Packet Routing: Complexity and Algorithms[⋆]

Britta Peis, Martin Skutella, and Andreas Wiese

Technische Universität Berlin, Straße des 17. Juni 136, 10623 Berlin, Germany
{peis,skutella,wiese}@math.tu-berlin.de

Abstract. Store-and-forward packet routing belongs to the most fundamental tasks in network optimization. Limited bandwidth requires that some packets cannot move to their destination directly but need to wait at intermediate nodes on their path or take detours. In particular, for time critical applications, it is desirable to find schedules that ensure fast delivery of the packets. It is thus a natural objective to minimize the makespan, i.e., the time at which the last packet arrives at its destination. In this paper we present several new ideas and techniques that lead to novel algorithms and hardness results.

1 Introduction

In this paper we study the packet routing problem. Given a set of packets in a network originating at possibly different start vertices, we want to transfer them to their respective destination vertices. The goal is to minimize the overall makespan, that is the time when the last packet arrives at its destination. We consider the offline version of the problem in which all information about the network and the packets, in particular the start- and destination vertices, are given in advance. In our routing model, we assume store-and-forward routing. This means that every node can store arbitrarily many packets but each link (a directed or undirected edge) can be used by only one packet at a time. We study the case where the paths of the packets are fixed in advance as well as the case where their computation is part of the problem. Moreover, we distinguish between different types of underlying graphs, e.g., directed graphs, undirected graphs, planar graphs or trees.

This problem has important applications in all settings where packets need to be transferred through a network. The priority of the packets in the schedule is not immediately clear and inappropriate routing rules can lead to inefficient schedules. Delay due to packet latency is not desirable. In particular, in time-critical applications, packets need to be delivered within a certain time frame in order to work accurately. Therefore, we are interested in schedules that guarantee the packets to arrive at their respective destinations as early as possible. Finding such efficient schedules in distributed systems is a challenging task.

⋆ This work was partially supported by Berlin Mathematical School, by DFG research center MATHEON, and by the DFG Focus Program 1307 within the project "Algorithm Engineering for Real-time Scheduling and Routing".

© Springer-Verlag Berlin Heidelberg 2010

1.1 Packet Routing Problem

The packet routing problem is defined as follows: Let $G = (V, E)$ be a directed or undirected graph. A packet $M_i = (s_i, t_i)$ is a tuple consisting of a start vertex $s_i \in V$ and a destination vertex $t_i \in V$. Let $\mathcal{M} = \{M_1, M_2, M_3, ..., M_{|\mathcal{M}|}\}$ be a set of packets.

Then $(G, \mathcal{M})$ is an instance of the *packet routing problem with variable paths.* The problem has two parts: First, for each packet M_i we need to find a path $P_i = (s_i = v_0, v_1, \ldots, v_{\ell-1}, v_\ell = t_i)$ from s_i to t_i such that $\{v_i, v_{i+1}\} \in E$ if G is undirected and $(v_i, v_{i+1}) \in E$ if G is directed for all i with $0 \leq i \leq \ell - 1$. Assuming that it takes one timestep to send a packet along an edge we need to find a routing schedule for the packets such that

– each message M_i follows its path P_i from s_i to t_i and
– each edge is used by at most one packet at a time

We assume that time is discrete and that all packets take their steps simultaneously. The objective is to minimize the makespan, i.e., the time when the last packet has reached its destination vertex. For each packet M_i we define D_i to be the length of the shortest path from s_i to t_i, assuming that all edges have unit length. Moreover, the *dilation* D is defined by $D := \max_i D_i$. It holds that D is a lower bound on the length of an optimal schedule.

Since there are algorithms known to determine paths for routing the packets (see [28,18] or simply take shortest paths) we will also consider the packet *routing problem with fixed paths.* An instance of this problem is a tripel $(G, \mathcal{M}, \mathcal{P})$ such that G is a (directed or undirected) graph, $\mathcal{M}$ is a set of packets and $\mathcal{P}$ is a set of predefined paths. Because the paths of the packets are given in advance they do not need to be computed here. The aim is to find a schedule with the properties described above such that the makespan is minimized. For each packet M_i we define D_i to be the length of the path P_i, again assuming that all edges have unit length. Like above we define the *dilation* D by $D := \max_i D_i$. For each edge e we define C_e to be the number of packets that are routed along edge e. The *congestion* C is then defined by $C := \max_e C_e$. It holds that C and D are lower bounds on the length of an optimal schedule.

