# Greedy edge-disjoint paths in complete graphs*

Paz Carmi[1], Thomas Erlebach[2], and Yoshio Okamoto[3]

[1] Department of Computer Science, Ben-Gurion University of the Negev, Israel. carmip@cs.bgu.ac.il

[2] Computer Engineering and Networks Laboratory, Department of Information Technology and Electrical Engineering, ETH Zürich, Switzerland. erlebach@tik.ee.ethz.ch

[3] Institute of Theoretical Computer Science, Department of Computer Science, ETH Zürich, Switzerland. okamotoy@inf.ethz.ch

**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, Shortening lemma.

## 1 Introduction

The maximum edge-disjoint paths problem (MEDP) is one of the most classical NP-hard problems [5]. Several real-world problems concerning the efficient operation of communication networks lead to problems that can be modeled as MEDP.

An instance of MEDP is given by a pair $(G, \mathcal{R})$ of an undirected connected 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*. Two paths are called *edge-disjoint* if they have no common edge. We consider the problem of connecting as many requests as possible along edge-disjoint paths. A feasible solution to the instance $(G, \mathcal{R})$ of

* 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), by the Swiss National Science Foundation under Contract No. 2100-63563.00 (AAPCN), and by EU Thematic Network APPOL II, IST-2001-32007, with funding by the Swiss Federal Office for Education and Science (BBW). A longer version of this paper is available as technical report [1].

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

| approximation ratio | algorithm | contributors [reference] |
|---|---|---|
| 27 | Tripartition | Erlebach and Vukadinović [4] |
| 54 | SGA | Erlebach and Vukadinović [4] |
| 17 | BGA ($L = 4$), SGA | Kolman and Scheideler [10] |
| 9 | BGA ($L = 4$), SGA | this paper |

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 unique 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 $\mathcal{A}$.

Since it is known that MEDP is NP-hard even if input graphs are restricted to complete graphs [4], we are interested in finding approximation algorithms for MEDP. An algorithm for MEDP is called a *$\rho$-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 $(G, \mathcal{R})$. The value $\rho$ is also called *approximation ratio*. We refer to [8] for an introduction and more background information about edge-disjoint paths problems and to [2] and the references given there for recent results about the approximability of MEDP in general undirected and directed graphs.

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 [4]. In this 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 those requests with both ends in $V_i$ or one end in $V_i$ and the other 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.

Greedy algorithms appear to be the simplest algorithms for MEDP. The first greedy algorithm that we consider is called the *bounded-length greedy algorithm* (BGA). It takes a parameter $L$ and accepts a request only if it can be routed along a path of length at most $L$. (The *length* of a path is defined as the number of edges on the path.) More formally, the algorithm processes the requests in given order and greedily accepts each request if it can be routed along a path of length at most $L$ that is edge-disjoint from the paths of all previously accepted requests. Otherwise, it rejects the request. BGA was first introduced by Kleinberg [8]. Note that BGA can process the requests in arbitrary order and is therefore an online algorithm, i.e., it can process each request without knowledge about future requests.

Kolman and Scheideler [10] invented a new network parameter $F$, 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 that will be explained in Section 2.1). Indeed, they discovered a "shortening lemma" and used it nicely to obtain this bound. We will see that complete graphs have flow number 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 $L = 4F$ is actually an $(8F + 1)$-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). This algorithm starts with $\mathcal{A} = \emptyset$ and repeats the following step until there is no request left in $\mathcal{R}$ that can be connected along a path in $G$: Let $\{s_i, t_i\}$ be a request in $\mathcal{R}$ such that a shortest path $\pi_i$ from $s_i$ to $t_i$ in $G$ has minimum length among all the requests in $\mathcal{R}$ (ties can be broken arbitrarily); accept $\{s_i, t_i\}$ (i.e., assign $\mathcal{A} \leftarrow \mathcal{A} \cup \{\{s_i, t_i\}\}$), assign it the path $\pi_i$, remove $\{s_i, t_i\}$ from $\mathcal{R}$, and delete all edges of $\pi_i$ from $G$. In other words, the algorithm greedily accepts the request that can be connected along a shortest possible path that is disjoint to the paths of previously accepted requests. This algorithm was first given by Kolliopoulos and Stein [9]. 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 have a larger value 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 [4].

We obtain another bound in terms of the maximum multiplicity (the maximum number of copies of the same request in the given instance), denoted by $\mu_{\max}$: we show that $\mu_{\max}$ is also an upper bound on the approximation ratio of SGA. Therefore, we get the following 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.

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* 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 $(v_1 — v_2 — \cdots — v_l)$, 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.

