数理解析研究所講究録 1297 巻 2002 年 12-23

Greedy edge-disjoint paths in complete graphs [1]

(完全グラフ上の最大辺素パス問題に対する貪欲近似アルゴリズム)

Paz Carmi (Ben-Gurion University of the Negev)
Thomas Erlebach (ETH Zurich)
岡本 吉央 (Yoshio Okamoto) (ETH Zurich)

概要

最大辺素パス問題 (MEDP) は最も古くから知られる NP 困難問題の中の一つである．ここでは，完全グラフ上の MEDP について考察する．Erlebach and Vukadinović (2001) は完全グラフ上の MEDP が NP 困難であり，近似比が定数となる近似アルゴリズムを持つことを示した．最近の Kolman and Scheideler (2002) の結果は Erlebach and Vukadinović のものよりもよい近似比のアルゴリズムを与えている．実際，彼らは (最大辺素パス問題の一般化である) 分割不可能フロー問題に対して「短縮補題」を与え，この補題をうまく用いることでより良い近似比を達成している．本稿では，最短路優先貪欲アルゴリズムの近似比について議論する．まず，「短縮補題」がより良い上界を与えることを観察し，最短路優先貪欲アルゴリズムが 9 近似アルゴリズムであることを示す．次に，近似比の下界を与えるインスタンスを構成する．このインスタンスにより，最短路優先貪欲アルゴリズムの近似比が 3 よりも良くならないことが分かる．

1 イントロダクション

最大辺素パス問題 (maximum edge-disjoint path problem, MEDP) は最も古くから知られる NP 困難問題の中の一つである [3]．MEDP はネットワーク上での通信を効率良く行なう問題の幾つかを単純化したモデルであり，応用上も重要な最適化問題である．

MEDP の定義を述べる．MEDP のインスタンスは無向グラフ $G = (V, E)$ と V の頂点の非順序対の多重集合 $\mathcal{R}$ の対 MEDP$(G, \mathcal{R})$ で与えられる．$\mathcal{R}$ の各要素は**リクエスト** (request) と呼ばれる．考察することは，共通の辺を持たないパスを用いてより多くのリクエストを結ぶことである．互いに共通の辺を持たない複数のパスを**辺素パス** (edge-disjoint paths) と呼ぶ．MEDP の許容解は $\mathcal{R}$ の部分多重集合 $\mathcal{A}$ と G の辺素パスの集合 $\mathcal{P}$ の対 $(\mathcal{A}, \mathcal{P})$ で，$\mathcal{A}$ に属する各リクエストが $\mathcal{P}$ に属するあるパスで結ばれているようなものとする．目的は $\mathcal{A}$ のサイズ ($|\mathcal{A}|$ で書き表す) を最大にすることである．許容解 $(\mathcal{A}, \mathcal{P})$ においては $|\mathcal{A}| = |\mathcal{P}|$ が成り立つことに注意する．リクエストが**受理** (accepted) されているとは，それが $\mathcal{A}$ の要素であることである．一般性を失わずに，$\mathcal{R}$ の各リクエストに対してそれを結ぶ G のパスが存在すると仮定する．(そうでない場合は，そのリクエストを $\mathcal{R}$ から取り除いてしまえばよい．)

MEDP は入力のグラフが完全グラフであっても NP 困難であることが知られている [2]．よって，我々の興味は MEDP に対する近似アルゴリズムに向くことになる．MEDP に対するアルゴリズムが **ρ 近似アルゴリズム** (ρ-approximation algorithm) であるとは，これが多項式時間アルゴリズムであり，任意のインスタンスに対して $\mathrm{Opt} \leq \rho|\mathcal{A}|$ を満たす許容解 $(\mathcal{A}, \mathcal{P})$ を出力することである．ここで，Opt は最適解が受理するリクエストの数である．この ρ のことを近似アルゴリズムの**近似比** (approximation ratio) と呼ぶ．

表 1 に完全グラフ上の MEDP に対する近似アルゴリズムをまとめた．近似比が定数である最初の近似アルゴリズムは Erlebach and Vukadinović [2] によって与えられた．このアルゴリズムでは，最初に頂点集合 V を V_1, V_2, V_3 という 3 つの概ね等しいサイズの部分に分け，リクエスト $\mathcal{R}$ も $\mathcal{R}_{12}, \mathcal{R}_{23}, \mathcal{R}_{31}$ という 3 つの部分に分ける．ここで，各 $\mathcal{R}_{ij}$ は両方とも V_i に属しているか，あるいは片方は V_i にもう片方は V_j に属しているようなリクエストの多重集合である．そして，それぞれの $\mathcal{R}_{ij}$ に対してある手続きを用いて独

[1] Supported by the Joint Berlin/Zürich Graduate Program "Combinatorics, Geometry, and Computation" (CGC) financed by ETH Zürich and the German Science Foundation (DFG).

表 1: 完全グラフ上の MEDP に対する近似アルゴリズム.

近似比	アルゴリズム	著者 [文献]
27	三分割	Erlebach and Vukadinović [2]
17	BGA ($L=4$), SGA	Kolman and Scheideler [8]
9	BGA ($L=4$), SGA	本稿

立に解を求め，3 つの中で一番良いものを出力する．このアルゴリズムを**三分割アルゴリズム** (tripartition algorithm) と呼ぶことにするが，ここでは詳細な記述は省略する．重要なこととして，彼らはこのアルゴリズムが 27 近似アルゴリズムであることを示している．

MEDP に対する単純なアルゴリズムとして貪欲アルゴリズムがある．MEDP に対しては幾つかの変種がある．まず，我々が見る最初の貪欲アルゴリズムは**有界長貪欲アルゴリズム** (bounded-length greedy algorith, BGA) である．このアルゴリズムを図 1 に示しておく．ここで，ステップ 2-2 にある length(π_i) はパス π_i の長さ，つまり，π_i の辺の数を表している．BGA を最初に導入したのは Kleinberg [7] である．BGA はパラメータ L を入力で要求していることに注意する．

