. In particular, an $n^{0.5-\epsilon}$ approximation is **NP**-hard. Due to the given L-reduction these results transfer to the PCN problems.

Corollary 2.5. *There exists an $\epsilon > 0$ such that approximating (s,t)-PCN$_1$ within a factor of $|\mathcal{P}|^\epsilon$ is* **NP***-hard.*

By the L-reduction from (s,t)-PCN$_1$ to MAX CLIQUE any algorithm, heuristic or algorithmic result for the MAX CLIQUE problem directly transfers to the (s,t)-PCN$_1$ problem. Note that we have no L-reduction from the (s,t)-PCN without unit capacity to MAX CLIQUE.

3. Planar Graphs and Bounded Treewidth

In many cases **NP**-complete problems can be solved efficiently on special graph classes, as the MAX CLIQUE problem on interval or perfect graphs. We studied the PCN problem on graphs with bounded tree width and planar graphs. In both cases the complexity of the problems remained the same.

In the 80s Robertson and Seymour introduced the notion of treewidths [7].

Definition 3.1. A *tree-decomposition* of a graph $G = (V, E)$ is a pair $(\{X_i | i \in I\}, T = (I, F))$ with $\{X_i | i \in I\}$ a family of subsets of V, one for each node of T, and T a tree such that

- $\bigcup_{i \in I} X_i = V$,
- for all arcs $(v, w) \in E$, there exists an $i \in I$ with $v \in X_i$ and $w \in X_i$,
- for all $i, j, k \in I$: if j is on the path from i to k in T, then $X_i \cap X_k \subset X_j$.

The *treewidth of a tree-decompositon* $(\{X_i | i \in I\}, T = (I, F))$ is $\max_{i \in I} |X_i| - 1$. The *treewidth of a graph* G is the minimum treewidth over all possible tree-decompositions of G.

For many **NP**-complete problems on graphs there exists a polynomial, often even a linear algorithm on graphs with bounded treewidth. For the PCN problem this is not the case. The clique-reducible pair $(G', \mathcal{P}')$ in the reduction of the proof of Theorem 2.4 can be constructed such that G' is a chain graph: A *chain graph* consists of n nodes $v_1, \ldots, v_n$ and parallel arcs between two nodes v_i and v_{i+1}. A simple tree-decomposition of a chain graph is definied by $X_i := \{v_i, v_{i+1}\}$ for $i \in I = \{1, ..., n-1\}$ and the path $T = (I, E)$ with arcs $(i, i+1) \in E$. The treewidth of this decomposition is 1. Due to the parallel arcs we can construct an (s, t)-PCN$_1$ instance I' on a chain graph for any MAX CLQIUE instance I such that two paths p_u and p_v are arc disjoint if and only if the vetices u and v are connected by an arc in I (Fig. 3). Therefore we have:

Theorem 3.2. *The MAX CLIQUE problem is L-reducible to the (s, t)-PCN$_1$ problem on graphs with bounded treewidth.*

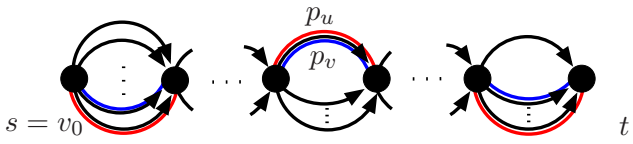

FIGURE 3. The L-reduction in Theorem 2.4 can be transfered to chain graphs.

Since chain graphs are planar the result holds for this graph class, too.

Corollary 3.3. *The (s, t)-PCN$_1$ problem on planar graphs is* **NP***-complete.*

An interesting class of graphs for line-planning are grid graphs since the street network of cities like Manhattan or Mannheim are based on this structur. Moreover the class of grid graphs has unbounded treewidth. But even

on this class of well structured graphs the (s,t)-PCN problem cannot be approximated within a constant factor, unless $\mathbf{P} = \mathbf{NP}$.