## 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 [10], 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 [10], they studied the unsplittable flow problem (UFP) and gave an improved approximation ratio using the flow number of a network and the shortening lemma. First, 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 $i \in \mathcal{R}$ are 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 unique 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 *$\rho$-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 $\rho$ is an *approximation ratio*, similarly to the case of MEDP. Note that these definitions are consistent with those for 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:

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

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 (using edge variables instead of path variables). See [7], for example.

Now we relax the $\{0,1\}$-constraints (3) and (4), that is, we replace them with $x_i \in [0,1]$ and $y^{(i)}_\pi \in [0,1]$. 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:

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

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 to the concurrent multicommodity flow problem (ConMFP) is also a feasible solution to the multicommodity flow problem (LP-UFP), but the converse is not always true.

Now we define the flow number of a network as in [10]. (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 a network $(G,u)$. Consider an instance $I(G,u)$ of the concurrent multicommodity flow problem that is constructed as follows. The set $\mathcal{R}$ of requests consists of all the unordered pairs of the vertices, i.e., $\mathcal{R} = \{\{s,t\} : s,t \in V,\ s \neq t\}$. The demand $d : \mathcal{R} \to \mathbb{R}_+$ is defined as $d(\{s,t\}) := (\sum_{\{s,v\}\in E} u(\{s,v\}) \sum_{\{t,v\}\in E} u(\{t,v\}))/(2\sum_{e\in E} u(e))$. So we have 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 $\xi$ is the length of a longest flow path in $\xi$, i.e., $D(\xi) := \max\left\{\text{length}(\pi) : y_\pi^{(i)} > 0\right\}$. The *congestion* $C(\xi)$ of $\xi$ is the inverse of the objective value of $\xi$, i.e., $C(\xi) := \frac{1}{x_1}$. Finally, the *flow number* $F(G,u)$ of the network $(G,u)$ is defined as

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

Now it is easy to 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 $F(G,u)=1$.*

*Proof.* First, it is clear that $F(G,u) \ge 1$. To see that $F(G,u) \le 1$, note that $I(G,u)$ consists of a request $\{s,t\}$ with demand $d(\{s,t\}) = (n-1)/n$ for every edge $\{s,t\} \in E$. This means that we can route the whole demand of every request along its direct edge, resulting in a solution with dilation 1 and congestion 1. □

Invoking the flow number, Kolman and Scheideler [10] derived several interesting statements. One is the so-called "shortening lemma."

**Lemma 2.2 (shortening lemma [10]).** *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 to $I$ with objective value $f$, there exists a feasible solution $\xi$ to $I$ such that the objective value is $f/(1+\epsilon)$ and $D(\xi) \leq 2(1+1/\epsilon)F$.*

From Lemma 2.2, we can conclude the following corollary.

**Corollary 2.3.** *If we are given a feasible solution to an instance of UFP in which the profit is equal to the demand, then we can transform this solution into a feasible solution to the corresponding instance of the multicommodity flow problem such that it uses only paths of length at most $4F$ but the objective value is half as much as the original one.*

*Proof.* 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 $\overline{I} = (G, \overline{\mathcal{R}}, u, d)$ 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 with $\epsilon = 1$. 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 shows that the objective value is half as large as the original solution and the dilation is at most $4F$. □

Using this corollary, Kolman and Scheideler [10] showed that BGA with $L = 4F$ is a $(16F + 1)$-approximation algorithm for UFP in networks with flow number $F$, provided that the profit of each request is equal to its demand, the maximum demand is at most the minimum capacity, and the algorithm processes the requests in order of non-increasing profits. Since these conditions are satisfied for MEDP and the flow number of complete graphs with unit capacity is one by Lemma 2.1, we immediately obtain that BGA with $L = 4$ (and thus also SGA) is a 17-approximation algorithm for MEDP in complete graphs. In the next subsection, we derive a better bound.

### 2.2 A 9-approximation algorithm

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 the analysis by Kolman and Scheideler for UFP [10] to obtain a better bound for MEDP.

**Theorem 2.4.** *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. Let further $\mathcal{O}$ be the set of accepted requests in an optimal solution. By Corollary 2.3, 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 $\xi$ is $|\mathcal{O}|/2$. Notice that the proof of Corollary 2.3 shows that $\sum_{\pi \in \mathcal{P}_i} y_\pi^{(i)} \leq 1/2$ holds for all $i \in \mathcal{R}$.

