Distributed quickest flow in sparse layered networks

Andrzej Czygrinow[1], Michał Hańckowiak[2]

[1]School of Mathematical and Statistical Sciences, Arizona State University, Tempe, AZ, 85287-1804, USA
[2]Faculty of Mathematics and Computer Science, Adam Mickiewicz University, Poznań, Poland

Abstract
We propose a fast distributed algorithm for a variant of the quickest flow problem in a special class of layered sparse graphs. The algorithm works in the CONGEST model and finds a $1/4$-approximation of an optimal solution which uses paths of length at most five.

Keywords
Distributed algorithms, CONGEST model, Quickest Flow

1. Introduction

Data transmission problems are ubiquitous in network communication research. Different variants and formalizations, tailored for specific applications have been proposed. In this note, we consider the following data transmission problem. Given a (simple) graph G, two vertices v and w in G, and $b \in \mathbb{Z}^+$ units of data at v, transfer all data from v to w as quickly as possible. We assume that the transfer of data proceeds in rounds, and require that intermediate nodes do not accumulate data as they are sent from v to w beyond what is needed to facilitate the communication since this will lead to a delay.

In addition, we assume that edges are bi-directional and only one unit of data, which corresponds to $O(\log n)$ of bits, can be transferred in each direction of a single edge at any given time. Consequently, upon receiving data, a vertex must send all of it out in the subsequent time step. To facilitate transmission, the vertices of G find (in a distributed way) a set $\mathcal{Q}$ of v, w-paths that can be used to send data from v to w. The paths from $\mathcal{Q}$ do not need to be internally disjoint and will be used repeatedly to send data from v to w.

Related Work Data transmission problems are fundamental questions in computer science that arise in many important applications. Traditionally, questions about data transfer are associated with the maximum flow problem [4] that has been extensively studied for different computational models. In the CONGEST model, advanced distributed algorithms exist for finding a maximum flow [5] or minimum-cost flow [9]. These are impressive results, but the running time of these algorithms is substantially larger than ours in the case of sparse graphs. More importantly, the traditional max-flow problem is not very well suited when the goal is to send a fixed amount of data from one vertex to another. Although useful modifications of the max-flow problem have been studied, their distributed complexity has not been explored. One such variant is the *dynamic max-flow problem*. In this problem, given T, the goal is to find a maximum source-sink flow that can be completed in time T. A closely related variant is the *quickest flow problem*. Here, given b, the aim is to find a flow that sends b units of data from a source to a sink in the shortest amount of time. In this paper, we consider the distributed version of the latter problem.

In the case of the dynamic max-flow problem, sequential algorithms can be obtained by modifying the underlying network and Burkad et al. [2] gave a polynomial time algorithm for the quickest flow problem based on parametric search. Lin and Jaillet [6] gave an algorithm that finds a quickest flow in time $O(nm \log (n^2/m) \log(nC))$ where n, m are the order and the size of the network and $C \in \mathbb{Z}^+$

ICTCS 2026: Italian Conference on Theoretical Computer Science, September 7–9, 2026, Udine, Italy
aczygri@asu.edu (A. Czygrinow); mhanckow@amu.edu.pl (M. Hańckowiak)

is the maximum capacity. Subsequently, Saho and Shigeno [8] further refined this idea and gave a $O(nm^2 \log^2 n)$-time algorithm.

Although much is now known about the sequential complexity of the quickest flow problem, its distributed complexity has not been studied. In this note, we study the special case of the quickest flow problem with capacities one for special types of layered sparse networks in the CONGEST model.

Our Contibution In its full generality, the problem seems quite challenging and we consider the following special case where graph G has a simple but non-trivial structure. Specifically, $V(G)$ consists of layers $\{v_{out}\}, V_1, V_2, \{v_{in}\}$, graphs $G[\{v_{out}\}, V_1]$ and $G[\{v_{in}\}, V_2]$ are complete bipartite graphs, and the graph $G[V_1, V_2]$ has a bounded arboricity. We find v_{in}, v_{out}-paths of length at most five to facilitate the transfer of b units of data from v_{in} to v_{out} (See Figure 1). To meet the aforementioned requirements, we consider the following directed graph that we identify with G. For $\{v_{out}, x\}$ we orient from x to v_{out} (and use xv_{out} for the arc), for $\{v_{in}, x\}$ we orient from v_{in} to x (and use $v_{in}x$), and for $\{x, y\} \in E(G[V_1, V_2])$ we add both arcs xy and yx. Recalling that a directed edge of a graph should not be used by two different paths at the same time slot, the goal becomes to find directed v_{in}, v_{out}-paths of length at most five such that for every directed edge e, there is at most one path in P that contains e. Note that for an (undirected) edge $\{x, y\}$ of G, we can have two different paths, one containing xy, and the other yx. Consequently, if the orientation is ignored on the directed paths, then the paths do not need to be edge-disjoint. (See Figure 1.) We prove the following theorem.

Theorem 1. *Let $a, b, n \in \mathbb{Z}^+$. There is a distributed algorithm that in a layered graph G with $G = (v_{out}, V_1, V_2, v_{in}, E)$ of order n such that $G[V_1, V_2]$ has arboricity at most a, finds a set $\mathcal{P}$ of directed edge-disjoint v_{in}, v_{out}-paths of length at most five such that the number of rounds, $\mathbf{time}(\mathcal{P})$, needed to send b units of data from v_{in} to v_{out} using paths from $\mathcal{P}$ satisfies*

$$\mathbf{time}(\mathcal{P}) \leq 4 \cdot \mathbf{time}(\mathcal{P}^*),$$

where $\mathbf{time}(\mathcal{P}^*)$ is the smallest number of rounds attained using edge-disjoint paths of length at most five. The algorithm runs in $O(a^2 + \log n)$ number of rounds in the CONGEST model of computations. Moreover, if an orientation D of E such that $\Delta^+(D) = O(a)$ is given, the number of rounds is $O(a^2)$.

Clearly, the main appeal of the algorithm from Theorem 1 comes in the case when G is sparse, that is when $a = O(1)$. We do not know how to extend the approach from Theorem 1 to paths of larger lengths and if this extension leads to a significant improvement. (See Section 4.)

Although the considered layered graphs might seem quite particular, it is not difficult to imagine natural scenarios that result in these types of networks. For example, suppose we have a set of clients, two sets of intermediate vertices, proxy 1 and proxy 2, and a set of server vertices. For a client v_{out} and a server v_{in}, v_{out} wants to communicate with v_{in} by a "broadband channel" $(v_{out}, V_1, V_2, v_{in})$, which should guarantee fast communication. Consequently, the channel should consist of edge-disjoint v_{in}, v_{out}-paths for client v_{out}, with edge-disjoint channels for different clients. Assuming there is exactly one server and a constant number of active clients, there is a simple way of finding the channels. Indeed, every proxy 2 vertex w constructs a table with columns v_c, q_c for every client v_c that specifies the number of edges between w and proxy 1 which are on the v_c,v_{out}-paths (as the second edges) that go via w. Then every proxy 2 vertex w sends its table to the server, which can decide which clients should w send data to. Since the server knows the size of all of the possible broadband channels, it can choose a desired solution.

