Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Paz Carmi, Thomas Erlebach, Yoshio Okamoto

Greedy edge-disjoint paths in complete graphs

TIK-Report
Nr. 155, February 2003

Institut für Technische Informatik und Kommunikationsnetze
Computer Engineering and Networks Laboratory

Paz Carmi, Thomas Erlebach, Yoshio Okamoto
Greedy edge-disjoint paths in complete graphs
February 2003
Version 1
TIK-Report Nr. 155

Computer Engineering and Networks Laboratory,
Swiss Federal Institute of Technology (ETH) Zurich

Institut für Technische Informatik und Kommunikationsnetze,
Eidgenössische Technische Hochschule Zürich

Gloriastrasse 35, ETH Zentrum, CH-8092 Zürich, Switzerland

Greedy edge-disjoint paths in complete graphs*

Paz Carmi† Thomas Erlebach‡ Yoshio Okamoto§

February 10, 2003

Abstract

The maximum edge-disjoint paths problem (MEDP) is one of the most classical NP-hard problems. We study the approximation ratio of a simple and practical approximation algorithm, the shortest-path-first greedy algorithm (SGA), for MEDP in complete graphs. Previously, it was known that this ratio is at most 54. Adapting results by Kolman and Scheideler [Proceedings of SODA, 2002, pp. 184–193], we show that SGA achieves approximation ratio $8F + 1$ for MEDP in undirected graphs with flow number F and, therefore, has approximation ratio at most 9 in complete graphs. Furthermore, we construct a family of instances that shows that SGA cannot be better than a 3-approximation algorithm. Our upper and lower bounds hold also for the bounded-length greedy algorithm, a simple on-line algorithm for MEDP.

Keywords: Approximation algorithm, Greedy algorithm, Maximum edge-disjoint paths problem, Shortening lemma

1 Introduction

The maximum edge-disjoint paths problem (MEDP) is one of the most classical NP-hard problems [4]. Some real-world problems concerning efficient transmission of messages over networks are reduced to MEDP.

Let us define MEDP. An instance of MEDP is given by a pair $(G, \mathcal{R})$ of an undirected graph $G = (V, E)$ and a multiset $\mathcal{R}$ of unordered pairs of vertices of V. Each element of $\mathcal{R}$ is called a *request*. We consider connecting many requests by paths such that they have no common edge. These paths are called *edge-disjoint paths*. A feasible solution to the instance $(G, \mathcal{R})$ of MEDP is given by a pair $(\mathcal{A}, \mathcal{P})$ of a subset $\mathcal{A}$ of $\mathcal{R}$ and a set $\mathcal{P}$ of edge-disjoint paths in G such that each request in $\mathcal{A}$ is connected by a path in $\mathcal{P}$. The task is to maximize the size of $\mathcal{A}$, which is denoted by $|\mathcal{A}|$. Note that $|\mathcal{A}| = |\mathcal{P}|$ in a feasible solution $(\mathcal{A}, \mathcal{P})$. A request is called *accepted* if it belongs to $\mathcal{A}$. Without loss of generality, we make the assumption that for each request there must be a

*This work was supported by the Berlin-Zürich Joint Graduate Program "Combinatorics, Geometry, and Computation" (CGC), financed by ETH Zürich and the German Science Foundation (DFG).

†Department of Computer Science, Ben-Gurion University of the Negev, Israel

‡Computer Engineering and Networks Laboratory, Department of Information Technology and Electrical Engineering, ETH Zürich, Switzerland

§Institute of Theoretical Computer Science, Department of Computer Science, ETH Zürich, Switzerland

path in G connecting it. (If there exists a request such that no path in G connects it, we simply remove the request from $\mathcal{R}$.)

Since it is known that MEDP is NP-hard even if input graphs are restricted to complete graphs [3], we are interested in finding approximation algorithms for MEDP. An algorithm for MEDP is called a *ρ-approximation algorithm* if it runs in polynomial time and always outputs a feasible solution $(\mathcal{A}, \mathcal{P})$ for any instance $(G, \mathcal{R})$ which satisfies $\mathrm{Opt} \leq \rho|\mathcal{A}|$, where Opt is the number of the accepted requests in an optimal solution to $(G, \mathcal{R})$. We say that ρ is an *approximation ratio*. We refer to [7] for an introduction and more background information about edge-disjoint paths problems and to [1] and the references given there for recent results about the approximability of MEDP in general undirected and directed graphs.

Table 1: Results about the approximation ratio for MEDP in complete graphs.

approximation ratio	algorithm	contributors [reference]
27	Tripartition	Erlebach and Vukadinović [3]
54	SGA	Erlebach and Vukadinović [3]
17	BGA ($L = 4$), SGA	Kolman and Scheideler [9]
9	BGA ($L = 4$), SGA	this paper

In this paper, we are mainly interested in MEDP in complete graphs. Table 1 shows the known and new results about the approximation ratio for MEDP in complete graphs. The first approximation algorithm with a constant approximation ratio was given by Erlebach and Vukadinović [3]. In their algorithm, we first divide the vertex set V into three parts V_1, V_2, V_3, and also divide the requests $\mathcal{R}$ into three parts $\mathcal{R}_{12}, \mathcal{R}_{23}, \mathcal{R}_{31}$ such that $\mathcal{R}_{ij}$ consists of the requests either whose both ends lie in V_i or whose one end lies in V_i and the other lies in V_j. Then, we solve the problem in which the requests are restricted to $\mathcal{R}_{ij}$ by a certain procedure independently and take the best solution. We call this algorithm the *tripartition algorithm*, but we do not precisely describe this algorithm here. It is important to say that they showed that this algorithm is a 27-approximation algorithm.

A simple algorithm for MEDP is a greedy algorithm. There are some variants for MEDP. The first greedy algorithm that we will see is called the *bounded-length greedy algorithm* (BGA). This algorithm is shown in Figure 1. Here, $\mathrm{length}(\pi_i)$ in Step 2-2 represents the length of the path π_i, i.e., the number of edges in π_i. BGA was first introduced by Kleinberg [7]. Note that BGA can process the requests in arbitrary order and is therefore an on-line algorithm, i.e., it can process each request without knowledge about future requests.

Kolman and Scheideler [9] invented a new parameter F for a network, called the flow number, and they showed that BGA with parameter $L = 4F$ is a $(16F + 1)$-approximation algorithm for the unsplittable flow problem (a generalization of MEDP). Indeed, they showed the "shortening lemma" and used it nicely to give their bound. Later we will show that the flow number of a complete graph is one. Thus their result implies a 17-approximation algorithm.

In this paper, we adapt their analysis to MEDP and prove using the shortening lemma that BGA with parameter $L = 4F$ actually is a $(8F+1)$-approximation algorithm for undirected graphs with flow number F. This shows that BGA with $L = 4$ is a 9-approximation algorithm for complete graphs.