アルゴリズム：	有界長貪欲アルゴリズム (BGA)
入力：	無向グラフ $G=(V,E)$, リクエストの多重集合 $\mathcal{R}=\{\{s_1,t_1\},\ldots,\{s_k,t_k\}\}$
	パラメータ L ;
出力：	$\mathcal{A}\subseteq\mathcal{R}$ と G の辺素パスの集合 $\mathcal{P}$ で
	$\mathcal{A}$ の各リクエストが $\mathcal{P}$ のパスで結ばれているもの ;
ステップ 1：	(初期化)
	$\mathcal{A}\leftarrow\emptyset$; $\mathcal{P}\leftarrow\emptyset$; $i\leftarrow 1$;
ステップ 2：	(主ループ)
	while $i\leq k$
ステップ 2-1：	s_i と t_i を結ぶ最短路 π_i を見つける ;
ステップ 2-2：	if length(π_i) $\leq L$ then
ステップ 2-2-1：	$\mathcal{A}\leftarrow\mathcal{A}\cup\{\{s_i,t_i\}\}$; $\mathcal{P}\leftarrow\mathcal{P}\cup\{\pi_i\}$;
ステップ 2-2-2：	G から π_i のすべての辺を取り除く ;
ステップ 2-3：	$i\leftarrow i+1$;
	end of while;
ステップ 3：	(出力)
	return $\mathcal{A}$ および $\mathcal{P}$.

図 1: MEDP に対する有界長貪欲アルゴリズム.

Kolman and Scheideler [8] はフロー数 (flow number) と呼ばれるネットワークに対する新しいパラメータ F を導入し，パラメータ $L=4F$ の BGA が分割不可能フロー問題 (unsplittable flow problem, MEDP の一般化) に対する $(16F+1)$ 近似アルゴリズムであることを示した．実際，彼らは「短縮補題 (shortening lemma)」を示し，この補題をうまく用いることでより良い近似比を達成している．後程，完全グラフのフロー数が 1 であることを示す．すなわち，彼らの結果から完全グラフ上の MEDP に対する 17 近似アルゴリズムが出来る．

他の種類の貪欲アルゴリズムとして，**最短路優先貪欲アルゴリズム** (shotest-path-first greedy algorithm, SGA) がある．図 2 にアルゴリズムの詳細を示しておく．このアルゴリズムでは，ステップ 2-1 で選ぶ候

補が複数あるとき，その中のどれか一つを任意に選ぶものとする．ステップ2-4も同様に実行するとする．このアルゴリズムを最初に導入したのはKolliopoulos and Stein [6]である．

アルゴリズム：	最短路優先アルゴリズム (SGA)
入力：	無向グラフ $G=(V,E)$ とリクエストの多重集合 $\mathcal{R}$;
出力：	$\mathcal{A}\subseteq\mathcal{R}$ と G の辺素パスの集合 $\mathcal{P}$ で
	$\mathcal{A}$ の各リクエストが $\mathcal{P}$ のあるパスで結ばれているもの：
ステップ1：	(初期化)
	$\mathcal{A}\leftarrow\emptyset$; $i\leftarrow 0$;
ステップ2：	(主ループ)
	while $\mathcal{R}$ のリクエストの中に G のパスで結ばれているものがある
ステップ2-1：	$\{s_i,t_i\}\leftarrow\mathcal{R}$ のリクエストで s_i と t_i を結ぶ最短路の長さが
	$\mathcal{R}$ の他のリクエストを結ぶ最短路の長さを越えないもの ;
ステップ2-2：	$\mathcal{A}\leftarrow\mathcal{A}\cup\{\{s_i,t_i\}\}$;
ステップ2-3：	$\mathcal{R}\leftarrow\mathcal{R}\setminus\{\{s_i,t_i\}\}$;
ステップ2-4：	$\pi_i\leftarrow s_i$ と t_i を結ぶ G 上の最短路 ;
ステップ2-5：	π_i の全ての辺を G から取り除く ;
ステップ2-6：	$i\leftarrow i+1$;
	end of while ;
ステップ3：	(出力)
	return $\mathcal{A}$ および $\mathcal{P}\leftarrow\{\pi_i:\{s_i,t_i\}\in\mathcal{A}\}$.

図 2: MEDP に対する最短路優先貪欲アルゴリズム.

本稿では，短縮補題を用いることで，SGAが9近似アルゴリズムであることを示す．特筆しておきたいこととして，Erlebach and Vukadinović [2] はSGAが54近似アルゴリズムであることを示している．

更に，別の上界を得るためにリクエストの最大多重度 ($\mu_{\max}$ で表記する) を考え，$\mu_{\max}$ もSGAの近似比の上界であることを示す．よって，次の定理を得ることになる．

定理 1.1 (主結果). 最短路優先貪欲アルゴリズムは完全グラフ上の最大辺素パス問題に対する $\min\{9,\mu_{\max}\}$ 近似アルゴリズムである.

次のステップとして，近似比の下界を考察する．本稿では，完全グラフ上のMEDPのインスタンスでSGAの近似比が3より良くならないものを構成する．

本稿の構成は以下の通りである．より良い上界 (定理1.1) の証明は次の節で行なう．第3節で下界を与えるインスタンスを構成する．最後の節で補足を幾つか加える．

記法 有限 (多重) 集合 S のサイズ，すなわち S の要素数を $|S|$ で表す．実数全体の集合を $\mathbb{R}$ で表し，非負実数全体の集合を $\mathbb{R}_+$ で表す．パス p をそのパスに沿って順に辿っていった頂点 $v_1,v_2,\ldots,v_l$ を使って，$(v_1 — v_2 — \cdots — v_l)$ のように書き表すことがある．パス p の長さ，すなわち，p の辺の数を $\mathrm{length}(p)$ で表す．二つのパス p と q に対して，$p\cap q$ で p と q が共通に持つ辺の集合をあらわす．よって，$p\cap q=\emptyset$ というのは，p と q が辺素であることを意味する．パス p と辺 e に対して，e が p の辺であることを $e\in p$ で表す．最後に，「$A:=B$」で「A を B で定義する」ことを表す.