Theorem 3.4. *The MAX CLIQUE problem is L-reducible to the (s,t)-PCN problem on grid graphs.*

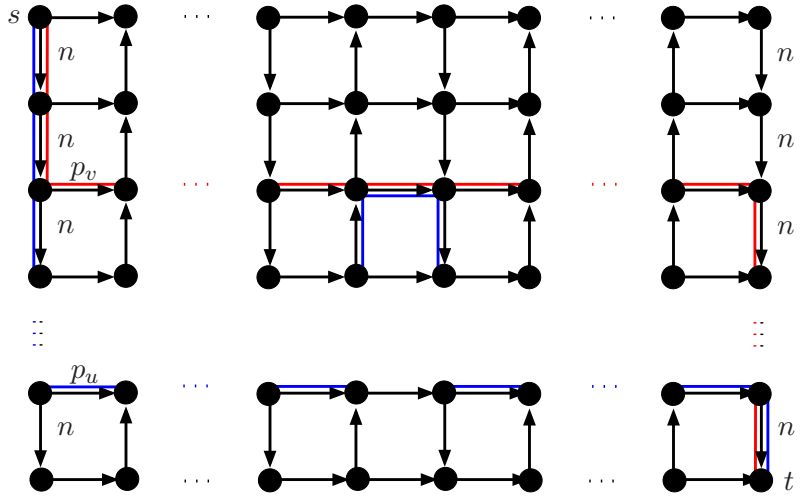

FIGURE 4. The paths p_u and p_v have a common arc since u, v is not in G.

The L-reduction is again based on the construction of Theorem 2.4 (Fig. 4). It is possible to construct the clique-reducible pair, such that the graph is a grid. Yet, in this case it is essential to use arbitrary capacities. In fact we show in the next section, that there is a polynomial time algorithm for the (s,t)-PCN$_1$ problem.

4. Tractable Graph Classes

So far we considered only classes of graphs on which the PCN problems remain difficult. In this section we consider two classes, grid graphs and tree like graphs, on which at least the (s,t)-PCN$_1$ problem is solvable in polynomial time.

(s,t)-PCN$_1$ on Grids. We have seen before that the (s,t)-PCN problem on grid graphs with arbitrary capacities c is **NP**-hard. In the case of unit capacities we can exploit the property of an optimal solution to consist of arc disjoint paths. Since the degree of s is at most 4 any optimal solution consists of at most 4 different paths. To check the feasibility of any subset of the set of paths $\mathcal{P}$ with at most 4 elements is possible in polynomial time. In that way we can compute the optimal solution. As the only feature of grids we used is the constant bound on the degree of s (respectively t), the idea extends to any (s,t)-PCN$_1$ problem for which the minimum of the degree of s and t is a constant.

(s,t)-PCN on Tree-Like Graphs. On trees the (s,t)-PCN problem is trivial since there exists only one path from s to t. If we set $y(p) := \min_{a \in p} c(a)$ with $p \in \mathcal{P}$ we obtain the optimal solution. A natural extension is to consider transshipments on directed trees, which is equivalent to the (s,t)-PCN

on (s,t)-extended outtrees: A graph $G = (V, A)$ is an (s,t)-*extended outtree* if $G - \{s,t\}$ is an outtree (Fig. 5). An *outtree* is a directed tree $T = (V', A')$ with a root node $r \in V'$ such that there is a directed path (not necessarily in $\mathcal{P}$) from r to any other node of V'.