Another variant of greedy algorithms is the *shortest-path-first greedy algorithm* (SGA), shown

Algorithm:	Bounded-length greedy algorithm (BGA)
Input:	an undirected graph $G = (V, E)$,
	a multiset $\mathcal{R} = \{\{s_1, t_1\}, \ldots, \{s_k, t_k\}\}$ of requests
	and a parameter L;
Output:	$\mathcal{A} \subseteq \mathcal{R}$ and a set $\mathcal{P}$ of edge-disjoint paths in G
	such that each request in $\mathcal{A}$ is connected by a path in $\mathcal{P}$;
Step 1:	(Initialize step)
	$\mathcal{A} \leftarrow \emptyset$; $\mathcal{P} \leftarrow \emptyset$; $i \leftarrow 1$;
Step 2:	(Main loop)
	while $i \leq k$
Step 2-1:	find a shortest path π_i connecting s_i and t_i in G;
Step 2-2:	if length$(\pi_i) \leq L$ then
Step 2-2-1:	$\mathcal{A} \leftarrow \mathcal{A} \cup \{\{s_i, t_i\}\}$ and $\mathcal{P} \leftarrow \mathcal{P} \cup \{\pi_i\}$;
Step 2-2-2:	remove all the edges of π_i from G;
Step 2-3:	$i \leftarrow i + 1$;
	end of while;
Step 3:	(Output)
	return $\mathcal{A}$ and $\mathcal{P}$.

Figure 1: The bounded-length greedy algorithm for MEDP.

in Figure 2. In this algorithm, if there are more than one candidates which can be selected at Step 2-1, then we choose one of them arbitrarily. We consider Step 2-4 to be executed in the same way. This algorithm was first given by Kolliopoulos and Stein [8]. Note that SGA is not an on-line algorithm, as it considers all remaining requests in each step.

It is clear that the approximation ratio of SGA is at least as good as that of BGA. To see this, consider an arbitrary instance of MEDP and let BGA process the given requests in the following order: first, all requests accepted by SGA (in the same order as SGA accepts them) and then, all requests rejected by SGA (in arbitrary order). BGA will accept all requests that were accepted by SGA along paths of length at most L, and no other paths. (We can assume that BGA routes the accepted requests along the same paths as SGA.) Thus, the solution produced by BGA cannot be larger than that of SGA if the requests are processed in this order.

Therefore, all upper bounds on the approximation ratio of BGA hold also for SGA. In particular, we obtain that SGA is a 9-approximation algorithm for complete graphs. It is worthwhile to say here that the best previous bound on the approximation ratio of SGA for complete graphs was 54, given by Erlebach and Vukadinović [3].

Moreover, in order to obtain another bound, we use the maximum multiplicity of the requests, denoted by $\mu_{\max}$. Then, we will show that $\mu_{\max}$ is also an upper bound on the approximation ratio of SGA. Therefore, we obtain the next theorem.

Theorem 1.1. *The bounded-length greedy algorithm with $L = 4$ is a 9-approximation algorithm for the maximum edge-disjoint paths problem in complete graphs. The shortest-path-first greedy algorithm is a $\min\{9, \mu_{\max}\}$-approximation algorithm.*

Finally, we consider lower bounds on the approximation ratio of BGA and SGA. We will construct a family of instances of MEDP in complete graphs that implies that SGA cannot be better than a 3-approximation algorithm. Of course, this lower bound applies to BGA as well.

Algorithm:	Shortest-path-first greedy algorithm (SGA)
Input:	an undirected graph $G = (V, E)$ and a multiset $\mathcal{R}$ of requests;
Output:	$\mathcal{A} \subseteq \mathcal{R}$ and a set $\mathcal{P}$ of edge-disjoint paths in G
	such that each request in $\mathcal{A}$ is connected by a path in $\mathcal{P}$;
Step 1:	(Initialize step)
	$\mathcal{A} \leftarrow \emptyset$; $i \leftarrow 0$;
Step 2:	(Main loop)
	while $\mathcal{R}$ contains a request which can be routed in G
Step 2-1:	$\{s_i, t_i\} \leftarrow$ a request in $\mathcal{R}$ such that a shortest path from s_i to t_i in G has minimum length among those of all requests in $\mathcal{R}$;
Step 2-2:	$\mathcal{A} \leftarrow \mathcal{A} \cup \{\{s_i, t_i\}\}$;
Step 2-3:	$\mathcal{R} \leftarrow \mathcal{R} \setminus \{\{s_i, t_i\}\}$;
Step 2-4:	$\pi_i \leftarrow$ a shortest path from s_i to t_i in G;
Step 2-5:	remove all the edges of π_i from G;
Step 2-6:	$i \leftarrow i + 1$;
	end of while;
Step 3:	(Output)
	return $\mathcal{A}$ and $\mathcal{P} \leftarrow \{\pi_i : \{s_i, t_i\} \in \mathcal{A}\}$.

Figure 2: The shortest-path-first greedy algorithm for MEDP.

The organization of this paper is as follows. We will prove the upper bounds (Theorem 1.1) in Section 2; we will construct some instances which give lower bounds in Section 3; some additional remarks will be stated in the last section.

Notation For a finite (multi)set S, $|S|$ represents the size of S, i.e., the number of elements in S. The set of reals is denoted by $\mathbb{R}$, and the set of nonnegative reals is denoted by $\mathbb{R}_+$. A path p is sometimes denoted by $(v_1 \text{ — } v_2 \text{ — } \cdots \text{ — } v_l)$, where $v_1, v_2, \ldots, v_l$ are the consecutive vertices along the path p. The length of the path p, i.e. the number of edges in p, is denoted by $\mathrm{length}(p)$. For two paths p and q, $p \cap q$ represents the set of the edges which p and q have in common. So $p \cap q = \emptyset$ means that p and q are edge-disjoint. Finally, "$A := B$" means that "A is defined by B."

2 Upper bounds

In this section, we will give a better upper bound on the approximation ratio of BGA and SGA for MEDP in complete graphs. In the first subsection, we will review the framework of Kolman and Scheideler [9], in particular the flow number and the shortening lemma, and will see how their result implies a better bound. This yields that BGA with parameter $L = 4$ is a 17-approximation algorithm for MEDP in complete graphs. In the second subsection, we will consider a further improvement. We will show that BGA with $L = 4F$ is an $(8F + 1)$-approximation algorithm for MEDP in graphs with flow number F and thus a 9-approximation algorithm for MEDP in complete graphs. While all bounds for BGA hold also for SGA, we give another bound for SGA depending on the maximum multiplicity of the requests. This gives the proof of the main theorem (Theorem 1.1).