2 上界

本節では，完全グラフ上のMEDPの近似比の上界を改良する．最初の小節では，Kolman and Scheideler [8] の枠組の中から特にフロー数と短縮補題を説明し，彼らの結果から直ちに近似比が改良できることを見る．すなわち，パラメータ $L = 4$ のBGAが完全グラフ上のMEDPに対する17近似アルゴリズムであることを示す．二つ目の小節では，更に改良することを目指し，完全グラフ上のMEDPに対してSGAが9近似アルゴリズムであることを示し，主定理 (定理1.1) を証明する．

2.1 完全グラフのフロー数

Kolman and Scheideler [8] の最近の研究では，ネットワークのフロー数 (flow number) と短縮補題 (shortening lemma) を用いて，**分割不可能フロー問題** (unsplittable flow problem, UFP) に対する近似アルゴリズムの近似比を改良している．ここでは，彼らの結果の紹介から始める．

UFPはMEDPの一般化である．まず，与えられているのは無向グラフ $G = (V, E)$ と関数 $u : E \to \mathbb{R}_+$ である．辺 $e \in E$ に対して，$u(e)$ は e の**容量** (capacity) と呼ばれる．また，MEDPと同様にリクエストの多重集合 $\mathcal{R}$ も与えられる．更に，$d : \mathcal{R} \to \mathbb{R}_+$ と $r : \mathcal{R} \to \mathbb{R}_+$ という二つの関数も与えられる．リクエスト $i \in \mathcal{R}$ に対して，$d(i)$ と $r(i)$ はそれぞれ i の**需要** (demand) と**利益** (profit, reward) と呼ばれる．まとめると，UFPのインスタンスは5つ組 $\mathrm{UFP}(G, \mathcal{R}, u, d, r)$ で与えられ，G は無向グラフ，$\mathcal{R}$ はリクエストの多重集合，u は容量，d は需要，r は利益である．UFPの許容解は $\mathcal{R}$ の部分多重集合 $\mathcal{A}$ と G の (必ずしも辺素とは限らない) パスの集合 $\mathcal{P}$ の対 $(\mathcal{A}, \mathcal{P})$ で，$\mathcal{A}$ の各リクエストが $\mathcal{P}$ のあるパスで結ばれていて，かつ，それぞれの辺 $e \in E$ に対して，e を通るパスで結ばれたリクエストの需要の和が e の容量 $u(e)$ 以下であるようなものとする．リクエストが**受理** (accepted) されているとは，それが $\mathcal{A}$ に属していることである．目的は，受理されたリクエストの利益の総和を最大化することである．MEDPがUFPの特殊ケースであることは，すべての $e \in E$ に対して $u(e) = 1$，すべての $i \in \mathcal{R}$ に対して $d(i) = r(i) = 1$ とすることで分かる．

UFPがNP困難問題であることは，UFPがMEDPを特殊ケースとして含むことから分かる．ここでは，UFPに対する近似アルゴリズムを定義する．(先程の節では，MEDPに対する近似アルゴリズムしか定義していなかった．) UFPに対するアルゴリズムが ρ **近似アルゴリズム** (ρ-approximation algorithm) であるとは，これが多項式時間アルゴリズムであり，任意のインスタンスに対して $\mathrm{Opt} \leq \rho \sum_{i \in \mathcal{A}} r(i)$ を満たすような許容解 $(\mathcal{A}, \mathcal{P})$ を出力することである．ここで，Optは最適解が受理するリクエストの利益の総和である．MEDPのときと同様に ρ をこのアルゴリズムの**近似比** (approximation ratio) という．

多くの組合せ最適化問題と同様に，UFPは $\{0,1\}$ 整数計画問題として記述できる．この記述のために，二種類の変数を用意する．一つ目は $x \in \{0,1\}^{\mathcal{R}}$ で，リクエスト i が受理されるとき $x_i = 1$ で，そうでないとき $x_i = 0$ となるものとする．二つ目の変数は $y^{(i)} \in \{0,1\}^{\mathcal{P}_i}$ で，ここで，$i \in \mathcal{R}$ であり，$\mathcal{P}_i$ はリクエスト i を結ぶ G 上のパス全体の集合である．変数 $y^{(i)}$ の意味は，リクエスト i がパス $\pi \in \mathcal{P}_i$ を使って受理されるとき $y^{(i)}_\pi = 1$ で，そうでないとき $y^{(i)}_\pi = 0$ となることである．

UFP を整数計画問題として定式化したのが次である.

IP-UFP$(G,\mathcal{R},u,d,r)$:

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

この定式化 IP-UFP$(G,\mathcal{R},u,d,r)$ は理解しやすいが，変数の数が指数関数的になっている．しかし，ネットワークフロー問題のときと同様に，多項式サイズの定式化も得ることが出来る．例えば，[5] を参照.

では，$\{0,1\}$ 制約式 ((3) と (4)) を緩和，すなわち，これらの制約式を $x_i\in[0,1]$ と $y_\pi^{(i)}\in[0,1]$ に置き換えてみる．このようにして，**多品種フロー問題** (multicommodity flow problem) と呼ばれる線形計画問題が得られる．この問題を LP-UFP$(G,\mathcal{R},u,d,r)$ で表す.

多品種フロー問題の変種として，並行多品種フロー問題がある．**並行多品種フロー問題** (concurrent multicommodity flow problem) とは次のように定義される問題である.

ConMFP$(G,\mathcal{R},u,d,r)$:

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

並行多品種フロー問題 ConMFP$(G,\mathcal{R},u,d,r)$ の許容解では，すべての x_i の値が制約 (5) によって同じになっている．つまり，この問題は，すべてのリクエストが同じ割合だけ受理されることを強制されている場合にその割合をどれだけ大きくすることができるかを考える問題である．並行多品種フロー問題 ConMFP$(G,\mathcal{R},u,d,r)$ の許容解は対応する多品種フロー問題 LP-UFP$(G,\mathcal{R},u,d,r)$ の許容解にもなっていることに注意する．但し，逆は成り立つとは限らない.