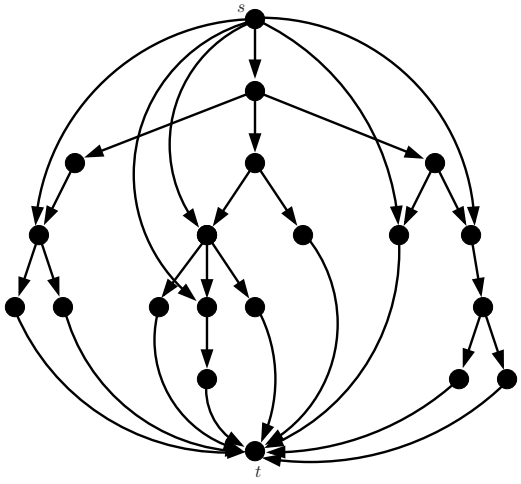

FIGURE 5. An (s,t)-extended outtree

In an (s,t)-extended outtree $G = (V, A)$ we denote the set of nodes incident to s with V_s and those incident to t with V_t. The nodes of the outtree $T = G - \{s,t\}$ can be divided in disjoint sets, called level L_i, according to their distance i to r. An (s,t)-path p in an (s,t)-extended outtree G is uniquely defined by a pair of nodes (a,b) with $a \in V_s$ and $b \in V_t$, thus we will denote this path by $[a,b]$. On this special class of graphs there exists an efficient algorithm for the (s,t)-PCN$_1$ problem. W.l.o.g. we assume for the remainder of this section that every arc is traversed by at least one path of $\mathcal{P}$, and any path ends in a leaf of T.

Theorem 4.1. *The (s,t)-PCN$_1$ problem on (s,t)-extended outtrees is solvable in polynomial time.*

To sketch the proof of Theorem 4.1, i.e., to describe the algorithm, we fix some notation. From now on, let $G = (V, A)$ be an (s,t)-extended outtree and $\mathcal{P}$ the path set of the (s,t)-PCN$_1$ instance $(G, \mathcal{P})$ in question. For any node $u \in V_s$ we denote by $\mathcal{P}_u \subseteq \mathcal{P}$ the subset of paths containing the arc (s,u). The level nodes of the outtree $T = G - \{s,t\}$ with largest distance to r shall be L_k.

Now we devise the pivotal lemma (without proof) for our algorithm, describing the structure of optimal solutions $\mathcal{P}^*$. (As we have unit capacities a solution is specified by a subset of $\mathcal{P}$.)

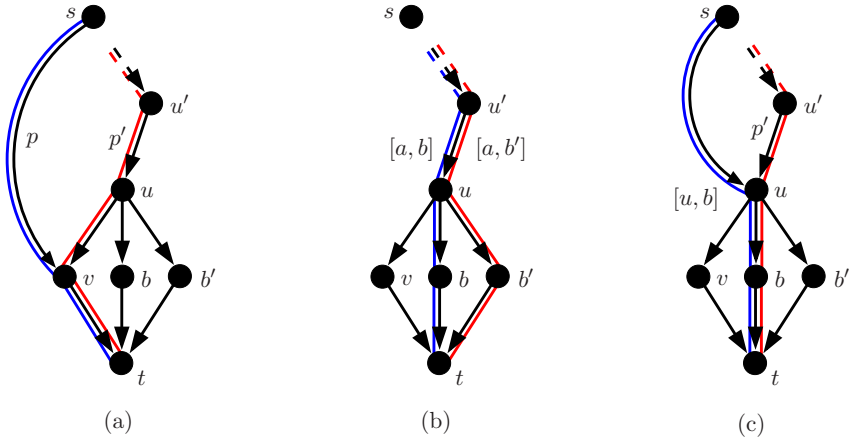

FIGURE 6. (a) Since $v \in V_s \cap V_t$ the path $s, v, t \in \mathcal{P}$. (b) A feasible solution contains at most one path from u to its children C_u. (c) If $u \in V_s$ there exist at most two paths crossing u in a feasible solution.