2.1 Flow number of complete graphs

In a recent work by Kolman and Scheideler [9], they considered the unsplittable flow problem (UFP) and gave an improved approximation ratio using the flow number of a network and the shortening lemma. Now we are going to review their results.

UFP is a generalization of MEDP. We are given an undirected graph $G = (V, E)$ and a function $u : E \to \mathbb{R}_+$. The number $u(e)$ associated with an edge $e \in E$ is called the *capacity* of e. We are also given a multiset $\mathcal{R}$ of requests as in MEDP. Moreover, we are given two functions $d : \mathcal{R} \to \mathbb{R}_+$ and $r : \mathcal{R} \to \mathbb{R}_+$, where $d(i)$ and $r(i)$ associated with a request $i \in \mathcal{R}$ is called the *demand* and the *profit* of i, respectively. To summarize, an instance of UFP is given by a 5-tuple $(G, \mathcal{R}, u, d, r)$ consisting of an undirected graph G, a multiset $\mathcal{R}$ of requests, the capacity u, the demand d and the profit r. Then a feasible solution to the instance $(G, \mathcal{R}, u, d, r)$ of UFP is given by a pair $(\mathcal{A}, \mathcal{P})$ of a subset $\mathcal{A}$ of $\mathcal{R}$ and a set $\mathcal{P}$ of paths (not necessarily edge-disjoint) in G such that each request in $\mathcal{A}$ is connected by a path in $\mathcal{P}$ and for each edge $e \in E$ the sum of the demands of the requests connected by the paths going through e does not exceed the capacity $u(e)$ of the edge e. A request is called *accepted* if it belongs to $\mathcal{A}$. The task is to maximize the total profit of the accepted requests. MEDP is a special case of UFP. To appreciate this, we only have to set $u(e) = 1$ for all $e \in E$ and $d(i) = r(i) = 1$ for all $i \in \mathcal{R}$.

UFP is also an NP-hard problem, since it contains MEDP as a special case. Let us define what an approximation algorithm for UFP is. (In the previous section, we just defined approximation algorithms for MEDP.) An algorithm for UFP is called a *ρ-approximation algorithm* if it runs in polynomial time and always outputs a feasible solution $(\mathcal{A}, \mathcal{P})$ for any instance $(G, \mathcal{R}, u, d, r)$ which satisfies $\mathrm{Opt} \leq \rho \sum_{i \in \mathcal{A}} r(i)$, where Opt is the total profit of the accepted requests in an optimal solution. We say that ρ is an *approximation ratio*, similarly to the case of MEDP.

We can formulate UFP as a $\{0,1\}$-integer programming problem, as usual in combinatorial optimization. To do that, we introduce two kinds of variables. One is $x \in \{0,1\}^{\mathcal{R}}$, which means that $x_i = 1$ if the request i is accepted and $x_i = 0$ if i is not accepted. The other one is $y^{(i)} \in \{0,1\}^{\mathcal{P}_i}$, where $i \in \mathcal{R}$ and $\mathcal{P}_i$ is the set of all paths in G that connect the request i. The variable $y^{(i)}$ represents that $y^{(i)}_\pi = 1$ if the request i is accepted through the path $\pi \in \mathcal{P}_i$ and $y^{(i)}_\pi = 0$ otherwise.

Here is a formulation of UFP via integer programming:

(IP-UFP):

$$\begin{aligned}
\text{maximize} \quad & \sum_{i \in \mathcal{R}} r(i) x_i & & \\
\text{subject to} \quad & \sum_{\substack{i \in \mathcal{R}, \pi \in \mathcal{P}_i \\ \text{s.t. } e \in \pi}} d(i) y^{(i)}_\pi \leq u(e) & \text{for all } e \in E, & (1) \\
& \sum_{\pi \in \mathcal{P}_i} y^{(i)}_\pi = x_i & \text{for all } i \in \mathcal{R}, & (2) \\
& x_i \in \{0,1\} & \text{for all } i \in \mathcal{R}, & (3) \\
& y^{(i)}_\pi \in \{0,1\} & \text{for all } i \in \mathcal{R} \text{ and } \pi \in \mathcal{P}_i. & (4)
\end{aligned}$$

The formulation (IP-UFP) is easy to understand but has exponentially many variables. However, we can indeed obtain a formulation of polynomial size as in the network flow problem. See [6], for example.

Now we relax the $\{0,1\}$-constraints (3) and (4), that is, we replace them with $x_i \in [0,1]$ and

$y_\pi^{(i)} \in [0,1]$. Then, we obtain a linear programming problem, called the *multicommodity flow problem* and denoted by (LP-UFP).

There is a variant of the multicommodity flow problem, called the concurrent multicommodity flow problem. The *concurrent multicommodity flow problem* is defined as follows:

(ConMFP):

$$\begin{aligned}
\text{maximize} \quad & x_1 \\
\text{subject to} \quad & \sum_{\substack{i\in\mathcal{R},\pi\in\mathcal{P}_i \\ \text{s.t. } e\in\pi}} d(i) y_\pi^{(i)} \le u(e) && \text{for all } e \in E, \\
& \sum_{\pi\in\mathcal{P}_i} y_\pi^{(i)} = x_i && \text{for all } i \in \mathcal{R}, \\
& x_i = x_j && \text{for all } i,j \in \mathcal{R}, \qquad (5) \\
& 0 \le x_i \le 1 && \text{for all } i \in \mathcal{R}, \\
& 0 \le y_\pi^{(i)} \le 1 && \text{for all } i \in \mathcal{R} \text{ and } \pi \in \mathcal{P}_i.
\end{aligned}$$

In a feasible solution to the concurrent multicommodity flow problem (ConMFP), all x_i's are forced to be the same by the constraint (5). Then, this problem represents the situation that if we are forced to accept the same fraction of all the requests, we want to know how large the fraction can be. Note that any feasible solution of the concurrent multicommodity flow problem (ConMFP) is also a feasible solution of the multicommodity flow problem (LP-UFP), but the converse is not always true.

Now we define the flow number of a network as in [9]. (Here, a *network* is a pair (G,u) of an undirected graph $G=(V,E)$ and a capacity function $u : E \to \mathbb{R}_+$.) We are given an undirected graph $G=(V,E)$ and a capacity function $u : E \to \mathbb{R}_+$. Now we construct an instance $I(G,u)$ of the concurrent multicommodity flow problem. The set $\mathcal{R}$ of requests consists of all the unordered pairs of the vertices, i.e., $\mathcal{R} = \{\{s,t\} : s,t \in V, s \ne t\}$. The demand $d : \mathcal{R} \to \mathbb{R}_+$ is defined as

$$d(\{s,t\}) := \frac{\displaystyle\sum_{\{s,v\}\in E} u(\{s,v\}) \sum_{\{t,v\}\in E} u(\{t,v\})}{\displaystyle 2\sum_{e\in E} u(e)}.$$

Thus, we have obtained an instance $I(G,u) := (G,\mathcal{R},u,d)$ of the concurrent multicommodity flow problem.