では，ネットワークのフロー数を定義しよう．(ここで，**ネットワーク** (network) とは無向グラフ $G=(V,E)$ と容量関数 $u:E\to\mathbb{R}_+$ の組 (G,u) のことである．) 無向グラフ $G=(V,E)$ と容量関数 $u:E\to\mathbb{R}_+$ が与えられている．ここで，並行多品種フロー問題のインスタンス ConMFP(G,u) を構成する．リクエストの集合 $\mathcal{R}$ は頂点の全ての非順序対で成り立つ，つまり，$\mathcal{R}=\{\{s,t\}:s,t\in V,s\neq t\}$ とする．需要 $d:\mathcal{R}\to\mathbb{R}_+$ は

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

で定義し，利益 $r:\mathcal{R}\to\mathbb{R}_+$ は d と同じである，つまり，すべての $i\in\mathcal{R}$ に対して $d(i)=r(i)$ であると

する．このようにして，並行多品種フロー問題のインスタンス ConMFP(G,u) =ConMFP$(G,\mathcal{R},u,d,r)$ を得る．

並行多品種フロー問題の許容解 $\xi=(x,(y^{(i)}:i\in\mathcal{R}))$ に対して，緩慢度と過密度という二つのパラメータを導入する．許容解 ξ の**緩慢度** (dilation) $D(\xi)$ とは ξ の中のパスの最大長である，すなわち，

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

許容解 ξ の**過密度** (congestion) $C(\xi)$ とは ξ の目的関数値の逆数である，すなわち，

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

ネットワーク (G,u) の**フロー数** (flow number) とは次で定義される量 $F(G,u)$ のことである．

$$F(G,u):=\min\{\max\{D(\xi),C(\xi)\}:\xi\text{ は ConMFP}(G,u)\text{ の許容解 }\}.$$

ネットワークのフロー数は容量のスケーリングに対して不変であることに注意する．

ここで，単位容量の完全グラフに対して，そのフロー数を定める．

補題 2.1 (完全グラフのフロー数). $G=(V,E)$ を完全グラフ，$u:E\to\mathbb{R}_+$ をすべての $e\in E$ に対して $u(e)=1$ であるとする．このとき，$F(G,u)=1$ が成り立つ．

証明. まず，ConMFP(G,u) が何になるかを調べる．リクエストの集合 $\mathcal{R}$ は E であり，需要は各 $\{s,t\}\in\mathcal{R}$ に対して，$d(\{s,t\})=(n-1)/n$ で定められる．

ここで，$\xi=(x,(y^{(i)}i\in\mathcal{R}))$ を ConMFP(G,u) の許容解とする．このとき，$0\le x_1\le 1$ であるから，$\max\{D(\xi),C(\xi)\}\ge C(\xi)=1/x_1\ge 1$ となる．すなわち，次が成立する．

$$\begin{aligned}F(G,u)&=\min\{\max\{D(\xi),C(\xi)\}:\xi\text{ は ConMFP}(G,u)\text{ の許容解 }\}\\&\ge\min\{1:\xi\text{ は ConMFP}(G,u)\text{ の許容解 }\}=1.\end{aligned}$$

次に，逆向きの不等式を示す．ξ' を ConMFP(G,u) の許容解で，任意のリクエスト $i=\{s,t\}$ と任意のパス $\pi=(s\text{ — }t)\in\mathcal{P}_i$ に対して $y_\pi^{(i)}=x_i=1$ で他の全ての変数は 0 としたものとする．(実際，ξ' が ConMFP(G,u) の許容解であることは簡単に確認できる．) このとき，$D(\xi')=C(\xi')=1$ が成立するので，

$$\begin{aligned}F(G,u)&=\min\{\max\{D(\xi),C(\xi)\}:\xi\text{ は ConMFP}(G,u)\text{ の許容解 }\}\\&\le\max\{D(\xi'),C(\xi')\}=1\end{aligned}$$

となる．これで証明が完結した． □

フロー数を用いることで，Kolman and Scheideler [8] は興味深い定理を示している．一つは，いわゆる「短縮補題」である．

補題 2.2 (短縮補題 [8]). 並行多品種フロー問題のインスタンスで，需要と利益が一致し，かつ，フロー数が F であるようなものを I とする．このとき，目的関数値が f となる I の任意の許容解に対して，I の許容解 ξ で，目的関数値が $f/(1+\epsilon)$ で緩慢度が $D(\xi)\le 2(1+1/\epsilon)F$ であるものが存在する．

$\epsilon=1$ とする．UFP の許容解が与えられているときに，その許容解を，対応する多品種フロー問題の許容解で，目的関数値は元の解の半分だけども，使うパスの長さが $4F$ 以下であるものに変換できることが短縮補題を用いると分かる．どうしてそうなるかを説明する．UFP のインスタンス $I=$ IP-UFP$(G,\mathcal{R},u,d,d)$ が与えられていて，$(x,(y^{(i)}:i\in\mathcal{R}))$ を I に対する許容解とする．このときに，$\bar{\mathcal{R}}:=\{i\in\mathcal{R}:x_i=1\}$ と定め，並行多品種フローのインスタンスとして $\bar{I}=$ ConMFP$(G,\bar{\mathcal{R}},u,d,d)$ を考える．ここで，すべての $i\in\bar{\mathcal{R}}$ に対して $\bar{x}_i=x_i$，すべての $i\in\bar{\mathcal{R}}$ と $\pi\in\mathcal{P}_i$ に対して $\bar{y}_\pi^{(i)}=y_\pi^{(i)}$ というものを考えると，$(\bar{x},(\bar{y}^{(i)}:i\in\bar{\mathcal{R}}))$

は $\bar{I}$ の許容解で，目的関数値は 1 になる．今，短縮補題をこの解に適用する．そうすると，$\bar{I}$ の許容解で，目的関数値が 1/2，緩慢度が $4F$ 以下のものが存在することが分かる．この解は元の UFP のインスタンス I に対応する多品種フロー問題の許容解 LP-UFP$(G, \mathcal{R}, u, d, d)$ でもあり，目的関数値は元の解の値の半分で，緩慢度は $4F$ 以下になっていることになる．