Throughout the paper we will use the notation $|S|$ for the length of a schedule S. We call a schedule *direct* if each packet is delayed only in its start vertex. For a packet routing instance I with fixed or variable paths let $OPT\,(I)$ denote a schedule with minimum makespan. For an algorithm $\mathcal{A}$ for the packet routing problem denote by $\mathcal{A}(I)$ the schedule computed by $\mathcal{A}$ for the instance I. The algorithm $\mathcal{A}$ is an α-approximation algorithm if it runs in polynomial time and for all instances I it holds that $|\mathcal{A}(I)| \leq \alpha \cdot |OPT(I)|$. We call α the *approximation ratio* or *performance ratio* of $\mathcal{A}$.

1.2 Related Work

Packet routing and related problems have been widely studied in the literature. Di Ianni [7] shows that the so-called delay routing problem is NP-hard. The proof implies that the packet routing problem is NP-hard as well. Leung et al. [21, chapter 37] study packet routing on different graph classes, including in- and out-trees. In particular, they show that for those trees the farthest-destination-first (FDF)-algorithm works optimally. Busch et al. [5] study the direct routing problem, i.e., the problem of finding the shortest direct schedule. They give complexity results and algorithms for finding direct schedules.

Mansour and Patt-Shamir [22] study greedy scheduling algorithms (algorithms that always forward a packet if they can) in the setting where the paths of all packets are shortest paths. They prove that in this setting every packet M_i reaches its destination after at most $D_i + |\mathcal{M}| - 1$ steps where D_i is the length of the path of M_i and $|\mathcal{M}|$ is the number of packets in the network. Thus, giving priority to the packets according to the lengths of their paths yields an optimal algorithm if we assume that the path-lengths are pairwise distinct.

Leighton et al. [19] show that there is always a routing schedule that finishes within $O(C + D)$ steps. In [20] Leighton et al. present an algorithm that finds such a schedule in polynomial time. However, this algorithm is not suitable for practical applications since the hidden constants are very large. There are also some local algorithms for this problem (algorithms in which each node must take the scheduling decisions for its packets without knowing the packets in the rest of the network) needing $O(C) + (\log^* |\mathcal{M}|)^{O(\log^* |\mathcal{M}|)} D + \mathit{poly}(\log |\mathcal{M}|)$ [26] and $O\left(C + D + \log^{1+\epsilon} |\mathcal{M}|\right)$ [24] steps with high probability. For the case that all paths are shortest paths, Meyer auf der Heide et al. [2] present a randomized online routing protocol which needs only $O(C + D + \log |\mathcal{M}|)$ steps with high probability. Busch, Magdon-Ismail, and Mavronicolas [4] present a bufferless routing algorithm whose length is bounded by $O\left((C + D) \cdot \log^3 (n + |\mathcal{M}|)\right)$ where n denotes the size of the network. Using the algorithm by Leighton et. al as a subroutine, Srinivasan and Teo [28] present an algorithm that solves the packet routing problem with variable paths with a constant approximation factor. This algorithm was recently improved by Koch et al. [18] for the more general message routing problem (where each message consists of several packets).

The packet routing problem is closely related to the multi-commodity flow over time problem [10,15,16]. In particular, Hall et al. [15] show that this problem is NP-hard, even in the very restricted case of series-parallel networks. We obtain the packet routing problem with variable paths if we additionally require unit edge capacities, unit transit times, and integral flow values. If there is only one start and one destination vertex then the packet routing problem is equivalent to the quickest flow problem. It can be solved optimally in polynomial time, e.g., using the Ford-Fulkerson algorithm for the maximum flow over time problem [11,12] together with a binary search framework. Using Megiddo's method of parametric search [23], Burkard, Dlaska, and Klinz [3] present a faster algorithm which solves the quickest flow problem in strongly polynomial time. Adler et al. [1] study the problem of scheduling as many packets as possible through a given network in a certain time frame. They give approximation algorithms and NP-hardness results.