For a feasible solution $\xi = (x, (y^{(i)} : i \in \mathcal{R}))$ to an instance $(G,\mathcal{R},u,d)$ of the concurrent multicommodity flow problem, we define two parameters: the dilation and the congestion. The *dilation* $D(\xi)$ of ξ is the length of a longest flow path in ξ, i.e.,

$$D(\xi) := \max\left\{\text{length}(\pi) : y_\pi^{(i)} > 0\right\}.$$

The *congestion* $C(\xi)$ of ξ is the inverse of the objective value of ξ, i.e.,

$$C(\xi) := \frac{1}{x_1}.$$

Then, the *flow number* $F(G,u)$ of the network (G,u) is defined as

$$F(G,u) := \min\{\max\{D(\xi), C(\xi)\} : \xi \text{ is a feasible solution to } I(G,u)\}.$$

Here, we determine the flow number of a complete graph with unit capacity.

Lemma 2.1. *Let $G = (V, E)$ be a complete graph and $u : E \to \mathbb{R}_+$ be given as $u(e) = 1$ for every $e \in E$. Then, it holds that $F(G, u) = 1$.*

Proof. Let us first check what $I(G, u)$ is. The set $\mathcal{R}$ of requests is E, and the demand is specified as $d(\{s, t\}) = (n-1)/n$ for every $\{s, t\} \in \mathcal{R}$.

Let $\xi = (x, (y^{(i)} : i \in \mathcal{R}))$ be a feasible solution to $I(G, u)$. Since $0 \leq x_1 \leq 1$, we have $\max\{D(\xi), C(\xi)\} \geq C(\xi) = 1/x_1 \geq 1$. Then it holds that

$$\begin{aligned} F(G, u) &= \min\{\max\{D(\xi), C(\xi)\} : \xi \text{ is a feasible solution to } I(G, u)\} \\ &\geq \min\{1 : \xi \text{ is a feasible solution to } I(G, u)\} = 1. \end{aligned}$$

Now we show the converse inequality. Let ξ' be the feasible solution to $I(G, u)$ such that for any request $i = \{s, t\}$ and a path $\pi = (s \text{ — } t) \in \mathcal{P}_i$ of length one, we set $y_\pi^{(i)} = x_i = 1$, and all the other variables are set to zero. (We can easily check that ξ' is indeed a feasible solution to $I(G, u)$.) Then we have $D(\xi') = C(\xi') = 1$. Therefore,

$$\begin{aligned} F(G, u) &= \min\{\max\{D(\xi), C(\xi)\} : \xi \text{ is a feasible solution to } I(G, u)\} \\ &\leq \max\{D(\xi'), C(\xi')\} = 1. \end{aligned}$$

This completes the proof. □

Invoking the flow number, Kolman and Scheideler [9] derived several interesting theorems. One is the so-called "shortening lemma."

Lemma 2.2 (shortening lemma [9]). *Let an instance I of the concurrent multicommodity flow problem in a graph with flow number F be given. Then for any $\epsilon \in (0, 1]$ and any feasible solution of I with objective value f, there exists a feasible solution ξ of I such that the objective value is $f/(1+\epsilon)$ and $D(\xi) \leq 2(1 + 1/\epsilon)F$.*

Set $\epsilon = 1$. By the shortening lemma, if we are given a feasible solution of an instance of UFP in which the profits are equal to the demands, then we can transform this solution into a feasible solution to the corresponding instance of the multicommodity flow problem such that it uses only short paths (of length at most $4F$), although the objective value is half as much as the original one. Now let us appreciate this. Given an instance $I = (G, \mathcal{R}, u, d, d)$ of UFP, let $(x, (y^{(i)} : i \in \mathcal{R}))$ be a feasible solution to I. Then we set $\overline{\mathcal{R}} := \{i \in \mathcal{R} : x_i = 1\}$, and consider the instance $\overline{I} = (G, \bar{\mathcal{R}}, u, d)$ of the concurrent multicommodity flow problem. Here if we set $\overline{x}_i = x_i$ for all $i \in \overline{\mathcal{R}}$ and $\overline{y}_\pi^{(i)} = y_\pi^{(i)}$ for all $i \in \overline{\mathcal{R}}$ and $\pi \in \mathcal{P}_i$, then $(\overline{x}, (\overline{y}^{(i)} : i \in \overline{\mathcal{R}}))$ is a feasible solution to $\overline{I}$ and the objective value is 1. Now we can apply the shortening lemma. Then we get a feasible solution to $\overline{I}$ such that the objective value is $1/2$ and the dilation is at most $4F$. This also forms a feasible solution to the instance of the multicommodity flow problem corresponding to the original instance I of UFP, and it gives the objective value half as large as the original solution and the dilation is at most $4F$.

Using this lemma, Kolman and Scheideler [9] showed another interesting theorem, which gives an upper bound on the approximation ratio in terms of the flow number.

Lemma 2.3 ([9]). *Let an instance of the unsplittable flow problem satisfying the following three conditions be given: (1) the demand is equal to the profit for any request; (2) the maximum demand is at most the minimum capacity; (3) the flow number is F. Then the bounded-length greedy algorithm with parameter $L = 4F$ is a $(16F + 1)$-approximation algorithm, for a suitable renumbering of the requests.*

In this lemma, "a suitable renumbering" means that we order the requests such that for two requests i and j if $r(i) > r(j)$ then i comes before j.

Since the flow number of complete graphs with unit capacity is one (Lemma 2.1) and MEDP satisfies the conditions of Lemma 2.3, we immediately obtain the following result.

Theorem 2.4. *The bounded-length greedy algorithm with parameter $L = 4$ (and thus also the shortest-path-first greedy algorithm) is a 17-approximation algorithm for the maximum edge-disjoint paths problem in complete graphs.*

In the next subsection, we derive a better bound.

2.2 A 9-approximation algorithm

Now we will show that SGA is a 9-approximation algorithm for MEDP in complete graphs. Notice again that MEDP is a special case of UFP, setting $u(e) = 1$ for all $e \in E$ and $d(i) = r(i) = 1$ for all $i \in \mathcal{R}$.

First we will refine Lemma 2.3 due to Kolman and Scheideler to obtain a better bound for MEDP.