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

$$\sum_{\substack{\pi \in \mathcal{P}(\xi) \\ \text{s.t. } \pi \cap p \neq \emptyset}} y_\pi^{(i)} \leq \sum_{\substack{\pi \in \mathcal{P}(\xi) \\ \text{s.t. } \exists e \in \pi \cap p}} y_\pi^{(i)} \leq \sum_{e \in p} \sum_{\substack{\pi \in \mathcal{P}(\xi) \\ \text{s.t. } e \in \pi}} y_\pi^{(i)} \leq \sum_{e \in p} 1 \leq \text{length}(p) \leq 4F, \quad (6)$$

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

*Claim.* 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., $\pi$ and $p$ have a common edge.

*Proof (of Claim).* Consider the contrary. Then, we must 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 we analyze the approximation ratio. Using the claim above and (6), we get

$$\sum_{\pi \in \mathcal{P}_1(\xi)} y_\pi^{(i)} \leq \sum_{p \in \mathcal{P}(\mathcal{A})} \sum_{\substack{\pi \in \mathcal{P}(\xi) \\ \text{s.t. } \pi \cap p \neq \emptyset}} y_\pi^{(i)} \leq 4F \ |\mathcal{P}(\mathcal{A})| = 4F \ |\mathcal{A}|.$$

On the other hand, we have $\sum_{\pi \in \mathcal{P}_2(\xi)} y_\pi^{(i)} \leq \sum_{i \in \mathcal{A}} \sum_{\pi \in \mathcal{P}_i} y_\pi^{(i)} \leq \sum_{i \in \mathcal{A}} 1/2 = |\mathcal{A}|/2$. This gives

$$\text{Opt} = |\mathcal{O}| = 2 \times (\text{the objective value for } \xi) = 2 \sum_{i \in \mathcal{R}} \sum_{\pi \in \mathcal{P}_i} y_\pi^{(i)} = 2 \sum_{\pi \in \mathcal{P}(\xi)} y_\pi^{(i)}$$

$$= 2 \left( \sum_{\pi \in \mathcal{P}_1(\xi)} y_\pi^{(i)} + \sum_{\pi \in \mathcal{P}_2(\xi)} y_\pi^{(i)} \right) \leq 2(4F|\mathcal{A}| + |\mathcal{A}|/2) = (8F + 1)|\mathcal{A}|.$$

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

From Lemma 2.1 and Theorem 2.4, we immediately obtain the following corollary.

**Corollary 2.5.** *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.*

Now we consider the multiplicities of requests. For a request $\{s,t\} \in \mathcal{R}$, if $\mathcal{R}$ has $\mu$ copies of $\{s,t\}$, then we say the *multiplicity* of $\{s,t\}$ in $\mathcal{R}$ is $\mu$. 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 $\mu_{\max}$. Let $\mathcal{R}_1$ be the set of requests neglecting the multiplicities (i.e., with only one copy of each unique request in $\mathcal{R}$). The proof of the following lemma is straightforward.

**Lemma 2.6.** *For the maximum edge-disjoint paths problem in complete graphs with the multiset $\mathcal{R}$ 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 one edge). Moreover, there exists an optimal solution with the same property.*

Then we have $|\mathcal{R}_1| \leq |\mathcal{A}|$. Moreover, $\mu_{\max}|\mathcal{R}_1|$ is an obvious upper bound on $|\mathcal{R}|$. Also, $|\mathcal{R}|$ bounds Opt. This shows Opt $\leq |\mathcal{R}| \leq \mu_{\max}|\mathcal{R}_1| \leq \mu_{\max}|\mathcal{A}|$. Combining this analysis and Corollary 2.5, we obtain Theorem 1.1.

## 3 Lower bounds

In the previous section, we have shown that SGA is a $\min\{9, \mu_{\max}\}$-approximation algorithm. The next question is: how tight is this analysis? 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. However, we can construct two instances of MEDP in complete graphs to show non-trivial lower bounds on the approximation ratio of SGA and BGA. The first one has $\mu_{\max} = 2$ and achieves the ratio 4/3. The second 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 instance 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 requests, 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 a set $\mathcal{R}$ consists of the following 16 requests: $r_1 = \{1,3\}$, $r_2 = \{1,3\}$, $r_3 = \{5,3\}$, $r_4 = \{5,3\}$, $r_5 = \{1,7\}$, $r_6 = \{1,7\}$, $r_7 = \{5,7\}$, $r_8 = \{5,7\}$, $r_9 = \{2,4\}$, $r_{10} = \{2,4\}$, $r_{11} = \{8,6\}$, $r_{12} = \{8,6\}$, $r_{13} = \{3,8\}$, $r_{14} = \{3,6\}$, $r_{15} = \{7,2\}$, $r_{16} = \{7,4\}$. Note that $\mu_{\max} = 2$.