Lemma 4.2. *In the setting described above, with $k > 2$, $v \in L_k$ a node of the maximum level, and the node u its father, the following holds:*

1. *In case $v \in V_s$, the path $p = (s, v, t) \in \mathcal{P}$, any optimal solution P^* contains a path $p' : v \in p'$, and $\mathcal{P}^{**} = \mathcal{P}^* - p' + p$ is also an optimal solution (Fig. 6 (a)).*
2. *For the case $u \notin V_s$ let C_u be the set of children of u. For an (s,t)-path of an optimal solution $[a, b'] \in \mathcal{P}^*$ entering T at a and leaving at b', and some path $[a, b] \in \mathcal{P}$, where b, $b' \in C_u \setminus V_s$, the set of path $\mathcal{P}^{**} = \mathcal{P}^* - [a, b'] + [a, b]$ is also an optimal solution (Fig. 6 (b)).*
3. *In case $u \in V_s$ and $|\mathcal{P}_u| > 1$, any optimal solution will contain a path (s, u, b, t) for some $b \in C_u \setminus V_s$ (Fig. 6 (c)).*
4. *In case $u \in V_s, \mathcal{P}_u = \{[u, b]\}$, and $b \notin V_s$, any optimal solution $\mathcal{P}^*$ contains a path $p' : b \in p'$. Further, the path p' can be replaced by $[u, b]$ maintaining optimality and feasibility of the solution (Fig. 6 (c)).*

The algorithm is based on dynamically delaying decisions. Throughout the algorithm we will maintain a candidate set for the path of the solution $\mathcal{P}'$, a stack of arces A' to store certain decisions, and we will shrink the graph and correspondingly manipulate the set of paths $\mathcal{P}$.

The basic idea is the following. We shrink the graph from the leaves to the root reducing our PCN instance. But, the decisions which paths shall be used in the parts we shrink cannot be taken completely at the moment of shrinking, because they depend on the part that has not yet been shrunk. Therefore, we take preliminary decisions choosing sub-paths for the solution. These decisions are stored in a stack of acrs A'. In a second phase we de-shrink the graph and can now turn the preliminary sub-paths into complete paths.

The algorithm itself is quite technical and so is the exact proof. In essence the optimality relies on the fact that the constructed set of arcs A' is a

feasible solution for the dual problem. We will now informally explain the algorithm. The exact description is given in the Appendix.

The algorithm consists of two phases. In Phase I the algorithm decides for every node $v \in V_s$ whether the constructed solution will contain a path entering T at node v. To this end, all nodes are checked starting with the ones in the highest levels and going down to the root r. If v is chosen, either the arc (s, v) or (v, x) (for some $x \in C_v$) will be added to the decision saving stack A'. The exact choice of this arc depends on a case distinction, derived from the cases (3) and (4) of Lemma 4.2 explained later.

In addition Phase I constructs a set of candidates $\mathcal{P}' \subseteq \mathcal{P}$ for the path in $\mathcal{P}^*$. The construction ensures that $\mathcal{P}'$ contains for every arc $(a, b) \in A'$ with $a \neq s$ at least one path, which contains arc (a, b) and is arc disjoint to all paths $[x, y] \in \mathcal{P}'$ starting in a level lower than a. The set $\mathcal{P}'$ is constructed by successively deleting paths starting from $\mathcal{P}$ as will be described later.

In Phase II an (s, t)-path $p \in \mathcal{P}' \subset \mathcal{P}$ is assigned to every arc $(a, b) \in A'$.
Phase I. has two steps. First, we take care of for all $v \in V_s \cap V_t$. Then there is a path (s, v, t) according to Lemma 4.2 (1), which wants to be part of our solution.

Therefore, we add (v, t) to A', delete (s, v) and v from G (v is a leaf) and all paths $[a, v]$ from $\mathcal{P}$. Finally, we delete all paths $[a, v]$ from $\mathcal{P}'$ except for (s, v, t).

In the second step all nodes in T are checked starting with the nodes in the highest level L_k. Let u be the father of a node in L_k. We distinguish three cases corresponding to the cases (2) – (4) of Lemma 4.2.

