2004－AL－98　(7)
2004／11／5

合流フローの混雑最小化アルゴリズムに対する実験的性能評価と貪欲改善

中央大学大学院 理工学研究科 情報工学専攻

二瓶 浩一　　浅野 孝夫

概要　ネットワーク・フローの中で,「すべての点においてフローが出て行く辺が 1 本以下」という条件を満たすものを合流フローという．インターネット・ルーティングをはじめとして，現実社会において合流フローとなる例は数多く存在する．入力として，有向グラフ・複数のシンク・各点の需要が与えられたとき，すべての点の需要をシンクに流す合流フローの中で混雑が最小になるものを求める問題を，合流フローの混雑最小化問題という．本稿では，この問題に対して 2004 年に Chen et al. によって提案された手法の実験的性能評価を行い，得られた解に貪欲改善を加えると性能が大きく向上することを示す.

Experimental Evaluation and Greedy Improvement of Algorithms for Congestion Minimization Confluent Flow Problem

Information and System Engineering Course,
Graduate School of Science and Engineering, Chuo University

Koichi NIHEI　　Takao ASANO

Abstract　A flow is said to be confluent if at any node all the flow leaves along a single edge. Confluent flows arise in various applications including Internet routing. We consider the congestion minimization confluent flow problem: we are given a directed graph, sinks and demands on all the nodes, and the goal is to find a minimum congestion confluent flow that routes all the demands to sinks. In this paper, we evaluate experimental performance of algorithms proposed by chen et al. (2004), and show the availability of our greedy algorithm.

1　序論

ネットワーク・フローの中で,「すべての点においてフローが出て行く辺が 1 本以下」という条件を満たすものを**合流フロー** (confluent flow) という．現実社会において，合流フローとなる例は数多く存在する．いくつかの例を以下に挙げる．

- **インターネット・ルーティング:** インターネットにおける多くのルータは，宛先の情報 (IP アドレスやネットワークアドレス) だけを利用してルーティングを行っている．そこで，宛先が同一のパケットの流れを抜き出すと，合流フローになる．
- **CDN:** CDN (content delivery network, コンテンツ配信ネットワーク) におけるデータは，オリジナルの配信サーバから各 ISP (Internet service provider) のキャッシュサーバを経てクライアントへと送られる．このデータの流れを逆向きにすると，合流フローになる．
- **避難経路:** ビルやホテルなどの各フロアには非常口誘導灯があり，多くの場合，非常口までの最短経路を示す一方向の矢印がついている．建物内の人が一斉に避難するとき，人の流れは合流フローになる．

他の合流フローの例には，ブロードキャスト・マルチキャストにおけるデータの流れなどがある．

本稿では，まず問題定義を行い，Chen et al. の手法 [1] と提案手法について述べたあと，それらの手法に対する実験的性能評価を行う．

2 問題定義

$G = (V, E)$ を有向グラフとし，各点の**需要** (demand) を $d : V \to \mathbb{R}_+$ とする．本稿では G の点数 $|V|$ を n，辺数 $|E|$ を m と表す．また，k 個の**シンク** (sink(s)) を $S = \{s_1, \ldots, s_k\} \subset V$ とし，各シンクの出次数 (out-degree) は 0 であるとする．需要をシンクに流すフロー $f : E \to \mathbb{R}_+$ は，各点 $v \in V - S$ において，フロー保存則

$$\sum_{e=(v,w)\in E} f(e) - \sum_{e=(u,v)\in E} f(e) = d(v)$$

を満たす．また，各点 v において

$$\mathrm{in}(v) = \sum_{e=(u,v)\in E} f(e)$$

としたとき，フロー f の点 v における**混雑** (congestion) を

$$c(v) = d(v) + \mathrm{in}(v), \tag{1}$$

フロー f の混雑を

$$\max_{v\in V} c(v) \tag{2}$$

と定義する．

合流フローは連結成分 $\{T_1, \ldots, T_k\}$ からなる G の部分グラフになり，各連結成分 T_i は根 s_i に向かう有向木になる．T_i において混雑が最大となる点はシンク s_i であり，その混雑は T_i に含まれる点の需要の合計である (これを $d(T_i)$ と表す)．このとき，フローの混雑は