For our algorithm on directed trees we need to find a path coloring for the paths of the packets. The path coloring problem is widely studied in the literature. For instance, Raghavan et al. [27] present a (3/2)-approximation algorithm for the path coloring problem on undirected trees. Erlebach et al. [9] give NP-hardness results and algorithms for the problem. In particular, they improve the algorithm by Raghavan et al. mentioned above and present a (4/3)-approximation. For coloring directed paths on bidirected trees (i.e., trees in which each edge represents two links, one in each direction) there are algorithms known which need at most $\frac{5}{3}L$ colors where L denotes the maximum load on a directed link [8]. This is tight since there are instances which actually need $\frac{5}{3}L$ colors [17]. Gargano et al. [14] investigate the problem of coloring all directed paths in a bidirected tree.

1.3 Our Contributions

We present three individual algorithms and several complexity results for the packet routing problem. If the underlying graph is a directed tree, we first show how to solve the path coloring problem optimally. This is based on ideas presented in [9,14]. Having computed such a coloring, we present a new technique which constructs a direct schedule whose length is bounded by $C + D - 1$. In comparison, the best known algorithm for computing direct schedules for packet routing on general trees guarantees a schedule of length $2C + D - 2$ [5]. (Note that the authors of [5] assume that the edges can be used in opposite directions at the same time.)

The new idea and method we employ is likely to be useful as a subroutine for packet routing on other topologies as well. Note that makespan $C + D - 1$ is a 2-approximation since C and D are both lower bounds on the optimum, but it guarantees a much better ratio if $C \ll D$ or $C \gg D$. Moreover, we show that $C + D - 1$ is the best ratio we can possibly guarantee in terms of C and D since there are instances which actually need this many steps. We show that even for the very simple case of directed trees the natural farthest-destination-first-algorithm (FDF) can yield arbitrarily large approximation factors. However, we show how to use it as a subroutine to obtain a 2-approximation algorithm for undirected trees. This guarantees a better performance ratio than the algorithm by Busch et al. [5] since $2C + D - 2$ can asymptotically as bad as a 3-approximation.

Then we present a very general condition which guarantees a direct schedule of a given time horizon T in directed graphs. Also, we present an algorithm which computes this schedule. This result is particularly substantial in the case where $T = D$ since then we can guarantee an optimal schedule. As an application, we show that if the paths of all packets are shortest paths and their lengths are pairwise different, we can compute an optimal direct schedule of length D. This improves [22] where it was shown that under this condition there exists a (not necessarily directed) schedule of this length. We would like to emphasize that in our understanding of directed graphs, each edge can be used only in the direction of the edge. For all our algorithms we show that the analysis of the respective approximation ratios is tight. The algorithms are presented in Section 2.

Then, in Section 3, we study the complexity of the packet routing problem: We show that it is NP-hard to approximate within a factor of $(6/5 - \epsilon)$, for any $\epsilon > 0$. This implies, in particular, that there is no polynomial time approximation scheme (PTAS), unless $P = NP$. We show that this still holds if we restrict the graph topology to directed trees or chain graphs with fixed paths. Then we investigate whether there can be an algorithm with an absolute error. We answer the question in the negative. We show that it is NP-hard to approximate the packet routing problem with fixed paths with an absolute error of k for any fixed $k \geq 0$. This holds even on planar graphs.

Due to space restrictions, some proofs are shortened or moved to the appendix. For full details we refer to our technical report [25].

2 Algorithms

In this section we study approximation algorithms for packet routing. For directed trees we show that the path coloring problem can be solved optimally in $O(n \log C)$ time, where n denotes the number of vertices in the graph and C the congestion of the packet routing instance. Based on this, we present an algorithm which computes a schedule whose length is bounded by $C + D - 1$ (D denotes the dilation). For undirected trees, we present a 2-approximation algorithm. It uses the farthest-destination-first-algorithm (FDF) as a subroutine. Even though the latter performs optimally on in- and out-trees [21], we show that on general directed trees it might compute arbitrarily bad schedules. Then we give a condition for the existence of a direct schedule with a given time horizon T and an algorithm which computes this schedule. In particular, if the lengths of the paths of the packets are pairwise different, all paths are shortest paths, and the graph is directed this yields an optimal direct schedule of length D. This improves [22] where for this setting the existence of a not necessarily direct schedule of that length was proven.

2.1 Approximation for Directed Trees

For directed trees we show how to construct a direct schedule of length at most $C + D - 1$ in polynomial time. The algorithm works as follows: First we find a coloring for the paths of the packets such that two paths that share an edge have different colors. We will show that the number of colors needed is exactly C. We assign to each packet the color of its path. Then we assign to each edge a time-dependent color. The idea behind this is that we transfer a packet P with color c_P along an edge $e = (u, v)$ only when e has the color c_P. We define the coloring such that for two consecutive edges $e = (u, v)$ and $e' = (v, w)$ it holds that at time t the edge e' has always the color that e had at time $t - 1$. This ensures that once a packet starts moving, it will never stop until it reaches its destination. In the sequel we describe the algorithm in detail.