Lemma 2.5. *The bounded-length greedy algorithm with parameter $L = 4F$ (and thus also the shortest-path-first greedy algorithm) is an $(8F+1)$-approximation algorithm for the maximum edge-disjoint paths problem in undirected graphs with flow number F.*

Proof. Let $\mathcal{A}$ and $\mathcal{P}(\mathcal{A})$ be the set of accepted requests and the set of paths obtained by BGA, respectively. Also let $\mathcal{O}$ be the set of accepted requests in an optimal solution. By the shortening lemma (Lemma 2.2), there exists a feasible solution $\xi = (x, (y^{(i)} : i \in \mathcal{R}))$ of the corresponding multicommodity flow problem such that $D(\xi) \leq 4F$ and the objective value for ξ is $|\mathcal{O}|/2$.

Let $\mathcal{P}(\xi)$ be the set of the paths π satisfying $y_\pi^{(i)} > 0$. Take any path $p \in \mathcal{P}(\mathcal{A})$. Note that $\mathrm{length}(p) \leq 4F$ by definition of BGA. Then we have

$$\sum_{\pi \cap p \neq \emptyset} y_\pi^{(i)} \leq \sum_{\substack{e \in p \\ \pi \cap e \neq \emptyset}} y_\pi^{(i)} \leq \sum_{e \in p} \sum_{e \in \pi} y_\pi^{(i)} \leq \sum_{e \in p} 1 \leq \mathrm{length}(p), \tag{6}$$

using the first constraint (1) of (LP-UFP).

Now we divide $\mathcal{P}(\xi)$ into two parts $\mathcal{P}_1(\xi)$ and $\mathcal{P}_2(\xi)$ as follows:

$$\mathcal{P}_1(\xi) = \{\pi \in \mathcal{P}(\xi) : \text{ the request connected by } \pi \text{ does not belong to } \mathcal{A}\},$$
$$\mathcal{P}_2(\xi) = \mathcal{P}(\xi) \setminus \mathcal{P}_1(\xi).$$

Then, we have the following claim.

Claim 2.5.1. *For each path $\pi \in \mathcal{P}_1(\xi)$ there exists some path $p \in \mathcal{P}(\mathcal{A})$ such that $\pi \cap p \neq \emptyset$, i.e., π and p have a common edge.*

Proof of Claim. Consider the contrary. Then, we happen to have some request $r \in \mathcal{R} \setminus \mathcal{A}$ which is connected by a path of length at most $4F$ consisting of edges that no path in $\mathcal{P}(\mathcal{A})$ uses. So BGA with $L = 4F$ should have accepted this request. A contradiction. □

Now it is time to analyze the approximation ratio. First, using Claim 2.5.1 and (6) we have

$$\sum_{\pi\in\mathcal{P}_1(\xi)} y_\pi^{(i)} \le \sum_{p\in\mathcal{P}(\mathcal{A})} \sum_{\pi\cap p\neq\emptyset} y_\pi^{(i)} \le \sum_{p\in\mathcal{P}(\mathcal{A})} \mathrm{length}(p) \le 4F\ |\mathcal{P}(\mathcal{A})| = 4F\ |\mathcal{A}|.$$

Secondly, by the discussion after Lemma 2.2 we have

$$\sum_{\pi\in\mathcal{P}_2(\xi)} y_\pi^{(i)} \le |\mathcal{A}|/2.$$

Therefore, it holds that

$$\begin{aligned}\mathrm{Opt} = |\mathcal{O}| &= 2 \times (\text{the objective value for } \xi) = 2\sum_{i\in\mathcal{R}} \sum_{\pi\in\mathcal{P}_i} y_\pi^{(i)} = 2 \sum_{\pi\in\mathcal{P}(\xi)} y_\pi^{(i)} \\ &= 2\left(\sum_{\pi\in\mathcal{P}_1(\xi)} y_\pi^{(i)} + \sum_{\pi\in\mathcal{P}_2(\xi)} y_\pi^{(i)}\right) \le 2(4F|\mathcal{A}| + |\mathcal{A}|/2) = (8F+1)|\mathcal{A}|.\end{aligned}$$

Thus, we have shown that the approximation ratio is $8F+1$. □

From Lemmas 2.1 and 2.5, we immediately obtain the following corollary.

Corollary 2.6. *The bounded-length greedy algorithm with parameter $L = 4$ (and thus also the shortest-path-first greedy algorithm) is a 9-approximation algorithm for the maximum edge-disjoint paths problem in complete graphs.*

For another bound, we define the multiplicities of requests. Consider the multiset $\mathcal{R}$ of requests. For a request $\{s,t\} \in \mathcal{R}$, if $\mathcal{R}$ has μ copies of $\{s,t\}$ (recall that $\mathcal{R}$ is a multiset), then we say the *multiplicity* of $\{s,t\}$ in $\mathcal{R}$ is μ. For example, when $\mathcal{R} = \{\{1,2\},\{1,2\},\{2,4\},\{3,4\},\{3,4\},\{3,4\}\}$, we observe that $\{1,2\}$ has the multiplicity two, $\{2,4\}$ has one, and $\{3,4\}$ has three. Define $\mu_{\max}(\mathcal{R})$ as the maximum of the multiplicities of the requests in $\mathcal{R}$. If there is no risk of confusion, we denote it simply by $\mu_{\max}$. For the example above, we have $\mu_{\max} = 3$.

Here, we have a slight improvement for small $\mu_{\max}$. Let $\mathcal{R}_1$ be the set of requests neglecting the multiplicities. For the example above, we have $\mathcal{R}_1 = \{\{1,2\},\{2,4\},\{3,4\}\}$.

Lemma 2.7. *For the maximum edge-disjoint paths problem in complete graphs with the set $\mathcal{R}$ of requests, the shortest-path-first greedy algorithm gives a solution $\mathcal{A}$ such that*

(*) *each request in $\mathcal{R}_1$ is accepted and connected by a path of length 1 (i.e. just an edge).*

Moreover, there exists an optimal solution which satisfies the condition (*).

Proof. Straightforward. □

Then we have $|\mathcal{R}_1| \le |\mathcal{A}|$. Moreover, $\mu_{\max}|\mathcal{R}_1|$ is an obvious upper bound on $|\mathcal{R}|$. Also, $|\mathcal{R}|$ bounds the optimal value Opt. Now, it is time to analyze the approximation ratio:

$$\mathrm{Opt} \le |\mathcal{R}| \le \mu_{\max}|\mathcal{R}_1| \le \mu_{\max}|\mathcal{A}|.$$

Combining this analysis and Corollary 2.6, we have Theorem 1.1.