Going back to the classical max-flow problem, assume the capacity of each edge is one. Then a flow can be viewed as an orientation of a subset $E' \subseteq E(G)$ such that for every $v \in V(G) \setminus \{v_{out}, v_{in}\}$, the out-degree of v in E' is equal to the in-degree of v in E'. The source vertex v_{in} sends $d^+_{E'}(v_{in})$ units of data by sending a single unit through each edge in E' that starts in v_{in}. The internal vertices of the directed v_{in}, v_{out}-paths simply propagate the data until it reaches v_{out}. Significantly, however, the lengths of the paths and as a result the overall time needed to transfer data cannot be anticipated by the nodes. Consequently, the simple strategy of dividing up the data to compensate for the lengths of

the paths used to transfer so that longer paths are allotted smaller amounts of data is not a feasible approach.
The rest of the note is structured as follows. In the next section, we introduce some useful terminology and prove a few initial observations. Section 3 contains the main algorithm and its analysis. In Section 4 we conclude with a few open questions.

2. Preliminaries

We will use $\mathbb{Z}^+$ and $\mathbb{N}$ to denote the sets of positive and non-negative integers. Let $G = (V, E)$ be a graph and let $f : V \to \mathbb{N}$. An f-matching in G is a set of edges $F \subseteq E$ such that for every $v \in V$, $d_F(v) \leq f(v)$. An f-matching F is called *maximum* if there is no f-matching F' such that $|F'| > |F|$, and is called *maximal* if for every $e \in E \setminus F$, $F \cup \{e\}$ is not an f-matching in G. In the case $f(v) = 1$ for every $v \in V$, an f-matching is called a matching. For a set $X \subseteq V(G)$ we say that an f-matching M *saturates* U if for every $v \in U$, $d_M(u) = f(u)$. In the case $U = \{u\}$, we say that u is M-saturated.

For a graph $G = (V, E)$, the arboricity of G, $arb(G)$, is the least integer k such that E can be partitioned into k forests. We have the following fundamental fact about $arb(G)$.

Lemma 2 (Nash-Williams [7])**.** *Let G be a graph and $k \in \mathbb{Z}^+$. Then $arb(G) \leq k$ if and only if for every nonempty subset $U \subseteq V$ the graph $H = (U, F)$ induced by U satisfies $|F| \leq k(|U| - 1)$.*

The following fact about f-matchings from [3] can be easily proved by appealing to Hall's theorem. We include a proof to make the paper self-contained.

Fact 3 ([3])**.** *If a bipartite graph $H = (A, B, E)$ has arboricity $a \in \mathbb{Z}^+$ and $f : A \cup B \to \mathbb{Z}^+$ is such that for every $v \in A$, $f(v) \leq d_H(v)/(2a)$, then there is an f-matching that saturates A.*

Proof. By the assumption, for every $v \in A$, $d_H(v) \geq 2a \cdot f(v)$. Let $H' = (A', B, E)$ be obtained from H as follows. For every $v \in A$, put an independent set I_v of size $f(v)$ so that every edge of H incident to v is incident to exactly one vertex from I_v and each vertex from I_v has degree at least $2a$. Since H has arboricity a and we do not add any new edges to H, graph H' has arboricity at most a. Let $X \subseteq A'$ and let $Y := N_{H'}(X)$. By Lemma 2, we have $2a|X| \leq |E_{H'}(X, Y)| \leq a(|X| + |Y|)$. Thus $|X| \leq |Y|$ and hence, by Hall's theorem, there is a matching M in H' which saturates A'. Since $|I_v| \leq f(v)$ for every $v \in A$, gluing together vertices in I_v for every v results in an f-matching in H that saturates A. □

We continue with some additional notation and terminology. Let v_{out}, v_{in} be two vertices and let V_1, V_2 be two disjoint sets of vertices that do not contain v_{out}, v_{in}. Let G_{12} denote a bipartite graph with bipartition V_1, V_2 and let G denote the graph obtained from G_{12} be adding all edges $\{v, v_{out}\}$ for $v \in V_1$ and all edges $\{v, v_{in}\}$ for $v \in V_2$. As indicated before, we will identify G_{12} and G with directed graphs obtained as follows. For every $\{u, v\} \in E(G_{12})$ we add both arcs uv and vu. For $v \in V_1$, we add the arc vv_{out}, and for $v \in V_2$ we add $v_{in}v$. With this set up, directed paths that we consider will always be in the directed graphs obtained from G.

We say that two directed paths are *edge-disjoint* if they contain different directed edges. For example, if $\{u, v\}$ is in G_{12} there can be a path that contains uv and a path with vu. Let $\mathcal{P}$ be a set of directed edge-disjoint v_{in}, v_{out}-paths in G and let $q_i := q_i(\mathcal{P})$ denote the number of paths in $\mathcal{P}$ of length $2i + 1$ for $i \in \{1, 2\}$.

Definition 4. *For $b \in \mathbb{Z}^+$, let $\mathbf{time}(q_1, q_2, b) := \mathbf{time}_{\mathcal{P}}(q_1, q_2, b)$ denote the least number of rounds needed to send b units of data from v_{in} to v_{out} using the paths from $\mathcal{P}$.*

We will use the following convention which is consistent with the definition in [1] . If u sends one unit of data to v, then it takes two rounds for this data to be available to v because only at the beginning of the second round v can use the data. Using this convention, we have the following observation.

Lemma 5. *Let $b, y \in \mathbb{Z}^+$.*

(a) If P is a directed v_{in}, v_{out}-path of length y, then the least number of rounds needed to transfer b units of data from v_{in} to v_{out} using P is equal to $b + y$.

(b) If $\mathcal{P}$ is a set of directed edge-disjoint v_{in}, v_{out}-paths each of length y, then the least number of rounds needed to transfer b units of data from v_{in} to v_{out} using $\mathcal{P}$ is $\lceil b/k \rceil + y$, where $k = |\mathcal{P}|$.

Proof. For the upper bound in (a), vertices can propagate data one unit at a time along path P starting at v_{in}. For the lower bound, note that the number of rounds needed to send b units of data from v_{in} to v_{out} is at least as much as sending one unit of data along P with the delay of $b - 1$ rounds. Sending one requires $1 + y$ rounds for the unit to reach v_{out}; y rounds to send along P, so that data is available to v_{out} at the beginning of round $1 + y$.

For (b), divide b units into k disjoint groups, each of size at most $\lceil b/k \rceil$, and transfer data in parallel, using different paths from $\mathcal{P}$ for different groups. For the lower bound, by the pigeonhole principle, there is a path $P \in \mathcal{P}$ that is used to send $\lceil b/k \rceil$ data. By part (a), this takes $\lceil b/k \rceil + y$ rounds. □

The next lemma gives a tight bound for the number of rounds in the case only paths of length three and five are used.