この補題を用いて，Kolman and Scheideler [8] は別の面白い定理を示した．これはフロー数を使って近似比の上界を与えている．

補題 2.3 (フロー数を用いた UFP に対する上界 [8]). 分離不可能フロー問題のインスタンスで，次の三条件を満たすものを考える．(1) 任意のリクエストに対して要求と利益が等しい．(2) 要求の最大値が容量の最小値以下である．(3) フロー数が F である．このとき，適当に番号づけを変更すると，パラメータを $4F$ とする有界長貪欲アルゴリズムは上の条件を満たす分離不可能フロー問題に対する $(16F+1)$ 近似アルゴリズムになる．

この補題において，「適当に番号づけを変更する」ということはリクエストの番号を次のように変えることである：任意の二つのリクエスト i と j に対して，$r(i) > r(j)$ ならば i は j の前に来る．この補題は SGA に対しても成立することに注意する．

単位容量の完全グラフのフロー数は 1 (補題 2.1) であり，MEDP は補題 2.3 にある三条件を満足するので，次の結果が直ちに得られる．

定理 2.4 (完全グラフ上の MEDP に対する BGA の近似比). パラメータを $L = 4$ とする有界長貪欲アルゴリズムは完全グラフ上の最大辺素パス問題に対する 17 近似アルゴリズムである．

この結果も SGA に対して成立する．次の小節で，この上界を改善する．

2.2 9 近似アルゴリズム

ここで，完全グラフ上の MEDP に対して SGA が 9 近似アルゴリズムであることを示す．もう一度，MEDP は UFP の特殊ケースであることに注意しておく．

補題 2.5 (MEDP に対する上界の改善). 最短路優先貪欲アルゴリズムはフロー数 F の無向グラフ上の最大辺素パス問題に対する $(8F+1)$ 近似アルゴリズムである．

証明. $(\mathcal{A}, \mathcal{P}(\mathcal{A}))$ を SGA によって得られた許容解とする．つまり，$\mathcal{A}$ は SGA が受理したリクエストの多重集合で，$\mathcal{P}(\mathcal{A})$ はそれらを結ぶ辺素パスの集合である．また，$\mathcal{O}$ で最適解が受理するリクエストの多重集合を表す．短縮補題 (補題 2.2) によって，対応する多品種フロー問題の許容解 $\xi = (x, (y^{(i)} : i \in \mathcal{R}))$ で，目的関数値が $|\mathcal{O}|/2$ で，なおかつ，$D(\xi) \leq 4F$ となるものが存在することが分かる．

$\mathcal{P}(\xi)$ で $y_\pi^{(i)} > 0$ を満たすパス全体の集合を表すことにする．また，$\mathcal{P}_{\leq 4F}(\mathcal{A}) := \{p \in \mathcal{P}(\mathcal{A}) : \mathrm{length}(p) \leq 4F\}$ と定義し，任意にパス $p \in \mathcal{P}_{\leq 4F}(\mathcal{A})$ を取って来る．このとき，式 (1) を使うと，

$$\sum_{\substack{\pi \in \mathcal{P}(\xi) \\ \text{s.t. } \pi \cap p \neq \emptyset}} y_\pi^{(i)} \leq \sum_{e \in p} \sum_{\substack{\pi \in \mathcal{P}(\xi) \\ \text{s.t. } e \in \pi}} y_\pi^{(i)} = \sum_{e \in p} \sum_{\substack{i \in \mathcal{R}, \pi \in \mathcal{P} \\ \text{s.t. } e \in \pi}} y_\pi^{(i)} \leq \sum_{e \in p} 1 \leq \mathrm{length}(p) \tag{6}$$

が成立することが分かる．

今，$\mathcal{P}(\xi)$ を次のように二つの部分 $\mathcal{P}_1(\xi)$ と $\mathcal{P}_2(\xi)$ に分割する．

$$\mathcal{P}_1(\xi) = \{\pi \in \mathcal{P}(\xi) : \pi \text{ が結ぶリクエストが } \mathcal{A} \text{ の要素ではない }\},$$
$$\mathcal{P}_2(\xi) = \mathcal{P}(\xi) \setminus \mathcal{P}_1(\xi).$$

このとき，次の要請が成り立つことを確認する．

要請 2.5.1. それぞれのパス $\pi \in \mathcal{P}_1(\xi)$ に対してあるパス $p \in \mathcal{P}(\mathcal{A})$ が存在して $\pi \cap p \neq \emptyset$ となる，すなわち，π と p が共通の辺を持つ．更に，p の長さは $4F$ 以下である．

要請の証明．最初の部分については，そうでないと仮定して考える．すると，$\mathcal{P}(\mathcal{A})$ のどのパスも使っていない辺だけから成り立つパスがあるリクエスト $r \in \mathcal{R} \setminus \mathcal{A}$ を結んでしまうことになる．そうなると，SGA はこのリクエストを受理しないといけなかったことになり，矛盾．

二つ目の部分については，もしそのようなすべてのパスの長さが $4F$ より長いとすると，SGA は p が結ぶリクエストを受理する代わりに π が結ぶリクエストを受理しないといけなかったことになり，また矛盾となる． □