Path Coloring. First we want to find a coloring for the paths such that two paths with the same color do not share an edge. Our algorithm works in two phases: in the first phase we consider each vertex v together with its adjacent vertices (this subgraph forms a star). For each of these subgraphs (together with the paths of the packets in this subgraph) we solve the path coloring problem optimally (here we use the fact that our tree is directed). Then we combine all these part-solutions and obtain a solution for the global path-coloring problem.

Phase one: Let v be a vertex and denote by T_v the subgraph induced by v and its adjacent vertices. We want to find a coloring for the paths which use edges in T_v. We reduce this problem to the edge-coloring problem on bipartite multigraphs (note that it is crucial that the edges in T_v are directed). This construction was already mentioned in [14].

Let U be the set of vertices which have outgoing edges to v, i.e., $U = \{u | (u, v) \in E\}$. Similarly, let W be the set of vertices having ingoing edges from v, i.e., $W = \{w | (v, w) \in E\}$. We construct an undirected graph B_v as follows: the set $U \cup W$ forms the set of vertices in B_v. For each path P that goes from a vertex $u \in U$ through v to a vertex $w \in W$ we introduce an edge $e_P := \{u, w\}$ in B_v. For all paths P that start in a vertex $u \in U$ and end in v we introduce a new vertex $w_P \in W$ and an edge $e_P := \{u, w_P\}$ in B_v. Similarly, for paths P that start in v and end in a vertex $w \in W$ we add a vertex v_P and introduce an edge $e_P := \{v_P, w\}$ in B_v. Thus, for the maximum degree $\Delta(B_v)$ of a node in B_v it holds that $\Delta(B_v) \leq C$. Also, it holds that two edges e_P and $e_{P'}$ share an end-vertex if and only if their corresponding paths P and P' share an edge in T_v. Thus, a valid edge-coloring for B_v implies a valid path coloring for T_v and vice versa. Moreover, from the construction it follows that B_v is bipartite. We compute a minimum edge coloring for B_v (e.g., see [6]). The number of colors needed equals the maximum node degree $\Delta(B_v)$ [6].

Phase two: Now we combine the found solutions for the graphs T_v one by one to obtain a global solution (a similar construction is described in [9, Lemma 2]). We start with an arbitrary vertex v and the path coloring of T_v. Now let v' be a vertex adjacent to v and consider the graph $T_{v'}$. We permute the colors of the paths in $T_{v'}$ such that the paths which use the edge (v, v') (or (v', v), respectively) have the same colors in T_v and $T_{v'}$. We iterate over the vertices by always adding a vertex that is adjacent to one of the vertices that have been considered already. Eventually, for each edge $e = (u, v)$ all paths that use e have the same color in T_u and T_v. Since T is a tree in each iteration we can find a valid permutation of the colors of the paths by using a simple greedy strategy. Since for each graph T_v we find path colorings with at most C colors, the resulting path coloring for T has C colors as well.

Time-Dependent Edge Coloring. Now we construct a time-dependent coloring $c : E \times \mathbb{N} \to \{1, 2, ..., C\}$ for the edges of T. It has the *consecutive property:* for two consecutive edges $e = (u, v)$ and $e' = (v, w)$ it holds that $c(e, i) = c(e', i + 1)$. Since our graph is a directed tree such a coloring can be found with a greedy method: Start with an arbitrary edge e and define its coloring $c(e, i) := i \bmod C$ for all $i \in \mathbb{N}$. Then inductively assign the colors to the remaining edges such that the consecutive property holds.

Routing Schedule. Finally, we describe the scheduling algorithm. First, we assign to each packet the color of its path. Now let M be a packet which is located on a vertex u at time t and which needs to use the edge $e = (u, v)$ next. Let c_P be the color of M. We move M along e in the first timestep t' with $t' \geq t$ and $c(e, t') = c_P$. Due to the consecutive property of the time-dependent edge coloring a packet is never delayed once it has left its start vertex. Denote by $DTREE(I)$ the resulting schedule for an instance I.