Lemma 6. *Let $b \in \mathbb{Z}^+$. Suppose $\mathcal{P}$ is a set of directed edge-disjoint v_{in}, v_{out}-paths in G of length three or five such that $q_1 \geq 1$. Then*

$$\mathbf{time}(q_1, q_2, b) = \begin{cases} \left\lceil \frac{b}{q_1+q_2} + \frac{3q_1+5q_2}{q_1+q_2} \right\rceil & \text{if } b \geq 2q_1 \\ \left\lceil \frac{b}{q_1} \right\rceil + 3 & \text{if } b \leq 2q_1. \end{cases}$$

Proof. First note that if $q_2 = 0$, then by Lemma 5 (b), the least number of rounds is $\lceil b/q_1 \rceil + 3$ for any $b \in \mathbb{Z}^+$. Now assume $q_2 > 0$. If $b \leq 2q_1$, then $\lceil b/q_1 \rceil + 3 \leq 5$. By Lemma 5 (b) using paths of length three from $\mathcal{P}$ completes the transfer in $\lceil b/q_1 \rceil$ rounds and by Lemma 5 (a) using a path of length five results in a larger number of rounds.

Suppose $b \geq 2q_1$. Let $\beta \in [0, 1]$ and suppose that $\lceil \beta b \rceil$ units of b are sent using paths of length three. Then $\lfloor (1 - \beta)b \rfloor$ are transmitted using paths of length five, using Lemma 5 (b). Let $T(\beta)$ denote the least number of rounds when $\lceil \beta b \rceil$ units of b are sent using paths of length three. Then, by Lemma 5 (b),

$$T(\beta) = \max(\lceil \lceil \beta b \rceil / q_1 \rceil + 3, \lceil \lfloor (1 - \beta)b \rfloor / q_2 \rceil + 5).$$

Since we can take any $\beta \in [0, 1]$ to distribute the data between the sets of paths of lengths three and five, the minimum number of rounds T satisfies

$$T = \min_{\beta \in [0,1]} \max(\lceil \lceil \beta b \rceil / q_1 \rceil + 3, \lceil \lfloor (1 - \beta)b \rfloor / q_2 \rceil + 5).$$

We will first show that $T \geq \left\lceil \frac{b}{q_1+q_2} + \frac{3q_1+5q_2}{q_1+q_2} \right\rceil$. To that end, note that for every $\beta \in [0, 1]$,

$$T(\beta) \geq \max(\beta b/q_1 + 3, \lceil (b - \lceil \beta b \rceil)/q_2 \rceil + 5).$$

We have $(b - \lceil \beta b \rceil)/q_2 > (b - \beta b - 1)/q_2$, and so $\lceil (b - \lceil \beta b \rceil)/q_2 \rceil + 5 \geq (b - \beta b)/q_2 + 5$. Thus

$$T(\beta) \geq \max(\beta b/q_1 + 3, (1 - \beta)b)/q_2 + 5). \tag{1}$$

Clearly, β_0 that satisfies $\beta_0 b/q_1 + 3 = (1 - \beta_0)b/q_2 + 5$ gives the least value $T(\beta_0)$ in (1). It follows that $\beta_0 = \frac{bq_1+2q_1q_2}{bq_1+bq_2}$, and since $b \geq 2q_1$, $\beta_0 \in [0, 1]$. In addition, $T(\beta_0) = \frac{b}{q_1+q_2} + \frac{3q_1+5q_2}{q_1+q_2}$. Since $T \in \mathbb{Z}$, $T \geq \lceil T(\beta_0) \rceil$.

In a similar way one can argue that $T \leq \left\lceil \frac{b}{q_1+q_2} + \frac{3q_1+5q_2}{q_1+q_2} \right\rceil$. Indeed, let $F(\beta) = \max(\beta b/q_1 + 4, (1 - \beta)b/q_2 + 6)$. For $m \in \mathbb{Z}$, $\lceil m/q_1 \rceil \leq (m + q_3 - 1)/q_1$, and so $\lceil \lceil \beta b \rceil / q_1 \rceil < ((\beta b + 1) + q_1 - 1)/q_1 = \beta b/q_1 + 1$. Trivially, $\lceil \lfloor (1 - \beta)b \rfloor / q_2 \rceil + 5 < (1 - \beta)b/q_2 + 6$. Therefore, for every $\beta \in [0, 1]$,

$$T(\beta) < F(\beta). \tag{2}$$

Using the same β_0 as above, we have $\beta_0 b/q_1 + 4 = (1-\beta_0)b/q_2 + 6$, and so

$$F(\beta_0) = \frac{b}{q_1+q_2} + \frac{3q_1+5q_2}{q_1+q_2} + 1. \tag{3}$$

By (2), $T(\beta_0) < \lceil F(\beta_0) \rceil$, and since both are integers, by (3), $T \leq T(\beta_0) \leq \left\lceil \frac{b}{q_1+q_2} + \frac{3q_1+5q_2}{q_1+q_2} \right\rceil$. □

In the next section, we will show how to find a family of directed v_{in}, v_{out}-paths in G that can be used in Lemma 6.

3. Algorithm

In this section, we will give the main algorithm. In what follows, we use standard notation for graphs and digraphs. In particular, if $D = (V, E)$ is a digraph and $v \in V$, then $N_D^+(v)$ is the set of out-neighbors of v, $N_D^-(v)$ is the set of in-neighbors of v, $d_D^+(v) := |N_D^+(v)|$, and $d_D^-(v) := |N_D^-(v)|$. We let $\Delta^+(D) := \max_v d_D^+(v)$. We will only consider simple digraphs, that is digraphs such that for any two vertices u, v there is at most one edge from u to v. Note that it is possible to have an edge from u to v and from v to u. We say that a (simple) digraph D is an *orientation* if for any two vertices u, v at most one of the edges uv and vu is in D. The algorithm will work in the layered graph G obtained from a bipartite graph G_{12} as described in the previous section. Let $a \in \mathbb{Z}^+$. We will assume that the edges of G_{12} have orientation D such that for every $v \in V_1 \cup V_2$, $d_D^+(v) \leq a$.

Definition 7. *Given a maximal matching M in G_{12} let W_1 be the set of vertices in V_1 that are M-saturated and let W_2 be the set of vertices in V_2 that are not M-saturated. Let $H := G_{12}[W_1, W_2]$.*

Note that, since M is maximal, there are no edges in G_{12} between $V_1 \setminus W_1$ and W_2. See Figure 1.

The main algorithm will rely on the notion of feasible paths. Although it will only use paths of length two, we define it for general lengths.