3 Lower bounds

In the previous section, we have shown that SGA is a $\min\{9, \mu_{\max}\}$-approximation algorithm. Then, the next question is: how can we construct a tight example? If we want to construct a tight example for the bound 9, then it must have maximum multiplicity at least 9. Unfortunately, we do not know an instance that achieves the bound above. In this section, we construct two instances of MEDP in complete graphs. The first one has $\mu_{\max} = 2$ and achieves the ratio 4/3. The next one has an arbitrarily large $\mu_{\max}$ and achieves the ratio $3 - \epsilon$ for arbitrarily small $\epsilon > 0$. Both lower bounds apply to SGA and therefore also to BGA.

3.1 Case of $\mu_{\max} = 2$

In this subsection, we construct an example of MEDP in complete graphs with $\mu_{\max} = 2$ for which SGA has approximation ratio 4/3. Our example has 8 vertices and 16 requests. The optimal solution accepts all the request, while SGA may return a set of 12 requests. Then the ratio is 4/3.

Let the vertex set be $V = \{1, 2, 3, 4, 5, 6, 7, 8\}$. Then 16 requests are given as follows:

$$
\begin{array}{llll}
r_1 := \{1, 3\}, & r_2 := \{1, 3\}, & r_3 := \{5, 3\}, & r_4 := \{5, 3\}, \\
r_5 := \{1, 7\}, & r_6 := \{1, 7\}, & r_7 := \{5, 7\}, & r_8 := \{5, 7\}, \\
r_9 := \{2, 4\}, & r_{10} := \{2, 4\}, & r_{11} := \{8, 6\}, & r_{12} := \{8, 6\}, \\
r_{13} := \{3, 8\}, & r_{14} := \{3, 6\}, & r_{15} := \{7, 2\}, & r_{16} := \{7, 4\}.
\end{array}
$$

The set $\mathcal{R}$ consists of these 16 requests, and we can see that $\mu_{\max} = 2$. Figure 3 shows how the requests are distributed on K_8, where each request r_i is indicated by $\{s_i, t_i\}$ and the vertices are labeled counterclockwise as the top is 1.

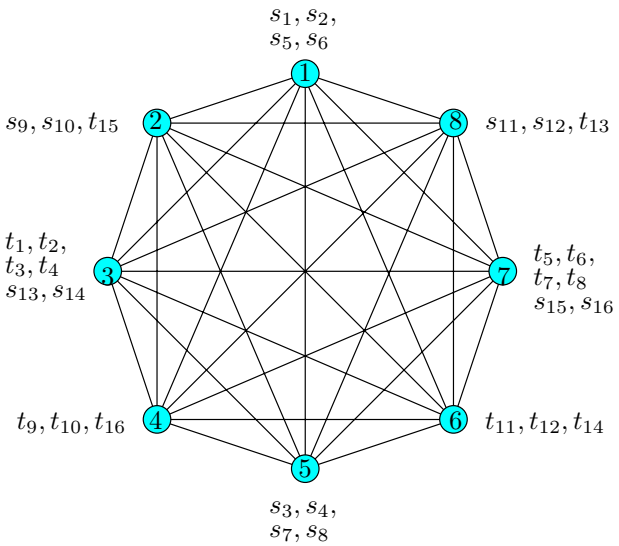

Figure 3: Example with 8 vertices and $\mu_{\max} = 2$.

By Lemma 2.7, the output of SGA and the optimal solution accept each request in $\mathcal{R}_1$ through the corresponding edge. In our example, we have $\mathcal{R}_1 = \{r_1, r_3, r_5, r_7, r_9, r_{11}, r_{13}, r_{14}, r_{15}, r_{16}\}$. So

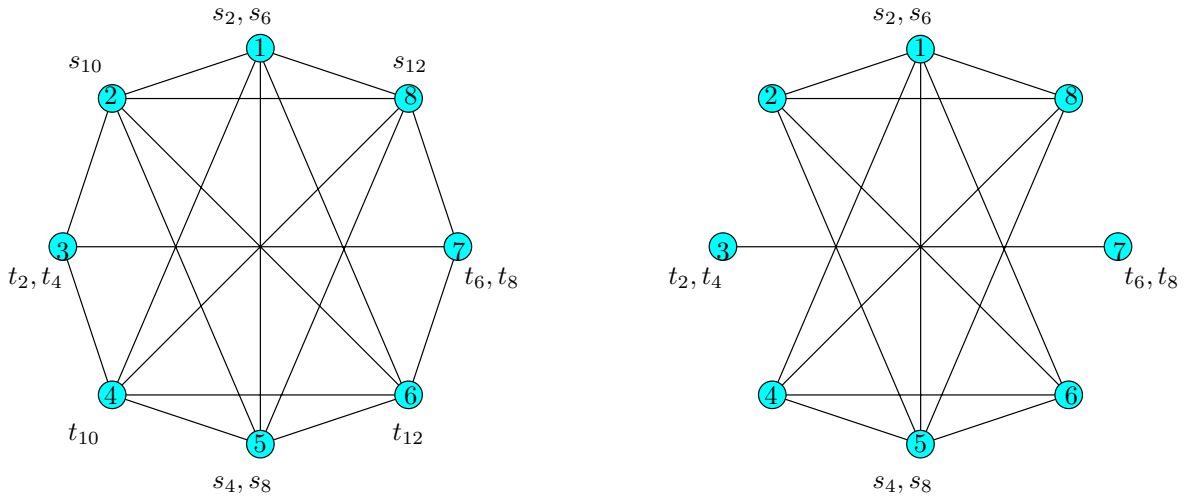

Figure 4: Situation after removing some requests and edges.

now, we remove these requests from $\mathcal{R}$ and also remove the corresponding edges from K_8. This procedure results in the situation shown by Figure 4 (left).

Now we see that an optimal solution accepts all of them as follows:

The request r_2 is connected by the path $(1 — 2 — 3)$;
The request r_4 is connected by the path $(5 — 4 — 3)$;
The request r_6 is connected by the path $(7 — 8 — 1)$;
The request r_8 is connected by the path $(5 — 6 — 7)$;
The request r_{10} is connected by the path $(2 — 8 — 4)$;
The request r_{12} is connected by the path $(6 — 1 — 5 — 8)$.

Next, consider what SGA returns. To do this, we first calculate a shortest path with respect to each request which has not been accepted.

For r_2, $(1 — 2 — 3)$ is a shortest path and the length is 2.
For r_4, $(5 — 4 — 3)$ is a shortest path and the length is 2.
For r_6, $(1 — 8 — 7)$ is a shortest path and the length is 2.
For r_8, $(5 — 6 — 7)$ is a shortest path and the length is 2.
For r_{10}, $(2 — 3 — 4)$ is a shortest path and the length is 2.
For r_{12}, $(8 — 7 — 6)$ is a shortest path and the length is 2.

Since we can choose any of them as a next accepted request, here we choose r_{10}. Then remove the request r_{10} and the edges $\{2, 3\}$ and $\{3, 4\}$. Now, the shortest paths for r_2 and r_4 are changed.