Theorem 1. *Let T be a directed tree and let $I = (T, \mathcal{M})$ be a packet routing instance. It holds that $|DTREE(I)| \leq C + D - 1$. A packet is never delayed once it has left its start vertex (direct routing). Moreover, $DTREE(I)$ can be computed in $O(n \cdot |\mathcal{M}| \cdot \log C)$.*

Proof. Since no two packets with the same color share an edge there can be at most one packet that uses an edge e at a time t. Each packet M waits in its origin vertex for at most $C - 1$ timesteps. Due to the consecutive property once it left its start vertex it moves to its destination without being delayed any further. Thus, the length of the overall makespan is bounded by $C - 1 + D$.

The edge coloring problem on bipartite multigraphs can be solved optimally in $O(m \log \Delta)$ where m denotes the number of edges in the graph and Δ the maximum node degree, see [6]. Thus, computing the optimal path coloring for one graph T_v can be done in $O(|\mathcal{M}| \cdot \log C)$ and for all graphs T_v in $O(n \cdot |\mathcal{M}| \cdot \log C)$.

Then we need to combine the colorings for the graphs T_v to a global path coloring. We say a path P *touches* a vertex v if P goes through v, starts in v or ends in v. We pick an arbitrary vertex v and color all paths which touch v in the colors that they have in T_v. After this initialization we iterate by taking vertices v' which are adjacent to already considered vertices. When we iterate we need to find a color permutation for $T_{v'}$ which is consistent with the coloring for T_v. This permutation is partly already defined by the color assignment for T_v. The remainder can be found in $O(C) \subseteq O(|\mathcal{M}|)$ steps. Since the order of the vertices can be obtained by a depth-first-search the second phase can be done in $O(n \cdot |\mathcal{M}|)$. This gives a total runtime of $O(n \cdot |\mathcal{M}| \cdot \log C)$. □

Note that the bound $C + D - 1$ is the best bound we can give in terms of C and D since there are packet routing instances which need this many steps. E.g., consider a path of length D with vertices $v_0, ..., v_D$ and C packets all with start vertex v_0 and destination vertex v_D.

2.2 Algorithm for Undirected Trees

It is not clear how the technique described in Section 2.1 could be applied to undirected trees. In particular, the path coloring problem on undirected trees is significantly harder than on directed trees. Also, it is not clear how the consecutive edge-coloring technique could be applied to undirected trees. However, we present a 2-approximation for the packet routing problem on undirected trees.

The algorithm works as follows: Let $T = (V, E)$ be a tree and let $I = (T, \mathcal{M})$ be a packet routing instance. Let v_r be an arbitrary vertex. We define v_r to be the root of the tree. We observe that the path of each packet can be split into two subparts: in the first part M_i moves towards v_r. In the second part M_i moves away from v_r. Let v_i be the vertex which divides these two parts. We split the routing problem into two subproblems: First we move each packet M_i from s_i to v_i. In the second part we move each packet M_i from v_i to t_i. We observe that the first part is a packet routing problem on an in-tree (a tree in which all vertices have an out-degree of at most one) and can therefore easily be solved optimally in polynomial time (FDF-algorithm, see [21]). Similarly, the second part is a packet routing instance on an out-tree (a tree in which all vertices have an in-degree of at most one) which can also be solved optimally in polynomial time (FDF-algorithm, see [21]). In the overall schedule for I, we run the optimal schedule for the first part. Then we run the optimal schedule for the second part. Denote by $TREE\,(I)$ the resulting schedule for the instance I.

Theorem 2. *For the schedule* $TREE\,(I)$ *it holds that* $|TREE(I)| \leq 2 \cdot |OPT(I)|$.

Proof. Since the length of an optimal schedule for each of the two subproblem forms a lower bound on the size of an optimal schedule for the original problem, we achieve an approximation ratio of two. □

It can be shown that the time needed to compute $TREE\,(I)$ is bounded by $O\left(n^2\right)$ where $n = \max\{|\mathcal{M}|, |V|\}$. For details see [25].

When implementing the algorithm one would not let a packet M_i wait in v_i until all other packets have finished the first part of the schedule. We would rather always move a packet when the next edge on its path is free (and prioritize the packets such that at each timestep each packet is at least as far on its path as in the original schedule described above). But even then there are instances which show that our analysis is asymptotically tight [25].