Definition 8. *A set $\mathcal{P}$ of directed paths in G_{12} is called feasible if it satisfies the following conditions.*

1. *Each path from $\mathcal{P}$ starts in W_1 and ends in $V_1 \setminus W_1$.*
2. *The paths are edge-disjoint.*
3. *For every $v \in V_1 \setminus W_1$ there is at most one path from $\mathcal{P}$ that contains v.*
4. *For every vertex $v \in W_1$ there are at most $d_H(v)$ paths that contain v.*

Note that paths from $\mathcal{P}$ have necessarily even lengths.

We will first give a high-level description of the main algorithm, FindPaths, with a detailed description and analysis of individual steps provided in the lemmas that follow.

The algorithm finds directed edge-disjoint v_{in}, v_{out}-paths that are used to transmit data from v_{in} to v_{out}. In finding these paths one of the vertices v_{in}, v_{out} gathers information about certain subgraphs of an input graph G. As a result, this vertex has perfect information about these subgraphs and can locally find desired structures in the subgraphs (in our case, certain f-matchings), that are further used to find paths.

Definition 9. *Let G be a connected graph, let F be a subgraph of G, and let $v \in V(G)$. We say that v gathers F when v obtains information about all vertices and edges of F.*

The idea of the procedure is as follows. Recall that we are given a layered graph G and G_{12} denotes the bipartite graph $G[V_1, V_2]$. In addition, we assume that there is an orientation D of G_{12} such that for every $v \in V_1 \cup V_2$, $d_D^+(v) \leq a$. Initially, we find a maximal matching M in G_{12}. Note that finding M efficiently is not completely trivial and we show how this can be done in Lemma 11. The orientation D is used to make sure that this can be completed in $O(a)$ rounds in the CONGEST model. Given the layered structure of G, the edges in M determine directed edge-disjoint v_{in}, v_{out}-paths of length three

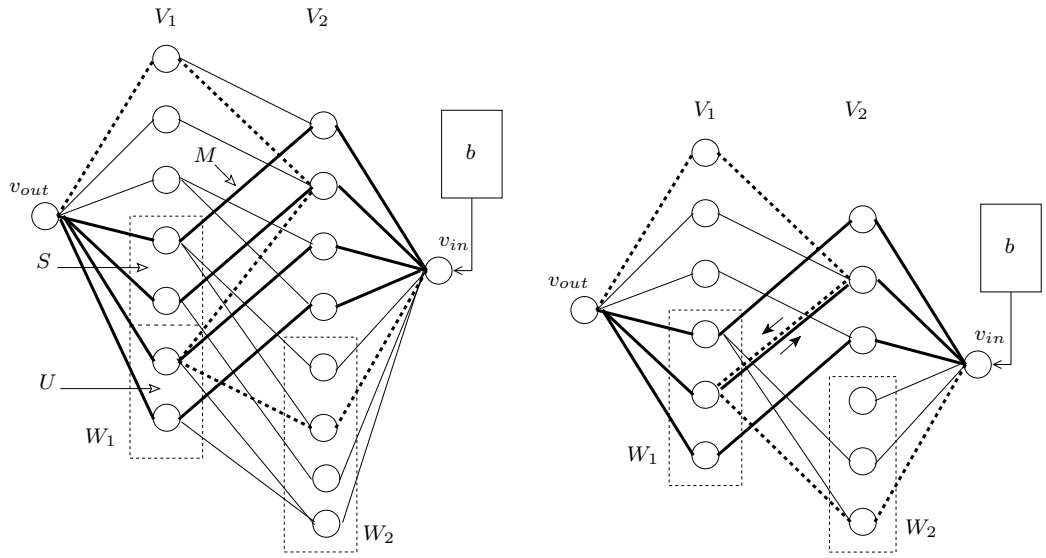

Figure 1: A maximal matching M in $G[V_1, V_2]$ gives sets W_1, W_2. In addition, two v_{in}, v_{out}- paths, one of length 3, the other of length 5 can share an edge when viewed as un-directed paths.

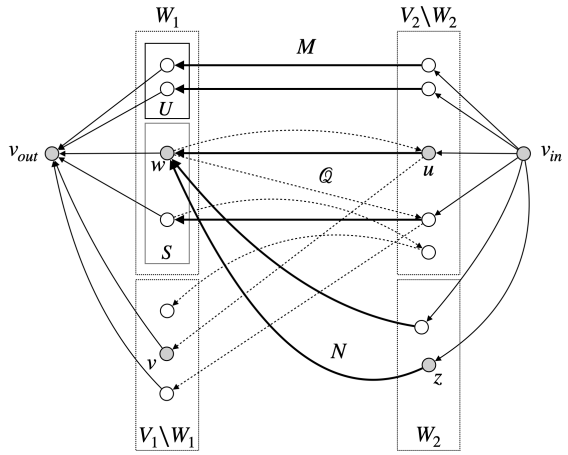

Figure 2: Procedure FindPaths: Set $\mathcal{Q}$ of feasible paths is extended using the f-matching N. Paths $v_{in}uwv_{out} \in \mathcal{P}_1$ and $v_{in}zwuvv_{out} \in \mathcal{P}_3$ are edge-disjoint directed v_{in}, v_{out}-paths.

in G. If $b < 2|M|$, then in view of Lemma 6 these paths are enough to guarantee the optimal number of rounds.

If $b \geq 2|M|$ however, then to assure optimality, we have to find additional v_{in}, v_{out}-paths of length five. Again, the main issue is to make sure that this can be accomplished efficiently in the CONGEST model. Ideally, we would like v_{out} to gather the whole graph, find an optimal set, and then let other vertices know about the paths. However, doing so efficiently does not seem to be possible. At the same time, we can show that v_{out} can gather efficiently $G_{12}[S \cup (V_1 \setminus W_1), V_2]$ for some subset $S \subseteq W_1$. (See Figure 1 and Figure 2.) It will then examine all sets of feasible paths of length two from S to $V_1 \setminus W_1$ and choose one which is the best option for extending the paths to v_{in}, v_{out}-paths of length five. This extending will be done as follows. If $P = wuv$ is a feasible path with $v \in V_1 \setminus W_1$, $w \in S$, then a vertex z from W_2 can be added to extend P to $zwuv$, which now gives $v_{in}zwuvv_{out}$. Note that there can be at most one feasible path ending at v but potentially more then one ending at w. To make sure we can extend as many paths as possible, the sets of feasible paths will be used to define values of a function f that is further used to consider an f-matching in $G_{12}[W_1, W_2]$. A useful f-matching is computed and

subsequently used to extend all paths ending at the vertices from S.

We will now give a high-level description of the algorithm. As indicated above, making sure that finding matchings, gathering and other computations can be performed efficiently requires additional details which are provided in Lemma 11.

Procedure FindPaths

Input: A layered graph $G = (v_{out}, V_1, V_2, v_{in}, E)$, $a, b \in \mathbb{Z}^+$, and orientation D of G_{12} such that $d^+_D(v) \leq a$ for every $v \in V_1 \cup V_2$.