By Lemma 2.6, the output of SGA and the optimal solution accept each request in $\mathcal{R}_1$ through the corresponding edge. 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 we can remove these requests from $\mathcal{R}$ and also remove the corresponding edges from $K_8$. This procedure results in the situation shown in Figure 1 (left). Now we can see that an optimal solution accepts all the remaining requests. Next, consider SGA. In the situation of Figure 1 (left), all remaining requests can be connected along paths of length 2. Assume that SGA accepts

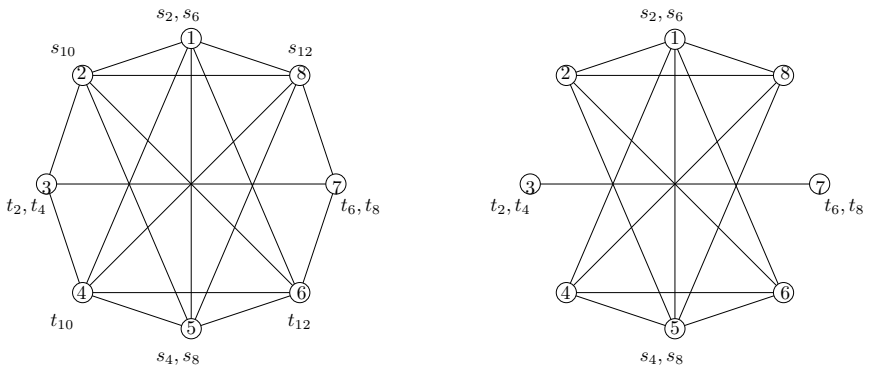


**Fig. 1.** Situation after removing the requests in $\mathcal{R}_1$ from $\mathcal{R}$ and their paths from $K_8$.

$r_{10}$ along (2 — 3 — 4) and $r_{12}$ along (6 — 7 — 8). The requests and the edges which are left are shown in Figure 1 (right). We can see that there is no path for $r_2$, $r_4$, $r_6$ or $r_8$. So SGA accepts only 12 of the 16 requests and its approximation ratio is 4/3 on this instance.

### 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 $k \in \mathbb{N}$ be a parameter that is divisible by 3 and satisfies $3n/5 \leq k < n$. Let $V_v \cup V_a \cup V_b \cup V_c$ be the vertex set, where $V_v = \{v_1, \ldots, v_{2(n-k)}\}$, $V_a = \{a_1, \ldots, a_{2k/3}\}$, $V_b = \{b_1, \ldots, b_{2k/3}\}$ and $V_c = \{c_1, \ldots, c_{2k/3}\}$. The set of requests $\mathcal{R} = \mathcal{R}_v \cup \mathcal{R}_a \cup \mathcal{R}_b \cup \mathcal{R}_c$ is as follows: $\mathcal{R}_v$ contains $2n - i$ requests between $v_i$ and $v_{i+1}$ for all $i \in \{1, 3, 5, \ldots, 2(n-k)-1\}$, a total of $n^2 - k^2$ requests. $\mathcal{R}_a$ contains $n - k + 1$ requests between $a_i$ and $a_{i+1}$ for all $i \in \{1, 3, 5, \ldots, 2k/3 - 1\}$, a total of $k(n-k+1)/3$ requests. $\mathcal{R}_b$ and $\mathcal{R}_c$ consist of $k(n-k+1)/3$ requests each and are constructed in the same way as $\mathcal{R}_a$.

First, observe that the optimal solution can accept all requests: For every $i \in \{1, 3, 5, \ldots, 2(n-k)-1\}$, the $2n-i$ requests $\{v_i, v_{i+1}\}$ are accepted by routing one of them along the direct edge $\{v_i, v_{i+1}\}$ and routing the other $2n-i-1$ along paths of length two via each of the $2n-2k-i-1$ vertices $v_{i+2}, v_{i+3}, \ldots, v_{2(n-k)}$ and each of the $2k$ vertices in $V_a \cup V_b \cup V_c$.

One copy of each of the requests in $\mathcal{R}_a \cup \mathcal{R}_b \cup \mathcal{R}_c$ is accepted along the direct edge. The remaining copies of requests of $\mathcal{R}_a$ can be accepted along paths of length two via intermediate vertices in $V_b$, those of $\mathcal{R}_b$ along paths of length two via intermediate vertices in $V_c$, and those of $\mathcal{R}_c$ along paths of length two via intermediate vertices in $V_a$. Note that there are $n - k$ remaining copies of each request in $\mathcal{R}_a \cup \mathcal{R}_b \cup \mathcal{R}_c$ and $2k/3$ vertices in each of $V_a, V_b, V_c$. By our choice of

$k$ with $k \geq 3n/5$, we have $n-k \leq 2k/3$, and the number of available candidates for intermediate vertices is indeed sufficient.