2.3 Directed Graphs and Shortest Paths

We give a condition which allows the existence of a direct schedule within a given time horizon T. As a corollary we obtain that if the lengths of the paths are pairwise different there is an optimal direct schedule of length D.

Let $G = (V, A)$ be a directed graph, let $T \geq 0$ be a time horizon and let $I = (G, \mathcal{M}, \mathcal{P})$ be a packet routing instance. We demand that for the lengths of the paths of the packets the following conditions hold:

- All paths are shortest paths.
- For each packet M_i denote by $\mathcal{M}_i$ the set of packets M_j such that P_j shares an edge with P_i and $D_j \geq D_i$. We demand that $|\mathcal{M}_i| \leq T - D_i + 1$.

For two packets M_i and M_j such that P_i and P_j share an edge we define a value $d\,(M_i, M_j)$. Let v_{ij} be the first vertex on P_i and P_j which is used by both paths. Denote by $d\,(s_i, v_{ij})$ and $d\,(s_j, v_{ij})$ the number of edges on P_i and P_j between s_i and v_{ij} and between s_j and v_{ij}, respectively. Then we define $d\,(M_i, M_j) := d\,(s_i, v_{ij}) - d\,(s_j, v_{ij})$. Note that $d\,(M_i, M_j) = -d\,(M_j, M_i)$.

Assuming that M_i is delayed for k timesteps in the beginning, M_i collides with M_j if and only if P_i and P_j share an edge and M_j is delayed for $d(M_i, M_j) + k$ steps (here we use the fact that G is a directed graph and all paths are shortest paths). We order the packets in decreasing order of the lengths of their paths. W.l.o.g. we assume that $M_0, ..., M_k$ is such an order. Now we iterate over the packets. In the i-th iteration, we consider the packet M_i. Denote by w_j with $0 \leq j \leq i-1$ the waiting time which was computed in a previous iteration for the packet M_j. (This implies that in our schedule the packet M_j waits for w_j steps and then moves to its destination without any further delay.) We say a packet M_j *blocks* a certain waiting time m for M_i if P_i and P_j share an edge and $m = d(M_i, M_j) + w_j$. Note that each packet whose path shares an edge with P_i blocks exactly one waiting time for P_i.

Let m be the smallest unblocked waiting time for M_i. We define $w_i := m$. In our schedule the packet M_i then waits for w_i steps and then moves to its destination without any further delay. We denote by D_i the length of P_i. We will prove in Theorem 3 that $D_i + w_i \leq T$ and thus M_i needs at most T steps to reach its destination. We denote by $SPATHS(I)$ the resulting schedule.

Theorem 3. *Let G be a directed graph, let $I = (G, \mathcal{M}, \mathcal{P})$ be a packet routing instance, and let $T \geq 0$ be a time horizon with the above conditions. Then for the schedule $SPATHS(I)$ it holds that $|SPATHS(I)| \leq T$. Moreover, $SPATHS(I)$ is a direct schedule.*

Proof. It remains to prove that $D_i + w_i \leq T$ for each packet M_i. Since we considered the packets in decreasing order of their path lengths, only packets in $\mathcal{M}_i$ can possibly block a certain waiting time for M_i. Since $|\mathcal{M}_i| \leq T - D_i + 1$ and $M_i \in \mathcal{M}_i$ we conclude that at most $T - D_i$ waiting times for M_i are blocked by packets M_j with $j < i$. This proves that $w_i \leq T - D_i$ which implies that $D_i + w_i \leq D_i + (T - D_i) = T$. □

Note that the bound for the length of $SPATH(I)$ is tight. E.g., let C and D be arbitrary positive integers and consider a packet routing instance as follows: Let the graph be a directed path with vertices $v_0, v_1, ..., v_D$ and consider C packets all with start vertex v_0 and destination vertex v_D. We define $T := C + D - 1$. Then the above conditions are satisfied (since for all packets M_i we have that $|\mathcal{M}_i| = |\mathcal{M}| = C = T - D_i + 1$) and the length of the optimal schedule is exactly T. Moreover, if we weaken our condition and require only that $|\mathcal{M}_i| \leq T - D_i + 2$ we cannot guarantee the existence of a schedule of length T anymore. E.g., take the above example with $C+1$ packets from v_0 to v_D and $T := C + D - 1$. Then each schedule needs at least $C + D > T$ steps.