$$\max_{i=1,\ldots,k} d(T_i) \tag{3}$$

となる．

本稿で扱う**合流フローの混雑最小化問題** (congestion minimization confluent flow problem) の目的は，フローの混雑が最小となる合流フローを求めることである．この問題は，NP-困難な点素パス問題 (vertex-disjoint path problem) からのリダクションにより，近似率を $\frac{1}{2}\log_2 k$ より良くすることは NP-困難であることが証明されている [1].

合流フローの概念は 1999 年に発表された Kleinberg, Rabani and Tardos [5] の一部で述べられている．本稿で扱う合流フローの混雑最小化問題は，2003 年に Chen, Rajaraman and Sundaram [2] によって定義されたものであり，[2] ではランダム・ラウンディングを使用した $O(\log^3 n)$-近似アルゴリズム (最適な分割可能フローの混雑を 1 とおいて) が提案されている．また，2004 年には Chen et al. [1] によって $(1+\log_2 k)$-近似，$(1+\ln k)$-近似の決定性アルゴリズムが提案されている．合流フロー問題は新しい問題であり，今後，研究が進んでいくと思われる．

2.1 近似率の下界

合流フローに似た概念として，分割不可能フロー (unsplittable flow) がある．分割不可能フローとは，「各点の需要を 1 本のパスで流さなければならない」という制約がついたものである．分割不可能フローの混雑最小化問題に対しては Kleinberg [4] や Dinitz, Garg and Goemans [3] によって定数近似アルゴリズムが与えられている．また，これらの制約がなく，自由に分けて流してよいものを分割可能フロー (splittable flow) という．混雑が最小となる分割可能フローは，多項式時間で求めることができる．

本稿では，合流フローの混雑の下界として，分割可能フローを用いる．分割可能フローの最適解が，合流フローの最適解以下となるのは明らかである．Chen et al. [1] において，分割可能フローと合流フローの間のギャップが H_k となる例が示されている．

3 Chen et al. (2004) のアルゴリズム [1]

この節では，Chen et al. の $(1+\log_2 k)$-近似アルゴリズムと $(1+\ln k)$-近似アルゴリズムについて述べる．$(1+\ln k)$-近似アルゴリズムは，$(1+\log_2 k)$-近似アルゴリズムの一部を変更したものである．

準備: まず，混雑が最小となる分割可能フロー f を求め (容量を 2 分探索により変化させて最大フローアルゴリズムを適用し，すべての点の需要を流しきることができる最小の容量を求める)，その混雑が 1 となるように，各点の需要と各辺のフローを一様にスケーリングする．このとき，フローが流れない辺をグラフから削除する．最大フローは有向閉路が存在しないようにできるので，このグラフは有向無

閉路グラフと仮定できる．この有向無閉路グラフにおいて，シンクに隣接する各点を**境界点** (frontier node) という．また，アルゴリズムが開始されるとき，すべてのシンクは活性であるとする．