では，近似比を解析しよう．まず，要請 2.5.1 と式 (6) を用いて，

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

を得る．次に，

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

が成立することが分かる．ゆえに，

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

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

となり，近似比が $8F+1$ になることが示された． □

この解析はパラメータを $L = 4F$ とする BGA にも適用できることを補足しておく．

補題 2.1 と 2.5 から直ちに次の系を得る．

系 2.6 (完全グラフ上の MEDP に対する上界の改善). 最短路優先貪欲アルゴリズムは完全グラフ上の最大辺素パス問題に対する 9 近似アルゴリズムである．

別の上界を得るために，リクエストの多重度を定義する．リクエストの多重集合 $\mathcal{R}$ を考える．リクエスト $\{s,t\} \in \mathcal{R}$ に対して，$\mathcal{R}$ が $\{s,t\}$ を μ 個持っているとき，$\mathcal{R}$ における $\{s,t\}$ の**多重度** (multiplicity) は μ である，と言うことにする．例えば，$\mathcal{R} = \{\{1,2\},\{1,2\},\{2,4\},\{3,4\},\{3,4\},\{3,4\}\}$ とすると，$\{1,2\}$ の多重度は 2，$\{2,4\}$ の多重度は 1，$\{3,4\}$ の多重度は 3 となることが分かる．$\mu_{\max}(\mathcal{R})$ で $\mathcal{R}$ におけるリクエストの多重度の最大値を表すことにする．混乱の危険がない場合には，単に $\mu_{\max}$ と書く．先程の例では，$\mu_{\max} = 3$ になる．

ここで，$\mu_{\max}$ が小さいときには近似比が少し良くなることを見る．$\mathcal{R}_1$ を $\mathcal{R}$ の多重度を無視した集合であるとする．先程の例では，$\mathcal{R}_1 = \{\{1,2\},\{2,4\},\{3,4\}\}$ となる．

補題 2.7 (SGA の出力の性質). 完全グラフ上の最大辺素パス問題に対して，$\mathcal{R}$ がリクエストの多重集合であるとき，最短路優先貪欲アルゴリズムが受理するリクエストの多重集合 $\mathcal{A}$ は次の条件を満たす．

(*) $\mathcal{R}_1$ の各リクエストは長さ 1 のパス (つまり，ただの 1 辺) で結ばれて受理される．

更に，条件 (*) を満たす最適解が存在する．

証明．容易なので省略する． □

つまり，$|\mathcal{R}_1| \leq |\mathcal{A}|$ が成立する．更に，$\mu_{\max}|\mathcal{R}_1|$ は $|\mathcal{R}|$ の上界であり，$|\mathcal{R}|$ は Opt の上界である．よって，近似比が次のように解析できる．

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

この解析と系 2.6 を合わせることで，定理 1.1 が得られる．

3 下界

前節では，SGA が $\min\{9, \mu_{\max}\}$ 近似アルゴリズムになることを示した．では，この近似比はどこまで良くすることが出来るのだろうか．つまり，本当の近似比は何なのだろうか．この値以下に近似が下げられないことを示すインスタンスはまだ見つかっていない．ただ，近似比が 9 より小さくならないインスタンスではリクエストの最大多重度が 9 以上でないといけないことに注意しておく．

この節では，完全グラフ上の MEDP のインスタンスを二つ構成する．最初のものは，$\mu_{\max} = 2$ で近似比が 4/3 になる．次のものは，$\mu_{\max}$ は任意に大きく，小さい $\epsilon > 0$ に対して $3 - \epsilon$ という近似比になる．

3.1 $\mu_{\max} = 2$ のインスタンス

この小節では，完全グラフ上の MEDP のインスタンスで，$\mu_{\max} = 2$ になっているものを構成して，$\mu_{\max} = 2$ のときに SGA の近似比が 4/3 より良くならないことを示す．ここで構成するインスタンスは頂点数 8 のもので，リクエストの数は 16 である．最適解ではすべてのリクエストを受理するが，SGA は 12 個しか受理しない可能性がある．よって，その比が 4/3 になる．

頂点集合を $V = \{1,2,3,4,5,6,7,8\}$ とする．16 個のリクエストは次のように与えられる．

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

多重集合 $\mathcal{R}$ はこの 16 個のリクエストから成り立っていて，$\mu_{\max} = 2$ であることが分かると思う．図 3 に，16 個のリクエストが頂点数 8 の完全グラフ上にどのように分布しているか描いてある．ここで，それぞれのリクエスト r_i は $\{s_i, t_i\}$ として記述されていて，頂点は一番上が 1 で反時計回り順に番号を付けてある．

補題 2.7 より，SGA の出力と最適解では，$\mathcal{R}_1$ の各リクエストを対応する 1 辺で受理する．このインスタンスにおいては，$\mathcal{R}_1 = \{r_1, r_3, r_5, r_7, r_9, r_{11}, r_{13}, r_{14}, r_{15}, r_{16}\}$ である．よって，まず，これらのリクエストを $\mathcal{R}$ から取り除いて，使用した辺もグラフから取り除くことにする．

では，最適解が残りのリクエストをすべて受理することを見よう．そのためには，最適解は次のようなパスを使えばよい．

リクエスト r_2 をパス (1 — 2 — 3) で結ぶ．
リクエスト r_4 をパス (5 — 4 — 3) で結ぶ．
リクエスト r_6 をパス (7 — 8 — 1) で結ぶ．
リクエスト r_8 をパス (5 — 6 — 7) で結ぶ．
リクエスト r_{10} をパス (2 — 8 — 4) で結ぶ．
リクエスト r_{12} をパス (6 — 1 — 5 — 8) で結ぶ．

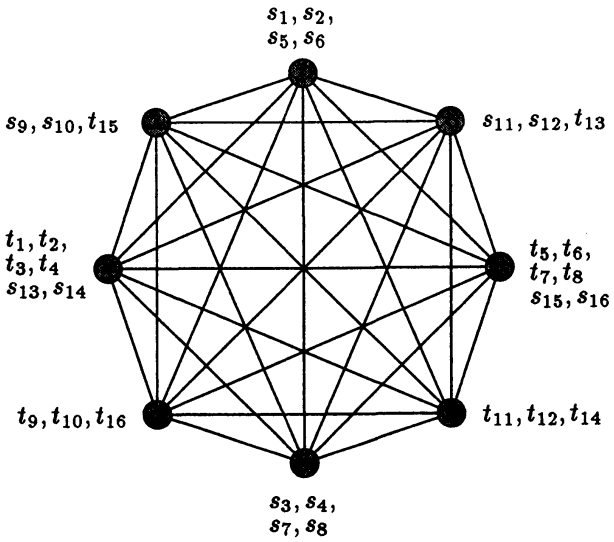

図 3: 頂点数 8 で $\mu_{\max} = 2$ のインスタンス.