For r_2, $(1 — 6 — 7 — 3)$ is a shortest path and the length is 3.
For r_4, $(5 — 6 — 7 — 3)$ is a shortest path and the length is 3.

So, at the next iteration, we can choose r_6, r_8 or r_{12}. Here we choose r_{12}, and remove r_{12} and the edges $\{6, 7\}$ and $\{7, 8\}$. The requests and the edges which are left for us are shown in Figure 4 (right). We can see that there is no path for r_2, r_4, r_6 or r_8. In this way, we get a solution with 12 accepted requests by SGA, and this gives the approximation ratio 4/3.

3.2 Not better than 3-approximation

In this subsection, we construct a family of instances of MEDP in complete graphs for which SGA has approximation ratio $3-\epsilon$ for arbitrarily small $\epsilon > 0$.

Consider the complete graph with $2n$ vertices for large n. Let $V_v \cup V_a \cup V_b \cup V_c$ be the vertex set, and for k being divisible by 3 and a little bit smaller than n, set $V_v = \{v_1, \ldots, v_{2(n-k)}\}$, $V_a = \{a_1, \ldots, a_{2k/3}\}$, $V_b = \{b_1, \ldots, b_{2k/3}\}$ and $V_c = \{c_1, \ldots, c_{2k/3}\}$. (More explicitly, k only has to satisfy that $3n/5 \leq k < n$ and should be divisible by 3.) Now the set of requests $\mathcal{A} = \mathcal{A}_v \cup \mathcal{A}_a \cup \mathcal{A}_b \cup \mathcal{A}_c$ is given as follows. $\mathcal{A}_v$ consists of $n^2 - k^2$ requests as for all $i \in \{1, 3, 5, \ldots, 2(n-k)-1\}$ we have $2n - i$ requests between the vertices v_i and v_{i+1}. $\mathcal{A}_a$ consists of $k(n-k+1)/3$ requests as for all $i \in \{1, 3, 5, \ldots, 2k/3 - 1\}$ we have $n - k + 1$ requests between the vertex a_i and a_{i+1}. $\mathcal{A}_b$ and $\mathcal{A}_c$ consist of $k(n-k+1)/3$ requests respectively and are constructed similarly as $\mathcal{A}_a$. Figure 5 shows the situation.

The optimal solution accepts all of these requests. How does it accept all of them? First consider accepting the requests of $\mathcal{A}_v$ as the following steps:

Step 1:	consider $2n-1$ requests between v_1 and v_2;
Step 1-1:	we accept one of them using only one edge $(v_1 \text{ — } v_2)$;
Step 1-2:	we accept the other $2n-2$ requests using two edges $(v_1 \text{ — } u \text{ — } v_2)$ for all $u \in (V_v \setminus \{v_1, v_2\}) \cup V_a \cup V_b \cup V_c$; thus we have accepted all of the $2n-1$ requests between v_1 and v_2;
Step 2:	consider $2n-3$ requests between v_3 and v_4;
Step 2-1:	we accept one of them using only one edge $(v_3 \text{ — } v_4)$;
Step 2-2:	we accept the other $2n-4$ requests using two edges $(v_3 \text{ — } u \text{ — } v_4)$ for all $u \in (V_v \setminus \{v_1, v_2, v_3, v_4\}) \cup V_a \cup V_b \cup V_c$; thus we have accepted all of the $2n-3$ requests between v_3 and v_4;
$\vdots$	
Step j:	consider $2n-2j+1$ requests between v_{2j-1} and v_{2j};
Step j-1:	we accept one of them using only one edge $(v_{2j-1} \text{ — } v_{2j})$;
Step j-2:	we accept the other $2n-2j$ requests using two edges $(v_{2j-1} \text{ — } u \text{ — } v_{2j})$ for all $u \in (V_v \setminus \{v_1, \ldots, v_{2j}\}) \cup V_a \cup V_b \cup V_c$; thus we have accepted all of the $2n-2j+1$ requests between v_{2j-1} and v_{2j};
$\vdots$	
Step $(n-k)$:	consider $2k+1$ requests between $v_{2(n-k)-1}$ and $v_{2(n-k)}$;
Step $(n-k)$-1:	we accept one of them using only one edge $(v_{2(n-k)-1} \text{ — } v_{2(n-k)})$;
Step $(n-k)$-2:	we accept the other $2k$ requests using two edges $(v_{2(n-k)-1} \text{ — } u \text{ — } v_{2(n-k)})$ for all $u \in V_a \cup V_b \cup V_c$; thus we have accepted all of the $2k+1$ requests between $v_{2(n-k)-1}$ and $v_{2(n-k)}$.

Then, we consider accepting the requests of $\mathcal{A}_a$ as follows: for any $i \in \{1, 3, \ldots, 2k/3 - 1\}$, among the $n-k+1$ requests between a_i and a_{i+1} one request is accepted via a path with only one edge $(a_i \text{ — } a_{i+1})$ and the other $n-k$ requests are accepted via paths with two edges $(a_i \text{ — } u \text{ — } a_{i+1})$ where $u \in V_b$. Here, the condition that $k \geq 3n/5$ enables all of them to be accepted in this manner. Similarly, we can accept all of the requests of $\mathcal{A}_b$ and $\mathcal{A}_c$ using V_c and V_a, respectively. Thus the optimal solution accepts all of the $n^2 + kn + k - 2k^2$ requests.

Now consider the output of SGA. The algorithm first accepts the requests with a shortest path of length 1. So we have n accepted requests: one between v_i and v_{i+1} for each $i \in \{1, 3, \ldots, 2(n-k)-$

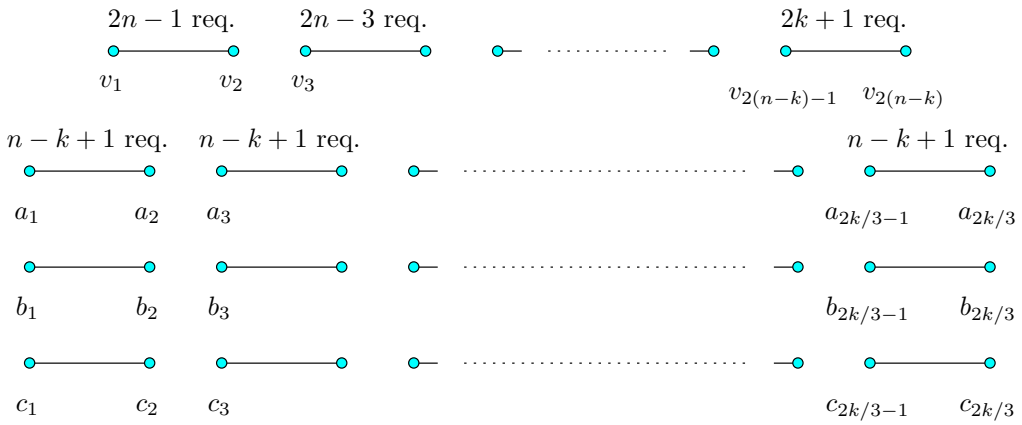

Figure 5: Construction of a bad example for the shortest-path-first greedy algorithm.