Output: A set of directed edge-disjoint v_{in}, v_{out}-paths of lengths three and five.

1. (a) Find a maximal matching M in G_{12}. Let $k_1 := |M|$.
   (b) The edges of M determine directed v_{in}, v_{out}-paths of length three. Let $\mathcal{P}_1$ denote the set of these paths.
   (c) If $b < 2k_1$, then $mode := 1$, $\mathcal{P}_2 := \emptyset$ and goto 6.
2. Vertex v_{out} gathers the graph $G_{12}[V_1, V_2 \setminus W_2]$.
3. Recall that $H = G_{12}[W_1, W_2]$. Let $S \subseteq W_1$ be the set of vertices $v \in W_1$ that have at least $d_H(v)/(2a)$ neighbors in $V_2 \setminus W_2$, and let $U := W_1 \setminus S$.
4. Vertex v_{out} gathers the graph $G_{12}[S, W_2]$. (Note that at this point v_{out} knows the whole graph $G_{12}[S \cup (V_1 \setminus W_1), V_2]$.)
5. (a) Vertex v_{out} examines all feasible sets of paths of length two in G_{12}.
   (b) For every feasible set of paths $\mathcal{Q}$ from 5(a), let $f_{\mathcal{Q}} : W_1 \cup W_2 \to \mathbb{N}$ be defined as follows: $f_{\mathcal{Q}}(v)$ equals the number of paths in $\mathcal{Q}$ that start at $v \in W_1$ and $f_{\mathcal{Q}}(u) = 1$ for $u \in W_2$. Moreover, for $\mathcal{Q}$ and $f_{\mathcal{Q}}$, v_{out} computes:
      (i) a maximum $f_{\mathcal{Q}}$-matching $N_{\mathcal{Q}}$ in $G_{12}[S, W_2]$;
      (ii) $x_{\mathcal{Q}} := |N_{\mathcal{Q}}| + \sum_{v \in U} f_{\mathcal{Q}}(v)$.
   (c) Vertex v_{out} chooses a set $\mathcal{Q}^*$ such that $x_{\mathcal{Q}^*}$ is maximum and finds a maximum $f_{\mathcal{Q}^*}$-matching $N := N_{\mathcal{Q}^*}$ in $G_{12}[S, W_2]$.
   (d) Vertex v_{in} gathers graph $G_{12}[U, W_2]$ and values $f_{\mathcal{Q}^*}(u)$ for $u \in U$ (analogously to step 2), and computes a maximum $f_{\mathcal{Q}^*}$-matching N' in $G_{12}[U, W_2]$.
   (e) For every vertex $v \in W_2$ there can be at most two edges from $N \cup N'$ that end in v. Keep only one of them discarding the other arbitrarily.
   (f) First use edges from $N \cup N'$ that have not been deleted in 5(e) to extend paths of length two to paths of length three. (Note that N' saturates U and N is maximum in $G_{12}[S, W_2]$.) Then extend each of these paths to a v_{in}, v_{out}-path of length five. Denote a resulting set of paths by $\mathcal{P}_2$. Let $mode := 2$.
6. Build routing tables on vertices from $V_1 \cup V_2 \cup \{v_{in}\}$ by Table. Note that v_{in}, v_{out}-paths described by these tables correspond to paths denoted by $\mathcal{P}_1$ and $\mathcal{P}_2$.

Note that FindPaths gives a high-level description of the algorithm with more details of how to accomplish the desired running time given in the forthcoming lemmas.

To facilitate data propagation, each vertex v builds a routing table that allows it to move incoming data. Such a routing table consists of an in- and out-column and each row of the table contains a pair (i, j) where i and j are the numbers of two edges incident to v. If the data comes through edge i, then it must be further sent via edge j.

Procedure Table

Input: $mode \in \{1, 2\}$; matching M, $\mathcal{Q}^*$, and f-matchings $N \subseteq S \times W_2$; $N' \subseteq U \times W_2$.

Output: A routing table on every vertex with exception of v_{out}.

1. For every $u_1u_2 \in M \subseteq V_1 \times V_2$, v_{out} creates a list $u_1u_2v_{in}$. If $mode = 1$ goto 5.
2. For every $u_1u_2u_3u_4$ where $u_1u_2u_3 \in \mathcal{Q}^*$ and u_3u_4 is its extension in $N \subseteq S \times W_2$, v_{out} creates a list $u_1u_2u_3u_4v_{in}$.
3. For every $u_1u_2u_3$ where $u_1u_2u_3 \in \mathcal{Q}^*$ and $u_3 \in U$, v_{out} creates a list $u_1u_2u_3$.
4. For every $u_1u_2 \in N' \subseteq U \times W_2$, v_{in} creates a list $u_2u_1 \in N'$.

5. Now, vertices v_{out} and v_{in} send the lists they have to the first vertex from each list.
6. If the received list is non-empty, the receiver deletes a first element from it and sends it to the first vertex from the truncated list.
7. If a vertex v receives a list through the edge of number i and sends it further through the edge of number j, then the rows for both (i, j) and (j, i) are added to the routing table in v.
8. If $mode = 2$ then: A vertex $v \in U$ that receives empty lists coming from v_{out} and v_{in}, pairs them arbitrarily and for a pair $\{i, j\}$, v adds both (i, j) and (j, i) to the rows of its table and sends an empty list via the just established route to v_{in}.
9. Finally, also v_{in} adds $(j, null)$ to its table for each empty list that came through the edge number j.

The following observation follows from the description of procedure Table.

Lemma 10. *Procedure* Table *runs in $O(1)$ rounds in the CONGEST model.*

We will now provide more details of how to efficiently implement procedure FindPaths. Note that although the orientation D specified as the input to FindPaths is not explicitly used in the high-level description of the procedure, it is certainly needed in a finer description given in the next lemma of how to implement the algorithm to run in $O(a^2)$ rounds.