次に，SGA が受理するリクエストを考える．これを考えるために，まず，まだ受理されていないリクエストを結ぶ最短路の長さを計算する．

r_2 に対して，(1 — 2 — 3) が最短路で長さは 2.
r_4 に対して，(5 — 4 — 3) が最短路で長さは 2.
r_6 に対して，(1 — 8 — 7) が最短路で長さは 2.
r_8 に対して，(5 — 6 — 7) が最短路で長さは 2.
r_{10} に対して，(2 — 3 — 4) が最短路で長さは 2.
r_{12} に対して，(8 — 7 — 6) が最短路で長さは 2.

これらのどのリクエストも SGA が次に受理する候補なので，ここでは r_{10} を選ぶことにする．そして，リクエスト r_{10} と 2 辺 $\{2,3\}$, $\{3,4\}$ を取り除く．すると，r_2 と r_4 に対する最短路が変化する．

r_2 に対して，(1 — 6 — 7 — 3) が最短路で長さは 3.
r_4 に対して，(5 — 6 — 7 — 3) が最短路で長さは 3.

よって，次の時点では r_6 か r_8 か r_{12} を選べる．ここでは r_{12} を選び，r_{12} と 2 辺 $\{6,7\}$, $\{7,8\}$ を取り除く．すると，残った r_2, r_4, r_6, r_8 のためのパスはもうないことが分かる．このようにして，SGA が受理したリクエストの数が 12 となる．

3.2 3 近似よりは良くならないインスタンス

この小節では，完全グラフ上の MEDP のインスタンスの列を構成する．このインスタンスの列に対する SGA の近似比は小さな $\epsilon > 0$ に対して $3 - \epsilon$ となり，漸近的に 3 である．

大きな n に対して，頂点数 $2n$ の完全グラフを考える．その頂点集合は $V_v \cup V_a \cup V_b \cup V_c$ であり，3 で割り切れて n よりも少し小さな数 k を用いて，$V_v = \{v_1, \ldots, v_{2(n-k)}\}$, $V_a = \{a_1, \ldots, a_{2k/3}\}$, $V_b = \{b_1, \ldots, b_{2k/3}\}$, $V_c = \{c_1, \ldots, c_{2k/3}\}$ とする．(より明示的に書くと，k が満たすべき条件は $3n/5 \leq k \leq n$ という不等式を満たすことと 3 で割り切れることである．) リクエストの多重集合 $\mathcal{A} = \mathcal{A}_v \cup \mathcal{A}_a \cup \mathcal{A}_b \cup \mathcal{A}_c$ は次のように与えられる．$\mathcal{A}_v$ のサイズは $n^2 - k^2$ で，すべての $i \in \{1, 3, 5, \ldots, 2(n-k)-1\}$ に対して，2 頂点 v_i と v_{i+1} の間に $2n - i$ 個のリクエストを持つ．$\mathcal{A}_a$ のサイズは $k(n-k+1)/3$ で，すべての $i \in \{1, 3, 5, \ldots, 2k/3-1\}$

に対して，2頂点 a_i と a_{i+1} の間に $n-k+1$ 個のリクエストを持つ．$\mathcal{A}_b$ と $\mathcal{A}_c$ のサイズも $k(n-k+1)/3$ で，$\mathcal{A}_a$ と同様に構成される．図4がこの状況を描いている．

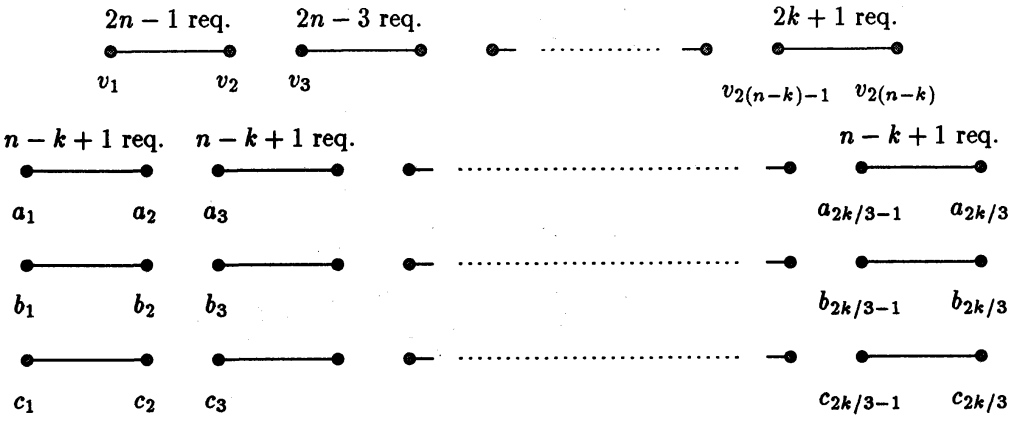

図 4: 最短路優先貪欲アルゴリズムに対する悪いインスタンス．

まず，最適解がすべてのリクエストを受理することを確かめる．どのように受理するのかを見よう．最初に，$\mathcal{A}_v$ のリクエストを次のようにしてすべて受理する．まず，v_1 と v_2 の間の $2n-1$ 個のリクエストを考える．その中の1つを $(v_1 — v_2)$ を用いて受理する．その他の $2n-2$ 個のリクエストは2辺から成るパス $(v_1 — u — v_2)$ を用いて受理する．ここで，$u \in (V_v \setminus \{v_1, v_2\}) \cup V_a \cup V_b \cup V_c$ である．これで，v_1 と v_2 の間の $2n-1$ 個のリクエストをすべて受理できた．次に，v_3 と v_4 の間の $2n-3$ 個のリクエストを考える．その中の1つを $(v_3 — v_4)$ を用いて受理する．その他の $2n-4$ 個のリクエストは2辺から成るパス $(v_3 — u — v_4)$ で受理する．ここで，$u \in (V_v \setminus \{v_1, v_2, v_3, v_4\}) \cup V_a \cup V_b \cup V_c$ である．これで，v_3 と v_4 の間の $2n-3$ 個のリクエストをすべて受理できた．このようにして，v_1，v_2，v_3，v_4，v_5，...と続けていく．一般的に，v_{2j-1} と v_{2j} の間の $2n-2j+1$ 個のリクエストを考えるとき，その中の1つは辺 $(v_{2j-1} — v_{2j})$ を用いて受理し，その他の $2n-2j$ 個のリクエストは2辺から成るパス $(v_{2j-1} — u — v_{2j})$ を用いて受理する．ここで，$u \in (V_v \setminus \{v_1, \ldots, v_{2j}\}) \cup V_a \cup V_b \cup V_c$ である．こうすれば，すべての $j = 1, \ldots, n-k$ において，v_{2j-1} と v_{2j} の間にある $2n-2j+1$ 個のリクエストが受理できる．