We obtain the following two corollaries:

Corollary 1. *Let G be a directed graph, let $I = (G, \mathcal{M}, \mathcal{P})$ be a packet routing instance, and let $T \geq 0$ be a time horizon with the following conditions:*

- *All paths are shortest paths.*
- *Let $\mathcal{M}_i$ denote the set of packets whose path has at least $D - i$ edges. For each $i \geq 0$ we have that $|\mathcal{M}_i| \leq i + 1$.*

Then the schedule $SPATHS(I)$ is optimal with $|SPATHS(I)| = D$.

Corollary 2. *Let G be a directed graph and let $I = (G, \mathcal{M}, \mathcal{P})$ be a packet routing instance such that all paths are shortest paths and the lengths of all paths are pairwise different. Then the schedule $SPATHS(I)$ is optimal with $|SPATHS(I)| = D$.*

Compare that in [22] it was shown that under the condition of Corollary 2 there is a (not necessarily direct) schedule whose length is bounded by D. We proved that in this case there is even a direct schedule with this makespan.

2.4 Farthest-Destination-First-Algorithm on Directed Trees

The farthest-destination-first-algorithm prioritizes the packets according to the length of their remaining path. That is, packets whose remaining path is longer have a higher priority than packets whose remaining path is shorter. Ties are broken arbitrarily. It was shown by Leung [21] that on in-trees and on out-trees the FDF-algorithm works optimally. However, we show that on general directed trees the FDF-algorithm can perform arbitrarily bad in terms of the achieved performance ratio. For an instance I of the packet routing problem, denote by $FDF(I)$ a longest schedule that the FDF-algorithm could possibly compute.

Theorem 4. *For every $k \geq 1$ there is a a directed tree T_k and a packet routing instance $I_k = (T_k, \mathcal{M}_k)$ such that*

$$|FDF\,(I_k)| \geq k \cdot |OPT\,(I_k)|$$

Due to space constraints we refer to our technical report [25] for the construction.

3 Complexity Results

In this section we study the complexity of the packet routing problem. Due to space constraints we refer to our technical report [25] for detailed descriptions of the reductions.

Theorem 5. *For all $\epsilon > 0$, there is no $(6/5 - \epsilon)$-approximation algorithm for the packet routing problem with fixed paths, unless $P = NP$.*

Proof (sketch). In the reduction we employ a technique which was used in [29] for showing that the general acyclic job shop problem is NP-hard to approximate within an approximation factor of $5/4 - \epsilon$. We reduce from 3-BOUNDED-3-SAT [13]. In this variant of 3-SAT in the given formula each variable occurs at most three times (positive and negative). □

We can modify the reduction to show that the packet routing problem is also NP-hard to approximate on planar graphs and even on directed trees. Note here that in the latter case it does not make a difference whether the paths of the packets are given in advance or not. As a corollary we obtain the same result for directed chain graphs with given paths.

All these reductions rely on creating a gap of one time unit between yes- and no-instances of 3-BOUNDED-3-SAT. This raises the question whether it is NP-hard to approximate the packet routing problem with an *absolute* error in polynomial time.

Theorem 6. *For all $k > 0$, there is no approximation algorithm for the packet routing problem with fixed paths which guarantees an absolute error of at most k, unless $P = NP$.*

Acknowledgements

We would like to thank Joachim Reichel, Ronald Koch and Daniel Plümpe for helpful discussions.