Lemma 11. *Let $a \in \mathbb{Z}^+$. Given a layered graph $G = (v_{out}, V_1, V_2, v_{in}, E)$ and an orientation D of G_{12} that satisfies $\Delta^+(D) \leq a$, procedure* FindPaths *can be implemented to run in $O(a^2)$ rounds in the CONGEST model.*

Proof. Let G_{12}^+ be the subgraph of G_{12} that contains edges $\{v_1, v_2\}$ such that $v_1 \in V_1$ and $v_1v_2 \in E(D)$, and G_{12}^- be the subgraph that contains edges $\{v_1, v_2\}$ such that $v_1 \in V_1$ and $v_2v_1 \in E(D)$. Then $E(G_{12}^+), E(G_{12}^-)$ partition the edge set of G_{12}.
Step 1: We will first show that it is possible to complete step one of the algorithm in $O(a)$ rounds. The graph G_{12}^+ contains edges $\{v, u\}$ such that $v \in V_1$, $u \in N_{G_{12}^+}(v)$ and for every vertex $v \in V_1$, $|N_{G_{12}^+}(v)| \leq a$. Similarly for G_{12}^-. Therefore v_{out} can gather the whole graph G_{12}^+ in $O(a)$ rounds and v_{in} can gather G_{12}^- in $O(a)$ rounds in the CONGEST model. We can find a maximal matching M in G_{12} as follows. First, vertex v_{out} finds a maximal matching M_1 in G_{12}^+. Second, vertices in $V(M_1) \cap (V_1 \cup V_2)$ are removed and vertex v_{in} finds a maximal matching in what is left of G_{12}^-. All matched vertices are marked, and both v_{out}, v_{in} can find the size of the resulting matching M.
Step 2: We will now analyze step two and show that v_{out} can gather in $O(a)$ rounds graph $G_{12}[V_1, V_2 \setminus W_2]$. Recall that v_{out} can gather graph G_{12}^+ in $O(a)$ rounds. We will now argue that v_{out} can also gather the edges of G_{12}^- that are incident to $V_2 \setminus W_2$. By the definition of W_2, vertices in $V_2 \setminus W_2$ are M-saturated by the matching M from step one. Let $v_1 \in V_1, v_2 \in V_2 \setminus W_2$ be such that $\{v_1, v_2\} \in M$. Since $d_D^+(v_2) \leq a$, there are at most a vertices u such that $\{u, v_2\} \in E(G_{12}^-)$. Therefore, using the edges of M, v_{out} can gather in $O(a)$ rounds the whole graph $G_{12}^-[V_1, V_2 \setminus W]$.
Step 3: This step can be implemented in $O(1)$ rounds.
Step 4: Recall that $H = G_{12}[W_1, W_2]$. We will show that v_{out} can gather $G_{12}[S, W_2]$ in $O(a^2)$ rounds. Let $v \in S$. First note that by the definition of S,

$$|N_{G_{12}}(v) \cap (V_2 \setminus W_2)| \geq d_H(v)/(2a). \tag{4}$$

We proceed based on the value of $|N_{G_{12}}(v) \cap (V_2 \setminus W_2)|$ and consider two cases.
If $|N_{G_{12}}(v) \cap (V_2 \setminus W_2)| \leq 2a$, then from (4)

$$d_H(v) \leq 2a|N_{G_{12}}(v) \cap (V_2 \setminus W_2)| \leq 4a^2.$$

In this case, information about all edges of H that are incident to v can be gathered by v_{out} in $O(a^2)$ rounds through the edge $\{v, v_{out}\}$.

Now suppose $|N_{G_{12}}(v) \cap (V_2 \setminus W_2)| > 2a$. Then information about all edges $\{v, w\}$ such that $w \in W_2$ can be evenly distributed among all vertices in $N_D^-(v) \cap (V_2 \setminus W_2)$. Indeed, since $|N_D^+(v)| \leq a$, we have

$$|N_D^-(v) \cap (V_2 \setminus W_2)| \geq |N_{G_{12}}(v) \cap (V_2 \setminus W_2)| - a. \tag{5}$$

Therefore, from (4) and (5),

$$\frac{d_H(v)}{|N_D^-(v) \cap (V_2 \setminus W_2)|} \leq \frac{d_H(v)}{|N_{G_{12}}(v) \cap (V_2 \setminus W_2)| - a} \leq \frac{2a|N_{G_{12}}(v) \cap (V_2 \setminus W_2)|}{|N_{G_{12}}(v) \cap (V_2 \setminus W_2)| - a} < 4a,$$

where the last inequality follows from $|N_{G_{12}}(v) \cap (V_2 \setminus W_2)| > 2a$. Therefore vertex v can distribute to each vertex from $N_D^-(v) \cap (V_2 \setminus W_2)$ information about $O(a)$ edges from H that are incident to v. For a vertex $u \in V_2 \setminus W_2$ edges in D from u to S are used to distribute this information. Consequently, summing over all vertices $v \in S$, each vertex $u \in V_2 \setminus W_2$ accumulates information about $O(|N_D^+(u)| \cdot a) = O(a^2)$ edges from $G_{12}[S, W_2]$. By (5) every vertex v has $N_D^-(v) \cap (V_2 \setminus W_2) \neq \emptyset$, and so information about $G_{12}[S, W_2]$ can be transmitted to v_{out} using the edges of M in $O(a^2)$ rounds.

Step 5: Since v_{out} knows the whole graph $G_{12}[S \cup (V_1 \setminus W_1), V_2]$, it can do the computations from steps 5(a)-5(c) in one round. Vertices v_{in} and v_{out} are symmetric, and so, analogously to step 2, v_{in} can gather information about the graph $G_{12}[V_1, W_2]$. Thus, since $U \subseteq V_1$, it can do the computations in $G_{12}[U, W_2]$ from step 5(d). Steps 5(e)-5(f) take a constant number of rounds. □

Before we continue with additional analysis, we want to make a comment about potential extensions to include paths of lengths larger than five. Note that in our approach, it is important that vertices $u \in U$ have a bounded number of feasible paths that end at u. This is because these reach u from vertices in $V_2 \setminus W_2$ (using maximality of M). This however is no longer the case when considering longer paths.

We now continue with some observations about paths from $\mathcal{P}_1$ and $\mathcal{P}_2$.

Lemma 12. *The directed paths in $\mathcal{P}_1 \cup \mathcal{P}_2$ are edge-disjoint.*

Proof. The paths in $\mathcal{P}_1$ are clearly edge-disjoint. We will now analyze the paths from $\mathcal{P}_2$. Let $P = wuz$ be a directed path in $\mathcal{Q}^*$ from step 5(c). Since $\mathcal{Q}^*$ is feasible, $w \in W_1$, $z \in V_1 \setminus W_1$, and by the maximality of M, $u \in V_2 \setminus W_2$. Clearly the edge $\{z, u\} \notin M$ and even if the edge $\{u, w\} \in M$, uw, not wu, is in a path from $\mathcal{P}_1$. The directed path P will be extended to vP using vw from $N \cup N'$. Since $v \in W_2$, vertices z and v are not M-saturated. In particular, the paths from $\mathcal{P}_1$ do not use the arcs $zv_{out}, v_{in}v$. Therefore, the paths from $\mathcal{P}_1$ are edge-disjoint from the paths in $\mathcal{P}_2$.