3.1 $(1+\log_2 k)$-近似アルゴリズム

下記の 1, 2, 3 の条件を順に判定し，一番はじめに条件を満たした操作を行う．

main loop: $|V(G)| > k$ の間，以下を実行する．

1. **node aggregation:** 境界点 v を始点とするすべての辺の終点が同一のシンク s_i のとき，その中の 1 本をマークし，v を s_i に縮約する．さらに，$d(v)$ を $d(s_i)$ に加え，main loop に戻る．
2. **breaking sawtooth cycle:** グラフ G における境界点からシンクへの各辺 $e=(v,s)$ に対して，逆辺 $e_{\mathrm{rev}}=(s,v)$ を付加したグラフを $\hat{G}$ とする．$\hat{G}$ が長さ 3 以上の単純有向閉路 C を持つとき f を以下のように更新する．$C \cap E(G)$ の辺におけるフローの最小値を $f_{\min}$ とする．G の各辺 e について，$e \in C$ ならば $f(e) := f(e) - f_{\min}$ とし，$e_{\mathrm{rev}} \in C$ ならば $f(e) := f(e) + f_{\min}$ とする．フローが 0 の辺をすべて削除し，main loop に戻る．
3. **sink deactivation:** 隣接する境界点が 1 つであるシンクを s_j とする．s_j に隣接する境界点を v，v に隣接する s_j 以外の任意のシンクを s_ℓ とする．このとき，$c(s_j)+f(v,s_\ell) < c(s_\ell)-f(v,s_\ell)$ ならば辺 (v,s_ℓ) を削除し，そこを流れるフローを (v,s_j) に流す．そうでないとき，辺 (v,s_j) を削除し，そこを流れるフローを (v,s_ℓ) に流して，s_j を不活性にする．その後，main loop に戻る．

output: マークされた辺を出力する．

1, 2 の条件を満たさない場合，境界点とシンクを結ぶすべての辺で誘導される G の (単純) 部分グラフは，向きを無視すると森になる．辺がある限り，この森には次数 1 の点が存在し，それが 3 の条件を満たすシンク s_j となる．

3.2 $(1+\ln k)$-近似アルゴリズム

$(1+\log_2 k)$-近似アルゴリズムの sink deactivation を以下のように変更する．

3. **parsimonious sink deactivation:** 以下の条件を満たす $G_1 \subseteq G$ を求める:
   - G_1 の各辺は境界点とシンクを結ぶ (G_1 は 2 部グラフになる)．
   - 境界点 v が G_1 に含まれるとき，v を始点とする G の辺はすべて G_1 に含まれる (したがって v から他の境界点に向かう G の辺はない)．
   - シンク s が G_1 に含まれるとき，s を終点とする G の辺はすべて G_1 に含まれる．
   - 各境界点は 2 つ以上のシンクに隣接している．

   G_1 に含まれるシンクの集合を S_1 と表す．G_1 に対して以下の操作を行う．

   i. **balancing:** G_1 において，$\sum_{s_i \in S_1} \mathrm{e}^{c(s_i)}$ が最小となるように各辺のフローを再分配し，フローが 0 となった辺を削除する．
   ii. $\mathrm{in}(s)$ が最小となるシンク $s \in S_1$ を求める．s に隣接する各境界点 v に対して，辺 (v,s) を流れるフローを v が隣接する他の任意のシンクに流し，辺 (v,s) を削除する．s を不活性にし，$S_1 := S_1 - \{s\}$ とする．
   iii. **balancing** を行い，main loop に戻る．

G_1 は以下のように求める．$\hat{G}$ を強連結成分分解し，各強連結成分を点とする有向無閉路グラフにおいて，出次数が 0 となる点集合が G_1 である．

次に，$\sum_{s_i \in S_1} \mathrm{e}^{c(s_i)}$ の最小化を考える．G_1 における境界点の集合を F_1 とする．また，辺 e を流れるフローを x_e と表すとき，balancing における $\sum_{s_i \in S_1} \mathrm{e}^{c(s_i)}$ の最小化は，以下の凸計画 (convex programming) 問題として定式化できる．ここで，y_s はシンク s の混雑を表す変数である．

$$\min \quad \sum_{s \in S_1} \mathrm{e}^{y_s} \tag{4}$$

$$\text{s.t.} \quad \sum_{e=(v,w) \in E(G_1)} x_e \geq c(v) \qquad (v \in F_1)$$

$$y_s \geq \sum_{e=(u,s) \in E(G_1)} x_e + d(s) \ (s \in S_1)$$

$$x_e \geq 0 \qquad (e \in E(G_1))$$

制約式中の $c(v)$ は，G における点 v の混雑を表す．また，凸計画問題は，多項式時間で最適解を求めることができる [6, 7].