(1) In case $u \notin N_s$, we delete all children C_u of u from u in G, replace all paths $[a, b] \in \mathcal{P}$ with $b \in C_u$ by the path $[a, u]$ and add the arc (u, t) to G.
(2) In case $u \in N_s$ and $|\mathcal{P}_u| > 1$, we save (s, u) in A' and delete (s, u) from G.
(3) In case $u \in N_s$ and $|\mathcal{P}_u| = 1$, we denote the only path in question by $[u, b]$. We save (u, b) in A' and delete (s, u) and (u, b) from G. Furthermore, we delete all paths from $\mathcal{P}$ traversing (u, b). From $\mathcal{P}'$ we remove the same path except for those containing (s, u).

Phase II. We pull the arcs in stack A' one by one, last-in-first-out. For each such arc a we choose an arbitrary path $P \in \mathcal{P}'$ that contains e. In addition, we delete all paths $\tilde{P}$ with $\tilde{P} \cap P \neq \emptyset$ from $\mathcal{P}'$. Then we pull the next arc from A'. Our construction of A' and $\mathcal{P}'$ ensures that for each e such a path is still in $\mathcal{P}'$. Obviously, the resulting solution is an arc disjoint set of paths.

Two natural questions arise from this problem: Is the (s, t)-PCN problem on (s, t)-extended outtrees solvable in polynomial time? Does an algorithm exist which solves the (s, t)-PCN problem on (s, t)-extended trees in polynomial time?

Acknowledgment

We are grateful for discussion with Stefan Hougardy, H. J. Prömel and Ines Spenke.

References

[1] Ralf Borndörfer, Martin Grötschel, and Marc E. Pfetsch. A path based model for line planning in public transport. *Report ZR-05-18, Zuse Institute Berlin*, 2005.

[2] Michael R. Bussieck, Peter Kreuzer, and Uwe T. Zimmernmann. Optimal lines for railway systems. *Eur. J. Oper. Res.*, 96(1):54–63, 1997.

[3] L.R. Ford Jr. and D.R. Fulkerson. *Flows in Networks*. Princeton University Press, Princeton, 1962.

[4] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, New York, 1979.

[5] Dorit S. Hochbaum. *Approximation Algorithms for NP-Hard Problems*. PWS Publishing Company, 1997.

[6] Bernhard Korte and Jens Vygen. *Combinatorial Optimization: Theory and Algorithms*. Springer, 2000/2002.

[7] N. Robertson and P.D. Seymour. Graph minors. i. excluding a forest. *J. Comb. Theory Series B*, 35:39–61, 1983.

[8] Anita Schöbel and Susanne Scholl. Line planning with minimal traveling time. *5th Workshop on algorithmic methods and models for optimization of railways, Dagstuhl Seminar Procedings 06901*, 2006.

Appendix A. An algorithm for the (s,t)-PCN$_1$ problem on (s,t)-extended outtrees

In section 4 we gave a short overview of the idea of the algorithm. The exact description is given below (Alg. 1).

Data: (s,t)-extended outtree $G=(V,A)$,
$\mathcal{P} \subseteq \{[a,b] |\ a \in N_s, b \in V_t\}$ set of paths in G
Result: optimal solution $\mathcal{P}^*$ of the (s,t)-PCN$_1$ instance,
optimal solution A^* of the dual (s,t)-PCN$_1$ instance

$A' := \varnothing$, $\mathcal{P}' := \mathcal{P}$, $\mathcal{P}^* := \varnothing$, $A^* := \varnothing$
$k_{\max}$:= maximum level of G
Define $\mathcal{P}_u := \{p \in \mathcal{P} |$ mit $(s,u) \in p\}$ for all $u \in N_s$

1. Step
forall $v \in N_s \cap V_t$ **do**
  $A' := A' \cup (v,t)$
  Delete v from G
  $\mathcal{P} := \mathcal{P} \setminus \cup_{a \in N_s} [a,v]$ `/*  Delete all paths from` $\mathcal{P}$ `which end in` v `*/`
  $\mathcal{P}' := (\mathcal{P}' \setminus \cup_{a \in N_s} [a,v]) \cup [s,v,t]$