Now let $P = wuz, P' = w'u'z'$ be two directed paths from $\mathcal{Q}^*$ that are extended in step 5(f). Since $\mathcal{Q}^*$ is feasible, $z \neq z'$ and P, P' are edge-disjoint. Suppose P and P' are extended to vP and $v'P'$. Then $vw, v'w' \in N \cup N'$. Consequently, $v \neq v'$, since every vertex in W_2 can be incident to at most two edges from $N \cup N'$, and if there are two, then one is discarded in 5(e). Therefore, the directed paths $vP, v'P'$ are edge-disjoint, and $z \neq z'$, $v \neq v'$. Consequently, $zv_{out} \neq z'v_{out}$ and $v_{in}v \neq v_{in}v'$. □

Corollary 13. *Given a layered graph $G = (v_{out}, V_1, V_2, v_{in}, E)$ procedure* FindPaths *returns a set of directed edge-disjoint v_{in}, v_{out}-paths in G of length three or five.*

We have the following definition.

Definition 14. *We say that a directed path Q of length three blocks a directed path P of length five if P and Q are not edge-disjoint.*

In the next two facts, we analyze how directed paths can block additional paths.

Fact 15. *Let $\mathcal{P}$ be a set of directed edge-disjoint v_{in}, v_{out}-paths of length five. A directed v_{in}, v_{out}-path of length three can block at most two paths from $\mathcal{P}$.*

Proof. Let $P = v_{in}u_2u_1v_{out}$ be a directed v_{in}, v_{out}-path of length three and let $e_{in} := v_{in}u_2$, $e_{21} = u_2u_1$, and $e_{out} = u_1v_{out}$. Since paths in $\mathcal{P}$ are edge-disjoint, for every directed edge e in P there is at most one path in $\mathcal{P}$ that contains e. Suppose $Q \in \mathcal{P}$ contains e_{21}. Then e_{21} cannot be the first edge in Q nor the fifth (as these are incident to v_{in} and v_{out} respectively). In addition, e_{21} cannot be the third edge of Q, since $e_{21} = u_2u_1$ where $u_2 \in V_2, u_1 \in V_1$ but for the third edge uu' in Q, $u \in V_1$ an $u' \in V_2$. If e_{21} is the second edge of Q, then e_{in} is the first edge in Q, and so $e_{in} \in E(Q) \cap E(P)$, and if e_{21} is the fourth, then $e_{out} \in E(Q) \cap E(P)$. Therefore, all paths in $\mathcal{P}$ that are blocked by P contain e_{in} or e_{out}. □

Fact 16. *Let $\mathcal{P}_1$ be the set of directed v_{in}, v_{out}-paths in G determined by M from step 1(b) of* FindPaths. *If $P = v_{in}u_4u_3u_2u_1v_{out}$ is a directed v_{in}, v_{out}-path in G that is not blocked by any of the paths from $\mathcal{P}_1$, then $u_1 \in V_1 \setminus W_1$, $u_2 \in V_2 \setminus W_2$, $u_3 \in W_1$, and $u_4 \in W_2$.*

Proof. We have $u_1 \in V_1 \setminus W_1$ as otherwise P is blocked by the path in $\mathcal{P}_1$ that contains the edge u_1v_{out}. Similarly, $u_4 \in W_2$. Since $\{u_1, u_2\}, \{u_3, u_4\}$ are in $E(G)$, by the maximality of M, $u_2 \in V \setminus W_2$ and $u_3 \in W_1$. □

We will now prove the main lemma in which we show that the set of paths obtained by FindPaths gives a 4-approximation of an optimal solution to the transfer of data problem that can be accomplished using paths of length at most five.

Lemma 17. *Let $\mathcal{P}_1$ and $\mathcal{P}_2$ be the sets returned by* FindPaths. *Let $k_i := |\mathcal{P}_i|$ and let k_i^* denote the number of paths of length $2i+1$ in an optimal solution to the problem of finding directed edge-disjoint v_{in}, v_{out}-paths of lengths three and five in graph G. If $k_1 \in \mathbb{Z}^+$ and $k_2 \in \mathbb{N}$, then for every $b \in \mathbb{Z}^+$,*

$$\mathbf{time}(k_1, k_2, b) < 4 \cdot \mathbf{time}(k_1^*, k_2^*, b).$$

Proof. When $b < 2k_1$ then $k_2 = 0$ (step 1(c)) and so, as in the proof of Lemma 6, $k_2^* = 0$. Thus, by Lemma 6, for $q_1 \in \{k_1, k_1^*\}$ and $q_2 = k_2 = k_2^* = 0$ we have $\mathbf{time}(q_1, q_2, b) = \left\lceil \frac{b}{q_1} \right\rceil + 3 = \left\lceil \frac{b}{q_1+q_2} + \frac{3q_1+5q_2}{q_1+q_2} \right\rceil$.

Note that if $k_1^* + k_2^* \leq 4(k_1 + k_2)$ then $\left\lceil \frac{b}{k_1^*+k_2^*} \right\rceil + 1 \geq \frac{b}{k_1^*+k_2^*} + 1 > \frac{1}{4}\left(\frac{b}{k_1+k_2} + 1\right) > \frac{1}{4}\left\lceil \frac{b}{k_1+k_2} \right\rceil$, and

$$\frac{\left\lceil \frac{b}{k_1^*+k_2^*} + \frac{3k_1^*+5k_2^*}{k_1^*+k_2^*} \right\rceil}{\left\lceil \frac{b}{k_1+k_2} + \frac{3k_1+5k_2}{k_1+k_2} \right\rceil} \geq \frac{\left\lceil \frac{b}{k_1^*+k_2^*} \right\rceil + 3}{\left\lceil \frac{b}{k_1+k_2} \right\rceil + 5} > \frac{1}{4}.$$

We will now show that $k_1^* + k_2^* \leq 4(k_1 + k_2)$. Let m^* denote the size of a maximum matching in G_{12}. We have $k_1^* \leq m^*$ and so

$$k_1 \geq m^*/2 \geq k_1^*/2 \tag{6}$$

because the matching M found in step 1(a) is maximal. In addition, we have the following claim.

Claim 18. $k_2 \geq \frac{1}{2}\max(k_2^* - 2k_1, 0)$.

Suppose the claim holds for the time being. If $k_2^* \leq 2k_1$ then by (6), $\frac{k_1^*+k_2^*}{k_1+k_2} \leq \frac{k_1^*+2k_1}{k_1} \leq 4$. If $k_2^* > 2k_1$ then by Claim 18

$$\frac{k_1^* + k_2^*}{k_1 + k_2} \leq \frac{k_1^* + k_2^*}{k_1 + \frac{1}{2}(k_2^* - 2k_1)} = 2\frac{k_1^*}{k_2^*} + 2.$$

We consider two subcases: If $\frac{k_1^*}{k_2^*} < 1$ then $\frac{k_1^*+k_2^*}{k_1+k_2} \leq 4$, and if $\frac{k_1^*}{k_2^*} \geq 1$ then again by (6), $\frac{k_1^*+k_2^*}{k_1+k_2} \leq \frac{2k_1^*}{k_1} \leq 4$. □

To finish the proof of Lemma 17 we verify Claim 18.