3.3 解析の概要

node aggregation と breaking sawtooth cycle を行ったとき，どの点においても混雑が増加せず，少なくとも 1 本の辺が減少する．

sink deactivation と parsimonious sink deactivation の解析にはポテンシャル関数を用いる．シンク s_i のポテンシャルを

$$\phi(s_i) = 2^{c(s_i)} \tag{5}$$

とし，フローのポテンシャルを活性なシンクのポテンシャルの合計とする．このとき，sink deactivation を行ってもフローのポテンシャルは増加しない．アルゴリズムが開始されるとき，各シンクのポテンシャルは高々2 であり (各点の混雑は高々1 なので)，フローのポテンシャルは $2k$ 以下となる．sink deactivation を行ったあともこの値は増加しないので，アルゴリズム終了時の各シンクのポテンシャルは $2k$ 以下であり，各シンクの混雑が $1 + \log_2 k$ $(= \log_2(2k))$ 以下となる．

次にポテンシャル関数を

$$\phi(s_i) = \mathrm{e}^{c(s_i)} \tag{6}$$

と変更する．このとき，parsimonious sink deactivation を行ってもフローのポテンシャルは増加しない．よって各シンクのポテンシャルは $\mathrm{e}k$ 以下となるので，アルゴリズム終了時の各シンクの混雑は $1 + \ln k$ 以下となる．

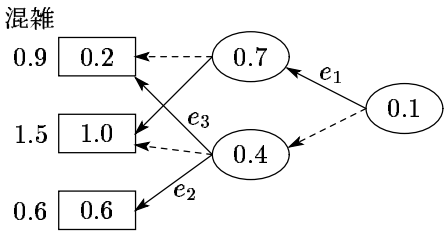
4 提案手法

Chen et al. のアルゴリズムで得られた解に対して，貪欲改善を行うと性能が大きく向上する．この節では，本稿で提案する手法について述べる．提案手法は，与えられた合流フローに対して，フローを流す辺を 1 本変えたときの混雑を求め，それが最小となるように解を変更していくものである．

与えられた合流フローにおいて，$d(T_i)$ が最大となる連結成分の集合を $T_{\max} \subseteq \{T_1, \ldots, T_k\}$ と表す (混雑が最大となるものが複数存在する場合もある)．このとき，本稿において提案するアルゴリズムを以下に述べる．

1. $u \in T_{\max}$, $v \notin T_{\max}$ となる各辺 $e = (u, v) \in E(G)$ に対して，点 u からのフローを辺 e に流したときのフローの混雑 (これを $r(e)$ と表す) を計算する．
2. $r(e)$ が最小となる辺 $e_{\min}$ を求める．
3. 現在のフローの混雑を $cong$ としたとき，$cong > r(e_{\min})$ ならば，合流フローを辺 $e_{\min}$ に変更して，1 へ戻る．$cong \leq r(e_{\min})$ ならば現在のフローを出力する．

具体例を図 1 に示す．現在の合流フローを破線で表すとき，$u \in T_{\max}, v \notin T_{\max}$ となるのは e_1, e_2, e_3 である．このとき，$r(e)$ が最小となるのは e_2 なので，e_2 にフローを変更し，同様の操作を繰り返す．



e	e_1	$\boldsymbol{e_2}$	e_3
$r(e)$	1.4	**1.1**	1.4

図 1　提案手法の具体例

5 実験的性能評価

5.1 方法

本稿では，以下の 5 手法の比較を行った．

A. $(1 + \log_2 k)$-近似アルゴリズム

B. $(1 + \ln k)$-近似アルゴリズム

C. $(1 + \log_2 k)$-近似アルゴリズムで得られた解を初期解として，提案手法を適用

D. $(1 + \ln k)$-近似アルゴリズムで得られた解を初期解として，提案手法を適用

E. 各点の需要を最も近いシンクに流す合流フローを初期解として，提案手法を適用

近似率の下界には，混雑最小の分割可能フローを用いる (2.1 節参照)．また，各点数，辺数，シンク数に対して，1,000 個のインスタンスを生成し，それらを A から E のすべての手法で解いた．実験で使用した計算機を表 1 に示す.