$1\}$, one request between a_i and a_{i+1} for each $i \in \{1, 3, \ldots, 2k/3-1\}$, one request between b_i and b_{i+1} for each $i \in \{1, 3, \ldots, 2k/3-1\}$, and one request between c_i and c_{i+1} for each $i \in \{1, 3, \ldots, 2k/3-1\}$.

Then we will accept the requests with a shortest path of length 2. Now consider the remaining $n-k$ requests between a_1 and a_2. They can be accepted along paths $(a_1 \text{ — } v_i \text{ — } a_2)$ where $i \in \{1, 3, \ldots, 2(n-k)-1\}$. In the same manner, for $j \in \{3, 5, \ldots, 2k/3-1\}$ we accept $n-k$ requests between a_j and a_{j+1} along paths $(a_j \text{ — } v_i \text{ — } a_{j+1})$ where $i \in \{1, 3, \ldots, 2(n-k)-1\}$. Thus we have accepted all of the $k(n-k+1)/3$ requests in $\mathcal{A}_a$. Similarly, we can accept all of the $2k(n-k+1)/3$ requests in $\mathcal{A}_b$ and $\mathcal{A}_c$ by using $v_1, v_3, v_5, \ldots, v_{2(n-k)-1}$ as internal vertices on paths of length 2.

Then we consider accepting some requests in $\mathcal{A}_v$. The remaining $2k$ requests between $v_{2(n-k)-1}$ and $v_{2(n-k)}$ can be accepted only through the vertices of $V_v \setminus \{v_{2(n-k)-1}, v_{2(n-k)}\}$. Since $|V_v \setminus \{v_{2(n-k)-1}, v_{2(n-k)}\}| = 2(n-k-1)$, we can only accept $2(n-k-1)$ requests out of $2k$. In total, among the $2k+1$ requests between $v_{2(n-k)-1}$ and $v_{2(n-k)}$ we have accepted only $2(n-k-1)+1$ requests. Next, we look at the $2k+2$ requests between $v_{2(n-k)-3}$ and $v_{2(n-k)-2}$, which can be accepted only through the vertices of $V_v \setminus \{v_{2(n-k)-3}, v_{2(n-k)-2}, v_{2(n-k)-1}, v_{2(n-k)}\}$. By the same reason as before, we can only accept $2(n-k-2)$ requests out of $2k+2$. In total, among the $2k+3$ requests between $v_{2(n-k)-3}$ and $v_{2(n-k)-2}$ we have accepted only $2(n-k-2)+1$ requests. Going in this way from right to left, we will accept only i requests among $2n-i$ requests between v_i and v_{i+1} for all $i \in \{1, 3, \ldots, 2(n-k)-1\}$. Then after accepting all of these requests, we cannot go further. So, we have accepted $n^2 - kn + k$ requests by SGA.

Let $k = \alpha n$ with $3/5 \leq \alpha < 1$. The approximation ratio is

$$\frac{n^2+kn+k-2k^2}{n^2-kn+k} = \frac{n^2+\alpha n^2+\alpha n-2\alpha^2 n^2}{n^2-\alpha n^2+\alpha n} \xrightarrow{n\to\infty} \frac{1+\alpha-2\alpha^2}{1-\alpha} = 1+2\alpha.$$

Set $\alpha = 1-\epsilon/2$, then we get $1+2\alpha = 3-\epsilon$ as desired.

4 Conclusion

We have studied the approximation ratio achieved by the simple and practical algorithms SGA and BGA for MEDP in complete graphs. Previously, only an upper bound of 54 on the ratio of SGA had been shown, and no non-trivial lower bound was known for these algorithms. We have proved

that BGA with $L = 4$ and SGA are 9-approximation algorithms for complete graphs and cannot be better than 3-approximation algorithms. This has substantially reduced the gap between upper and lower bounds on the approximation ratio of these algorithms. However, a gap remains and it would be interesting to discover the exact worst-case approximation ratio of these algorithms.

Furthermore, MEDP in complete graphs is NP-hard, but we are not aware of any inapproximability results. In particular, it is not known whether MEDP in complete graphs is APX-hard. Therefore, there is still the possibility that a polynomial-time approximation scheme can be obtained. (MEDP in general undirected graphs is known to be APX-hard, even for trees of rings [2, 5].) So the inapproximability of the problem should be considered as well as better approximation algorithms.

References

[1] C. Chekuri and S. Khanna, Edge disjoint paths revisited. In: *Proceedings of the 14th Annual ACM-SIAM Symposium on Discrete Algorithms* (SODA 2003), 2003, 628–637.

[2] T. Erlebach, Approximation algorithms and complexity results for path problems in trees of rings. Technical Report TIK-Report 109, Computer Engineering and Networks Laboratory (TIK), ETH Zürich, June 2001.

[3] T. Erlebach and D. Vukadinović, New results for path problems in generalized stars, complete graphs, and brick wall graphs. In: Proceedings of the 13th International Symposium on Fundamentals of Computation Theory (FCT 2001), *Lecture Notes in Computer Science* **2138**, 2001, 483–494.

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

[5] N. Garg, V.V. Vazirani and M. Yannakakis, Primal-dual approximation algorithms for integral flow and multicut in trees. *Algorithmica* **18**, 1997, 3–20.

[6] V. Guruswami, S. Khanna, R. Rajaraman, B. Shepherd and M. Yannakakis, Near-optimal hardness results and approximation algorithms for edge-disjoint paths and related problems. In: *Proceedings of the 31st Annual ACM Symposium on Theory of Computing* (STOC'99), 1999, 19–28.

[7] J.M. Kleinberg, *Approximation algorithms for disjoint paths problems*. Ph.D. thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, 1996.

[8] S.G. Kolliopoulos and C. Stein, Approximating disjoint-path problems using greedy algorithms and packing integer programs. In: Integer Programming and Combinatorial Optimization, Proceedings of the 6th Integer Programming and Combinatorial Optimization Conference IPCO VI, *Lecture Notes in Computer Science* **1412**, 1998, 153–168.

[9] P. Kolman and C. Scheideler, Improved bounds for the unsplittable flow problem. In: *Proceedings of the 13th Annual ACM-SIAM Symposium on Discrete Algorithms* (SODA 2002), 2002, 184–193.