Proof of Claim 18. Let $\mathcal{P}_2^*$ be a set of directed edge-disjoint v_{in}, v_{out}-paths of length five that satisfies $|\mathcal{P}_2^*| = k_2^*$. By Fact 15, every path from $\mathcal{P}_1$ blocks at most two paths from $\mathcal{P}_2^*$. Let $\mathcal{P}$ be the set of paths in $\mathcal{P}_2^*$ that are not blocked by paths from $\mathcal{P}_1$. Then

$$|\mathcal{P}| \geq \max(k_2^* - 2k_1, 0). \tag{7}$$

In addition, by Fact 16, every path $P = v_{in}u_4u_3u_2u_1v_{out}$ from $\mathcal{P}$ satisfies $u_1 \in V_1 \setminus W_1, u_2 \in V_2 \setminus W_2, u_3 \in W_1, u_4 \in W_2$. Since paths in $\mathcal{P}$ are edge disjoint, for every $w \in V_1 \setminus W$ there is at most one path $P \in \mathcal{P}$ with $u_1 = w$. Consider the set $\mathcal{R}$ of truncated paths u_3Pu_1 of length two for each $P \in \mathcal{P}$. For every $w \in V_1 \setminus W_1$ there is at most one path in $\mathcal{R}$ that ends at w and there are at most $d_H(v)$ paths that end at v for $v \in W_1$, as paths in $\mathcal{P}$ are edge-disjoint and $H = G_{12}[W_1, W_2]$. Thus $\mathcal{R}$ is a feasible sets of paths of length two. In step 5(a), v_{out} examines all feasible sets of paths of length two, and so it must encounter $\mathcal{R}$.

For every vertex $v \in W_1$, $f_{\mathcal{R}}(v)$ is the number of paths from $\mathcal{R}$ that start at v. As a result, if N^* is a maximum $f_{\mathcal{R}}$-matching in H, then

$$|N^*| \geq |\mathcal{P}|. \tag{8}$$

In step 5(c), v_{out} chooses $\mathcal{Q}^*$ and an $f_{\mathcal{Q}^*}$-matching N in $G_{12}[S, W_2]$ such that $x_{\mathcal{Q}^*} = |N| + \sum_{v \in U} f_{\mathcal{Q}^*}(v)$ is maximum. In 5(d), v_{in} chooses a maximum $f_{\mathcal{Q}^*}$-matching N' in $G_{12}[U, W_2]$. For $v \in U$, $f_{\mathcal{Q}^*}(v) \leq |N_{G_{12}}(v) \cap (V_2 \setminus W_2)| < d_H(v)/(2a)$, because the paths in $\mathcal{Q}^*$ that start at v are edge-disjoint.

Thus, by Fact 3,

$$|N'| = \sum_{v \in U} f_{\mathcal{Q}^*}(v). \tag{9}$$

Let $N^*_S \subseteq N^*$ be the set of edges in N^* that have one of the endpoints in S and $N^*_U \subseteq N^*$ the set of edges with one of the endpoints in U. We have

$$|N^*| = |N^*_S| + |N^*_U| \leq |N^*_S| + \sum_{u \in U} f_{\mathcal{R}}(u). \tag{10}$$

Since N is an $f_{\mathcal{Q}^*}$-matching that maximizes $|N| + \sum_{u \in U} f_{\mathcal{Q}^*}(u)$, by (8), (9), and (10),

$$|N| + |N'| = |N| + \sum_{u \in U} f_{\mathcal{Q}^*}(u) \geq |N^*_S| + |N^*_U| = |N^*| \geq |\mathcal{P}|. \tag{11}$$

Consequently, by (7) and (11),

$$|N| + |N'| \geq \max(k_2^* - 2k_1, 0). \tag{12}$$

In step 5(e) the paths of length two are extended using edges of $N \cup N'$, but two edges, one from N, and the other from N' can end at the same vertex from W_2, and the algorithm keeps one edge from two. Consequently, in step 5(e), the algorithm has a set of paths of size at least $(|N| + |N'|)/2$. After extending the paths to v_{in}, v_{out}-paths of length five the algorithm returns the set of paths $\mathcal{P}_2$. Therefore, by (12),

$$k_2 = |\mathcal{P}_2| \geq \frac{1}{2} \max(k_2^* - 2k_1, 0).$$

□

Proof of Theorem 1. Note that an orientation D of $G = (v_{out}, V_1, V_2, v_{in}, E)$ such that $\Delta^+(D) = O(a)$ can be obtained in $O(\log n)$ rounds as follow. Each vertex v of degree at most $O(a)$ orients the edges from v. The edges are deleted and the process is iterated. Then use FindPaths to obtain $\mathcal{P}$. By Lemma 17, $\mathbf{time}(\mathcal{P}) \leq 4 \cdot \mathbf{time}(\mathcal{P}^*)$. By Lemma 11, the set $\mathcal{P}$ is found in $O(a^2)$ rounds in the CONGEST model. □

4. Conclusions

In this paper, we studied a distributed version of the quickest flow problem. Although the algorithm is very fast it is limited to graphs that are both sparse and layered (with the number of layers equal to two). Extending this regime to larger classes of graphs is maybe the most interesting question related to this line of research. In addition, it would be interesting to generalize our approach to include v_{in}, v_{out}-paths of length $2i + 1$ for $i > 2$.

Declaration on Generative AI

The authors have not employed any Generative AI tools.

References

[1] A. D. Bhandari, T. N. Dhamala, On quickest flow problem using improved binary search algorithm, IJISET - International Journal of Innovative Science, Engineering & Technology, Vol. 7 Issue 12, (2020) ISSN (Online), 2348.

[2] R.E. Burkard, K. Dlaska, B. Klinz, The quickest flow problem, ZOR Methods Models Oper Res, 37, (1993), 21–58.

[3] A. Czygrinow, M. Hańćkowiak, A. Rumiński, M. Witkowski, Distributed approximation for f-matching, Theoretical Computer Science, 1014, (2024), 114760.

[4] L. R. Ford, D. R. Fulkerson, Flows in networks, Princeton University Press, Princeton, NJ, 1962.

[5] M. Ghaffari, A. Karrenbauer, F. Kuhn, Ch. Lenzen, B. Patt-Shamir, Near-Optimal Distributed Maximum Flow, (2015), https://arxiv.org/abs/1508.04747.

[6] M. Lin, P. Jaillet, On the quickest flow problem in dynamic networks- A parametric min-cost flow approach, Proc. 26th Ann. ACM-SIAM Symp. on Discrete Algorithms, San Diego, CA, (2015), 1343–1356.

[7] C. St. J. A.Nash-Williams, Decomposition of finite graphs into forests. J. Lond. Math. Soc. 39 (1964), 12.

[8] M. Saho, M. Shigeno, Cancel-and-Tighten Algorithm for Quickest Flow Problems, Networks, 69, 2, (2017), 179–188.

[9] Tijn de Vos, Minimum Cost Flow in the CONGEST Model, (2023), https://arxiv.org/abs/2304.01600.