表 1　実験環境

CPU	Intel Pentium4 1.7 GHz
メモリ	1,024 MB RDRAM
OS	Red Hat Linux 7.3
コンパイラ	g++ 2.96

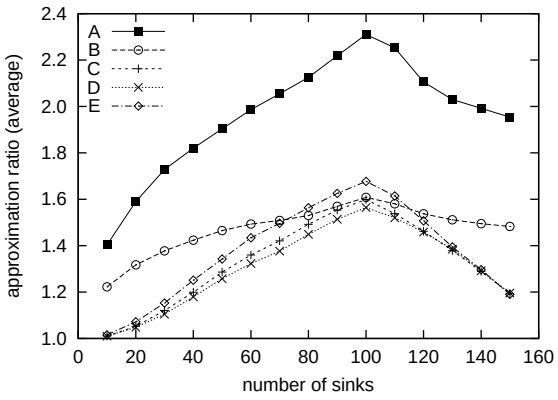


図 2　シンク数の変化

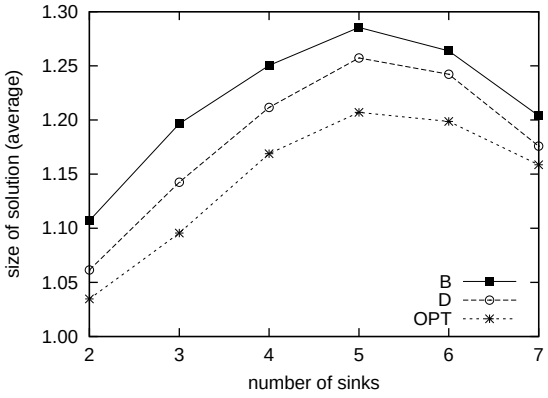


図 3　最適解との比較

5.2　結果

5.2.1　シンク数を変化させた場合

まず，点数・辺数を固定し，シンク数を変化させた．図 2 は，点数を 200，辺数を 1,000 として，シンク数を変化させた場合の近似率の平均である．各点の需要は一様乱数で与えた．どの手法を用いた場合にも，シンク数が 100 のときに最も悪い結果となった．シンクが少ない場合や多い場合には，提案手法の結果が非常に良くなったが，シンク数が 100 付近では，従来手法との差が小さくなった．点数を 100, 300 として同様の実験を行ったところ，どちらの場合もシンク数が点数のちょうど半分のときに全手法で最悪の結果となり，従来手法と提案手法の性能差も小さくなった.

この原因を調べるために，最適解との比較を行った．大きなインスタンスに対して最適解を求めるのは困難なので，点数 10，辺数 20 のグラフに対してシンク数を変化させる．結果を図 3 に示す．グラフは分割可能フローの混雑を 1 としたときの，最適解 (OPT)，手法 B, D の解の平均である．シンク数が点数の半分のときに，下界 (分割可能フロー) と最適解の間のギャップが大きくなっており，最適解と手法 B, D の間には，シンク数による混雑の変化は見られなかった．大きなインスタンスにおいても，同様のことが予想できる．また，点数に対するシンクの割合が等しい場合には，点数が多い (つまりシンク数が多い) 場合の方が近似率が悪くなる (表 2).