References

1. Adler, M., Khanna, S., Rajaraman, R., Rosén, A.: Time-constrained scheduling of weighted packets on trees and meshes. Algorithmica 36, 123–152 (2003)
2. Meyer auf der Heide, F., Vöcking, B.: Shortest-path routing in arbitrary networks. Journal of Algorithms 31 (1999)
3. Burkard, R.E., Dlaska, K., Klinz, B.: The quickest flow problem. ZOR — Methods and Models of Operations Research 37, 31–58 (1993)
4. Busch, C., Magdon-Ismail, M., Mavronicolas, M.: Universal bufferless packet switching. SIAM Journal on Computing 37, 1139–1162 (2007)
5. Busch, C., Magdon-Ismail, M., Mavronicolas, M., Spirakis, P.: Direct routing: Algorithms and complexity. Algorithmica 45, 45–68 (2006)
6. Cole, R., Ost, K., Schirra, S.: Edge-coloring bipartite multigraphs in $O(E \log D)$ time. Combinatorica 21, 5–12 (2001)
7. di Ianni, M.: Efficient delay routing. Theoretical Computer Science 196, 131–151 (1998)
8. Erlebach, T., Jansen, K.: An optimal greedy algorithm for wavelength allocation in directed tree networks. In: Proceedings of the DIMACS Workshop on Network Design: Connectivity and Facilities Location, vol. 40, pp. 117–129. AMS (1997)
9. Erlebach, T., Jansen, K.: The complexity of path coloring and call scheduling. Theoretical Computer Science 255, 33–50 (2001)
10. Fleischer, L., Skutell, M.: Quickest flows over time. SIAM Journal on Computing 36, 1600–1630 (2007)
11. Ford, L.R., Fulkerson, D.R.: Constructing maximal dynamic flows from static flows. Operations Research 6, 419–433 (1958)
12. Ford, L.R., Fulkerson, D.R.: Flows in Networks. Princeton University Press, Princeton (1962)
13. Garey, M., Johnson, D.: Computers and Intractability: A Guide to the theory of NP-completeness. Freeman, New York (1979)
14. Gargano, L., Hell, P., Perennes, S.: Colouring paths in directed symmetric trees with applications to WDM routing. In: Degano, P., Gorrieri, R., Marchetti-Spaccamela, A. (eds.) ICALP 1997. LNCS, vol. 1256, pp. 505–515. Springer, Heidelberg (1997)

15. Hall, A., Hippler, S., Skutella, M.: Multicommodity flows over time: Efficient algorithms and complexity. In: Baeten, J.C.M., Lenstra, J.K., Parrow, J., Woeginger, G.J. (eds.) ICALP 2003. LNCS, vol. 2719, pp. 397–409. Springer, Heidelberg (2003)
16. Hoppe, B., Tardos, E.: The quickest transshipment problem. Mathematics of Operations Research 25, 36–62 (2000)
17. Jansen, K.: Approximation results for wavelength routing in directed binary trees. In: Proceedings of the Workshop on Optics and Computer Science (1997)
18. Koch, R., Peis, B., Skutella, M., Wiese, A.: Real-time message routing and scheduling. In: Dinur, I., Jansen, K., Naor, J., Rolim, J. (eds.) Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques. LNCS, vol. 5687, pp. 217–230. Springer, Heidelberg (2009)
19. Leighton, F.T., Maggs, B.M., Rao, S.B.: Packet routing and job-scheduling in $O(\mathit{congestion} + \mathit{dilation})$ steps. Combinatorica 14, 167–186 (1994)
20. Leighton, F.T., Maggs, B.M., Richa, A.W.: Fast algorithms for finding $O(\mathit{congestion} + \mathit{dilation})$ packet routing schedules. Combinatorica 19, 375–401 (1999)
21. Leung, J.Y.-T.: Handbook of Scheduling: Algorithms, Models and Performance Analysis (2004)
22. Mansour, Y., Patt-Shamir, B.: Greedy packet scheduling on shortest paths. Journal of Algorithms 14 (1993)
23. Megiddo, N.: Combinatorial optimization with rational objective functions. Mathematics of Operations Research 4, 414–424 (1979)
24. Ostrovsky, R., Rabani, Y.: Universal $O(\mathit{congestion} + \mathit{dilation} + \log^{1+\epsilon} N)$ local control packet switching algorithms. In: Proceedings of the 29th annual ACM Symposium on Theory of Computing, pp. 644–653 (1997)
25. Peis, B., Skutella, M., Wiese, A.: Packet routing: Complexity and algorithms. Technical Report 003-2009, Technische Universität Berlin (February 2009)
26. Rabani, Y., Tardos, É.: Distributed packet switching in arbitrary networks. In: Proceedings of the 28th annual ACM Symposium on Theory of Computing, pp. 366–375. ACM, New York (1996)
27. Raghavan, P., Upfal, E.: Efficient routing in all-optical networks. In: Proceedings of the 26th annual ACM Symposium on Theory of Computing, pp. 134–143. ACM, New York (1994)
28. Srinivasan, A., Teo, C.-P.: A constant-factor approximation algorithm for packet routing and balancing local vs. global criteria. SIAM Journal on Computing 30 (2001)
29. Williamson, D.P., Hall, L.A., Hoogeveen, J.A., Hurkens, C.A.J., Lenstra, J.K., Sevast'janov, S.V., Shmoys, D.B.: Short shop schedules. Operations Research 45, 288–294 (1997)