2. Step: Contraction of G and $\mathcal{P}$
$\mathcal{P}_0 := \mathcal{P}$, $\mathcal{P}'_0 := \mathcal{P}'$, $A'_0 = A'$, $G_0 = G$, $i = 0$
while $k_{\max} > 0$ **do**
  $v :=$ node of $L_{k_{\max}}$
  $u :=$ father of v
  $C_u :=$set of all children from u
  if $u \notin N_s$ **then**
    $G_{i+1} := G_i \setminus C_u$ `/*  Delete` C_u `from` G_i `*/`
    $\mathcal{P}_{i+1} := \mathcal{P}_i$
    forall $[a,b] \in \mathcal{P}_i$ *mit* $b \in C_u$ **do**
      $\mathcal{P}_{i+1} := (\mathcal{P}_{i+1} \setminus [a,b]) \cup [a,u]$ `/*  Replace path` $[a,b]$ `by` $[a,u]$ `*/`
      $\mathcal{P}'_{i+1} = \mathcal{P}'_i$, $A'_{i+1} = A'_i$
  else
    if $|\mathcal{P}_u| > 1$ **then**
      $A_{i+1} := A_i \cup (s,u)$
      $G_{i+1} := G_i \setminus (s,u)$ `/*  Delete` (s,u) `from` G_i `*/`
      $\mathcal{P}_{i+1} := \mathcal{P}_i \setminus \mathcal{P}_u$ `/*  Delete` $\mathcal{P}_u$ `from` $\mathcal{P}_{i+1}$ `*/`
      $\mathcal{P}'_{i+1} := \mathcal{P}'_i$
      Regrad u as father node $u \notin N_s$
    else
      $[u,b] = \mathcal{P}_u$
      $A_{i+1} := A_i \cup (u,b)$
      $G_{i+1} = G_i \setminus \{b, (s,u), (u,b)\}$ `/*  Delete` b, (s,u), (u,b) `from` G_i `*/`
      $\mathcal{P}_{i+1} := \mathcal{P}_i$
      forall $[a,b] \in \mathcal{P}_i$ **do**
        $\mathcal{P}_{i+1} := (\mathcal{P}_{i+1} \setminus [a,b])$ `/*  Delete all paths` $[a,b]$ `from` $\mathcal{P}_{i+1}$ `*/`
      $\mathcal{P}'_{i+1} := \mathcal{P}'_i$
      forall $p' \in \mathcal{P}'_i$ *with* $(u,b) \in p'$ *and* $(s,u) \notin p'$ **do**
        $\mathcal{P}'_{i+1} := \mathcal{P}'_{i+1} \setminus p'$
  Update $k_{\max}$, N_s and $\mathcal{P}_u$, $i := i+1$
$A^* := A'_i$

3. Step: Selection of $\mathcal{P}^*$
$A'_{(0)} := A'_i$, $\mathcal{P}'_{(0)} := \mathcal{P}'_i$, $j := 0$
while $A'_{(j)} \neq \varnothing$ **do**
  $(x,y) :=$ last incoming arc of $A'_{(j)}$
  Select $p \in \mathcal{P}'_{(j)}$ with $(x,y) \in p$
  $\mathcal{P}^* := \mathcal{P}^* \cup p$
  $\mathcal{P}'_{(j+1)} := \mathcal{P}'_{(j)} \setminus \{p' \in \mathcal{P}'_{(j)} | p' \cap p \neq \varnothing\}$
  $A'_{(j+1)} := A'_{(j)} \setminus (x,y)$
  $j := j+1$
return A^*, $\mathcal{P}^*$

Algorithm 1: (s,t)-PCN$_1$ on (s,t)-extended outtrees