表 2　グラフの大きさの変化

インスタンス			近似率 (平均)	
n	m	k	B	D
100	500	50	1.554324	1.527063
200	1,000	100	1.608197	1.564448
300	1,500	150	1.638836	1.588455

5.2.2　辺数を変化させた場合

次に，点数を 200，シンク数を 20 に固定して辺数 (辺の密度) を変化させた (図 4)．点数に対するシンクの割合が小さいので，提案手法の結果が非常に良くなった．辺数が増加すると，どの手法も得られる解が多少良くなるが，大きな変化はない.

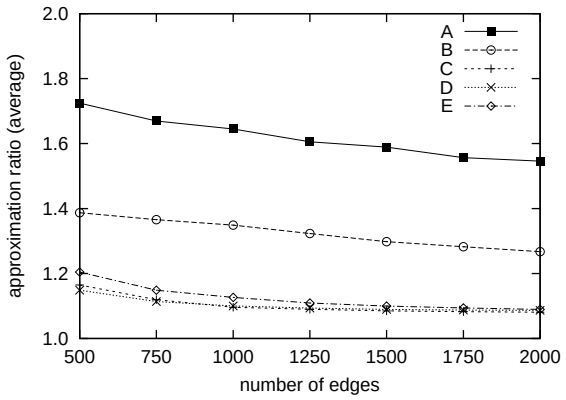


図 4　辺数の変化

5.2.3　需要を変化させた場合

こんどは，各点の需要をシンクまでの距離によって定めた．各点 v から最も近いシンクまでの距離 (ホップ数) を $h(v)$ とする．このとき，各点の需要を $h(v)+1$ とした場合 (以下**昇順**) と，$1/(h(v)+1)$ とした場合 (以下**降順**) について，一様乱数との比較を行った．表 3 は，それぞれの需要の与え方と手法に対して，点数 200，辺数 1,000，シンク数 50 の 1,000 個のインスタンスにおける近似率の分布である．

手法 A, B は昇順に対してはあまり変化せず，降順の場合には一様乱数の場合より良い結果となった．Chen et al. の手法は，シンクに隣接した点 (境界点) からフローを決定するため，シンクに近い点の需要が大きい降順で良い結果が得られたと考えられる．

手法 C は，一様乱数と比べて昇順，降順ともに悪くなり，特に昇順の結果が悪くなった．C の初期解である A は，降順において一様乱数より良い結果となったが，C の解は悪くなったことから，提案手法の性能は必ずしも初期解の良さに依存するわけではないといえる．

手法 D は，昇順に対してやや悪くなり，降順に対しては一様乱数よりも良い結果になった．悪い結果になった昇順の場合でも，すべての手法の中では最も良い結果になっている．

手法 E は，一様乱数の場合には A, B よりも良い結果になったが，需要を昇順，降順とした場合には，非常に悪い結果になった．昇順では，近似率 4 を超える場合もあった．手法 E の反復回数は，需要を一様乱数で与えた場合には平均 57.8 回だったのに対して，昇順の場合には 3.1 回であり，初期解からの改善がほとんど行われていない (表 5).

5.2.4　初期解に対する提案手法の解の変化

さらに，初期解に対する提案手法の解の分布を求めた．表 4 は，点数 200，辺数 1,000，シンク数 50，需要を一様乱数で与えたときの，初期解 (手法 B により得られる解) と提案手法の解 (手法 D により得られる解) の関係を表したものである．この例では，B の解の大きさによらず，D の解が $[1.2, 1.3)$ となったものが多かった．また，B の解が悪かったにもかかわらず D の解が良くなったもの (表の左下) や，B の解をほとんど改善できなかったもの (表の右側) もあった．