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 $i \in \{1,3,\ldots,2(n-k)-1\}$, one between $a_i$ and $a_{i+1}$ for $i \in \{1,3,\ldots,2k/3-1\}$, one between $b_i$ and $b_{i+1}$ for $i \in \{1,3,\ldots,2k/3-1\}$, and one between $c_i$ and $c_{i+1}$ for $i \in \{1,3,\ldots,2k/3-1\}$. Then the algorithm accepts requests with a shortest path of length 2. Consider the remaining $n-k$ requests between $a_1$ and $a_2$. Assume that they are 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 $j \in \{3,5,\ldots,2k/3-1\}$, assume that the $n-k$ requests between $a_j$ and $a_{j+1}$ are accepted along paths $(a_j \text{ — } v_i \text{ — } a_{j+1})$ where $i \in \{1,3,\ldots,2(n-k)-1\}$. Thus, all of the $k(n-k+1)/3$ requests in $\mathcal{R}_a$ are accepted. Similarly, assume that all of the $2k(n-k)/3$ remaining requests in $\mathcal{R}_b \cup \mathcal{R}_c$ are accepted by using $v_1, v_3, v_5, \ldots, v_{2(n-k)-1}$ as internal vertices on paths of length 2.

All requests in $\mathcal{R}_a \cup \mathcal{R}_b \cup \mathcal{R}_c$ have been accepted now, but all edges from $v_1, v_3, v_5, \ldots, v_{2(n-k)-1}$ to vertices in $V_a \cup V_b \cup V_c$ have already been used by accepted paths. Therefore, the vertices in $V_a \cup V_b \cup V_c$ cannot be used as intermediate vertices on paths for requests in $\mathcal{R}_v$.

Consider the requests in $\mathcal{R}_v$. The remaining $2k$ requests between $v_{2(n-k)-1}$ and $v_{2(n-k)}$ can only be accepted via the vertices of $V_v \setminus \{v_{2(n-k)-1}, v_{2(n-k)}\}$. Since $|V_v \setminus \{v_{2(n-k)-1}, v_{2(n-k)}\}| = 2(n-k-1)$, SGA 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 $v_{2(n-k)}$, SGA has accepted only $2(n-k-1)+1$ requests. Next, consider the $2k+2$ requests between $v_{2(n-k)-3}$ and $v_{2(n-k)-2}$. They can be accepted only via 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, SGA can accept only $2(n-k-2)$ requests out of $2k+2$. In total, among the $2k+3$ requests between $v_{2(n-k)-3}$ and $v_{2(n-k)-2}$, SGA accepts only $2(n-k-2)+1$ requests. Continuing in this way from high to low indices, SGA accepts only $i$ requests among $2n-i$ requests between $v_i$ and $v_{i+1}$, for $i \in \{1,3,\ldots,2(n-k)-1\}$. Then the algorithm terminates since all edges in the subgraph induced by $V_v$ are exhausted. So SGA accepts only $n^2-kn+k$ requests in total.

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

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

With $\alpha = 1-\epsilon/2$, 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.

MEDP in complete graphs is NP-hard, but it is not known whether the problem 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 [3, 6].) So the inapproximability of the problem should be considered as well as better approximation algorithms.

*Acknowledgements* The authors are grateful to anonymous referees for comments on the earlier version.

## References

1. P. Carmi, T. Erlebach and Y. Okamoto, Greedy edge-disjoint paths in complete graphs. TIK-Report 155, Computer Engineering and Networks Laboratory (TIK), ETH Zürich, February 2003.
2. 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.
3. T. Erlebach, Approximation algorithms and complexity results for path problems in trees of rings. TIK-Report 109, Computer Engineering and Networks Laboratory (TIK), ETH Zürich, June 2001.
4. 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.
5. 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.
6. N. Garg, V.V. Vazirani and M. Yannakakis, Primal-dual approximation algorithms for integral flow and multicut in trees. *Algorithmica* **18**, 1997, 3–20.
7. 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.
8. J.M. Kleinberg, *Approximation algorithms for disjoint paths problems*. Ph.D. thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, 1996.
9. S.G. Kolliopoulos and C. Stein, Approximating disjoint-path problems using greedy algorithms and packing integer programs. In: Proceedings of the 6th Integer Programming and Combinatorial Optimization Conference IPCO VI, *Lecture Notes in Computer Science* **1412**, 1998, 153–168.
10. 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.