2番目に，$\mathcal{A}_a$ のリクエストを次のように受理する．任意の $i \in \{1, 3, \ldots, 2k/3-1\}$ に対して，a_i と a_{i+1} の間にある $n-k+1$ 個のリクエストの中で，1つは1辺 $(a_i — a_{i+1})$ で受理し，その他の $n-k$ 個は長さ2のパス $(a_i — u — a_{i+1})$ で受理する．但し，$u \in V_b$ とする．ここで，$k \geq 3n/5$ という条件から，すべてのリクエストが受理できることが分かる．同様に，$\mathcal{A}_b$ と $\mathcal{A}_c$ のリクエストはそれぞれ V_c と V_a を用いてすべて受理できることが分かる．ゆえに，最適解は $n^2 + kn + k - k^2$ 個のリクエストすべてを受理する．

では，SGAの出力を考えよう．SGAはまず長さ1の最短路で n 個のリクエストを受理する．次に，長さ2の最短路で受理し始めるが，まず a_1 と a_2 の間に残っている $n-k$ 個のリクエストを考える．これらのリクエストを $(a_1 — v_i — a_2)$ というパス (但し，$i \in \{1, 3, \ldots, 2(n-k)-1\}$) で受理する．同様にして，$a_j$ と a_{j+1} の間にある $n-k$ 個のリクエストを $(a_j — v_i — a_{j+1})$ というパス (但し，$i \in \{1, 3, \ldots, 2(n-k)-1\}$) で受理する．このようにして，$\mathcal{A}_a$ のリクエスト $k(n-k+1)/3$ 個すべてをSGAは受理した．同様にして，SGAは $\mathcal{A}_b$ と $\mathcal{A}_c$ のリクエストも $v_1, v_3, v_5, \ldots, v_{2(n-k)-1}$ を長さ2のパスの中間点として用いることですべて受理できる．

では，$\mathcal{A}_v$ に残っているリクエストを受理することを考える．$v_{2(n-k)-1}$ と $v_{2(n-k)}$ の間に残っている $2k$ 個のリクエストを受理するには $V_v \setminus \{v_{2(n-k)-1}, v_{2(n-k)}\}$ の頂点を通るパスを使わなくてはならない．しかし，$|V_v \setminus \{v_{2(n-k)-1}, v_{2(n-k)}\}| = 2(n-k-1)$ なので，受理できるリクエストの数は $2k$ 個中 $2(n-k-1)$ 個だけである．合計して，$v_{2(n-k)-1}$ と $v_{2(n-k)}$ の間の $2k+1$ 個のリクエストの中の $2(n-k-1)+1$ 個だけが受理できたことになる．次に，$v_{2(n-k)-3}$ と $v_{2(n-k)-2}$ の間 $2k+2$ 個のリクエストを見てみる．こ

れらのリクエストを受理するには $V_v \setminus \{v_{2(n-k)-3}, v_{2(n-k)-2}, v_{2(n-k)-1}, v_{2(n-k)}\}$ の頂点を通るパスを使わなくてはならない．すると，先と同じ理由により，$2k+2$ 個中 $2(n-k-2)$ 個しか受理できないことになる．合計して，$v_{2(n-k)-3}$ と $v_{2(n-k)-2}$ 間の $2k+3$ 個のリクエストの中から $2(n-k-2)+1$ 個だけが受理できたことになる．このようにして，右から左に順番に行くと，任意の $i \in \{1,3,\ldots,2(n-k)-1\}$ に対して，SGA は v_i と v_{i+1} の間の $2n-i$ 個のリクエストの中の i 個しか受理できないことになる．そして，これだけ受理し終わると，もうそれ以上受理できなくなる．よって，SGA は n^2-kn+k 個のリクエストを受理した，ということが分かった．

$3/5 \leq \alpha < 1$ を満たす α 用いて，$k=\alpha n$ とおく．すると，近似比は次のようになる．

$$\frac{n^2+kn+k-2k^2}{n^2-kn+k} = \frac{n^2+\alpha n^2+\alpha n-2\alpha^2 n^2}{n^2-\alpha n^2+\alpha n} \xrightarrow{n\to\infty} \frac{1+\alpha-2\alpha^2}{1-\alpha} = 1+2\alpha \xrightarrow{\alpha\to 1} 3.$$

4 補足

完全グラフ上の MEDP に対して SGA が 9 近似アルゴリズムであり，また 3 近似よりは良くなれないことを示した．つまり，まだギャップがあることになる．また，完全グラフ上の MEDP に対する多項式時間近似スキーム (polynomial-time approximation scheme) が存在する可能性さえあることも注意しておきたい．実際，完全グラフ上の MEDP が APX 困難であるかどうかも知られていない．(一般の無向グラフ上の MEDP は APX 困難であることが知られている [1, 4]．) より良いアルゴリズムと同じように近似不可能性の方も今後の課題となる．

参考文献

[1] 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.

[2] 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.

[3] 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.

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

[5] V. Guruswami, S. Khanna, R. Rajaraman, F.B. Shepherd and M. Yannakakis, Near-optimal hardness results and approximation algorithms for edgedisjoint paths and related problems. In: *Proceedings of the 31st Annual ACM Syposium on Theory of Computing*, 1999, 19–28.

[6] 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.

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