しかし，需要を昇順，降順としたときに手法 E の解が悪くなったように，提案手法は悪い初期解を与えた場合に必ずしも大きく改善できるわけではない．

5.2.5　計算時間と反復回数

最後に，計算時間と提案手法の反復回数について述べる．手法 C, D, E の計算時間には，初期解を求める時間も含まれる．いくつかの結果を表 5 に示す．

提案手法の計算時間は反復回数に依存する．需要を昇順，降順で与えた場合には，一様乱数の場合に比べて，どの手法も反復回数が大きく減少している．これは，一様乱数に比べて局所解が多くなり，すぐに反復が停止してしまうためだと考えられる．また，辺数が多くなると提案手法の反復回数は増加する．

インスタンスが大きくなると，反復回数の少ない手法 D の計算時間が提案手法の中では最も短くなる．手法 D は，5,000 点のインスタンスに対して 3 分程度で解が求まっており，これは実用時間内であるといえる．

5.3　考察

本節では，Chen et al. の手法と提案手法に対して実験的性能評価を行った．Chen et al. の手法は，理論的に証明されている近似保証よりも良い結果になった．

シンク数が点数の 1/2 のときにはどの手法も悪い結果となったが，これは合流フローの最適解と，下界として用いている分割可能フローとのギャップが大きくなるためだと考えられる．

どのような入力に対しても，提案手法である手法 D の解が最も良くなった．点数 5,000，辺数 50,000，シンク数 500，需要を一様乱数で与えた場合 (表 5

表 3　需要の変化

	一様乱数					昇順					降順				
近似率	A	B	C	D	E	A	B	C	D	E	A	B	C	D	E
[1.0, 1.2)	0	0	96	205	24	0	0	0	5	0	0	22	0	227	0
[1.2, 1.4)	0	294	819	754	747	0	370	18	811	0	20	674	388	757	0
[1.4, 1.6)	35	597	79	40	207	53	524	392	170	0	355	291	523	16	20
[1.6, 1.8)	312	102	6	1	22	257	93	437	13	2	404	12	77	0	261
[1.8, 2.0)	379	7	0	0	0	441	13	130	1	26	178	1	12	0	344
[2.0, 2.2)	184	0	0	0	0	161	0	19	0	146	38	0	0	0	250
[2.2, 2.4)	67	0	0	0	0	70	0	4	0	250	4	0	0	0	102
[2.4, 2.6)	21	0	0	0	0	13	0	0	0	213	1	0	0	0	16
[2.6, 2.8)	1	0	0	0	0	5	0	0	0	188	0	0	0	0	5
[2.8, 3.0)	0	0	0	0	0	0	0	0	0	92	0	0	0	0	2
[3.0, 3.2)	1	0	0	0	0	0	0	0	0	42	0	0	0	0	0
[3.2, 3.4)	0	0	0	0	0	0	0	0	0	21	0	0	0	0	0
[3.4, 3.6)	0	0	0	0	0	0	0	0	0	10	0	0	0	0	0
[3.6, 3.8)	0	0	0	0	0	0	0	0	0	5	0	0	0	0	0
[3.8, 4.0)	0	0	0	0	0	0	0	0	0	4	0	0	0	0	0
[4.0, 4.2)	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0

表 4　初期解と提案手法の解の関係

初期解	提案手法の解 (D の解)							
(B の解)	[1.1, 1.2)	[1.2, 1.3)	[1.3, 1.4)	[1.4, 1.5)	[1.5, 1.6)	[1.6, 1.7)	[1.7, 1.8)	計
[1.2, 1.3)	9	27	0	0	0	0	0	36
[1.3, 1.4)	63	151	44	0	0	0	0	258
[1.4, 1.5)	77	224	77	18	0	0	0	396
[1.5, 1.6)	41	116	35	5	4	0	0	201
[1.6, 1.7)	14	42	20	7	1	0	0	84
[1.7, 1.8)	1	10	3	3	0	0	1	18
[1.8, 1.9)	0	1	2	0	2	0	0	5
[1.9, 2.0)	0	1	1	0	0	0	0	2
計	205	572	182	33	7	0	1	1,000

表 5　計算時間と反復回数

	インスタンス				A	B	C		D		E	
	n	m	k	需要	時間	時間	時間	反復	時間	反復	時間	反復
1	200	1,000	50	昇順	0.1017	0.2186	0.1130	1.75	0.2330	2.30	0.0115	3.14
2	200	1,000	50	降順	0.0961	0.2150	0.1068	2.04	0.2260	1.21	0.0114	3.59
3	200	1,000	50	乱数	0.0988	0.2279	0.1572	18.51	0.2492	5.67	0.1346	57.84
4	200	1,000	100	乱数	0.1441	0.4933	0.2087	15.79	0.5006	1.04	0.1718	60.97
5	200	2,000	50	乱数	0.1144	0.2421	0.3289	28.46	0.3355	10.12	0.5061	93.44
6	500	5,000	50	乱数	0.4182	0.5990	2.0120	75.10	1.6599	49.32	3.3198	203.6
7	1000	10,000	100	乱数	1.7189	2.2631	8.0462	138.2	6.3063	84.23	17.469	445.1
8	5000	50,000	500	乱数	78.564	73.333	286.20	574.3	187.95	307.3	646.03	2241

計算時間，反復回数ともに平均値である．また，計算時間の単位は秒である．

の 8)，手法 B の近似率の平均が 1.425 だったのに対して，手法 D では 1.065 に改善された．大きなインスタンスに対して，手法 D は提案手法の中で最も短い計算時間で解が求まり，従来手法と比較しても大きな増加にはなっていない．

初期解に近似保証が与えられていない手法 E は，需要が一様乱数の場合には良い結果が得られるが，昇順，降順とした場合には非常に悪い結果となるので，有用ではないといえる．

6　結論

本稿では，合流フローの混雑最小化問題に対する近似アルゴリズムの実験的性能評価を行った．本稿で提案した貪欲改善法は Chen et al. の手法 [1] に比べて，実際的性能が大きく向上した．

今後の課題としては，以下のことが挙げられる．

- 近似率・計算時間に関して，提案手法の理論的解析を行う．
- 新たな手法を提案する．
- モデルを拡張する (シンクの処理能力が異なる場合や多品種の問題など)．

謝辞

本研究は，一部，21 世紀 COE プログラムおよび文部科学省科学研究費補助金からの援助のもとで行われたものである．

参考文献

[1] J. Chen, R. D. Kleinberg, L. Lovasz, R. Rajaraman, R. Sundaram and A. Vetta: (Almost) Tight Bounds and Existence Theorems for Confluent Flows. *Proceedings of the 36th Annual ACM Symposium on Theory of Computing*, Chicago, Illinois, June 2004, pp. 529–538.

[2] J. Chen, R. Rajaraman and R. Sundaram: Meet and Merge: Approximation Algorithms for Confluent Flows. *Proceedings of the 35th Annual ACM Symposium on Theory of Computing*, San Diego, California, June 2003, pp. 373–382.

[3] Y. Dinitz, N. Garg and M. Goemans: On the Single-Source Unsplittable Flow Problem. *Proceedings of the 39th Annual IEEE Symposium on Foundations of Computer Science*, Palo Alto, California, November 1998, pp. 290–299.

[4] J. Kleinberg: Single-Source Unsplittable Flow. *Proceedings of the 37th Annual IEEE Symposium on Foundations of Computer Science*, Burlington, Vermont, October 1996, pp. 68–77.

[5] J. Kleinberg, Y. Rabani and E. Tardos: Fairness in Routing and Load Balancing. *Proceedings of the 40th Annual IEEE Symposium on Foundations of Computer Science*, New York City, New York, October 1999, pp. 568–578.

[6] 小島政和・土谷隆・水野真治・矢部博 :『内点法』. 朝倉書店，東京，2001.

[7] 田村明久・村松正和 :『工系数学講座 17 最適化法』. 共立出版，東京，2002.