Simplicial Decomposition of the Asymmetric Traffic Assignment Problem

Lawphongpanich, S. & Hearn, D. W. (1984)

Transportation Research Part B: Methodological, 18(2), 123-133

理論談話会#2

2019/04/25

M1 熊野 孝彦

目次

1. 概要

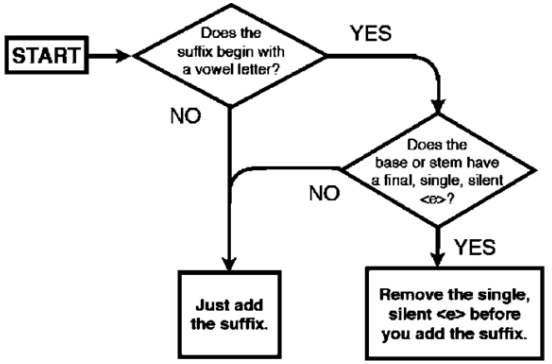
目的

非対称交通量配分問題の変分不等式計算のための
収束するSimplicial Decompositionアルゴリズムを紹介

・費用関数に基づいた最短経路ツリーの作成と，単純な凸制約の条件下にある変分不等式の解の計算を繰り返し行う．

→ Frank-Wolfe 法を非対称問題に一般化したもの

・dropping flow pattern を採用．



1. 概要

・確率的利用者均衡配分(SUE)の解法の一つ

【特徴】
経路交通量を未知数とする
Logit型SUE配分専用
収束が早い

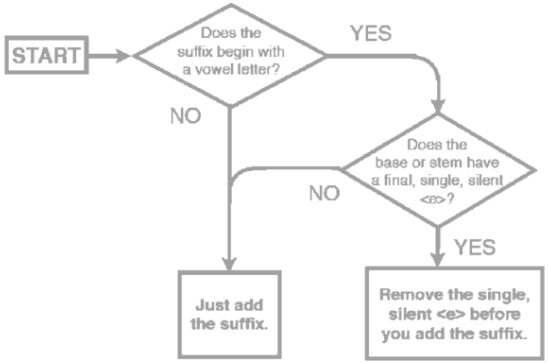
目的

非対称 のための
収束するSimplicial Decompositionアルゴリズムを紹介

・費用関数に基づいた最短経路ツリーの作成と，単純な凸制約の条件下にある変分不等式の解の計算を繰り返し行う．

→ Frank-Wolfe 法を非対称問題に一般化したもの

・dropping flow pattern を採用．



1. 概要

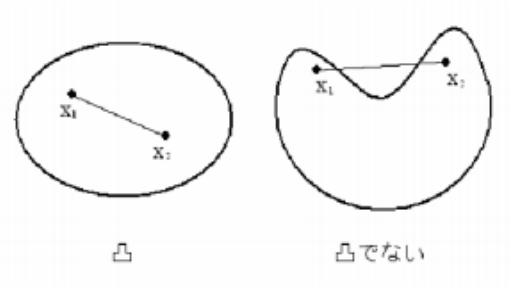
目的

非対称交通量配分問題の変分不等式計算のための
収束するSimplicial Decompositionアルゴリズムを紹介

・費用関数に基づいた最短経路ツリーの作成と，単純な**凸制約**の条件下にある変分不等式の解の計算を繰り返し行う．

→ Frank-Wolfe 法を非対称問題に一般化したもの

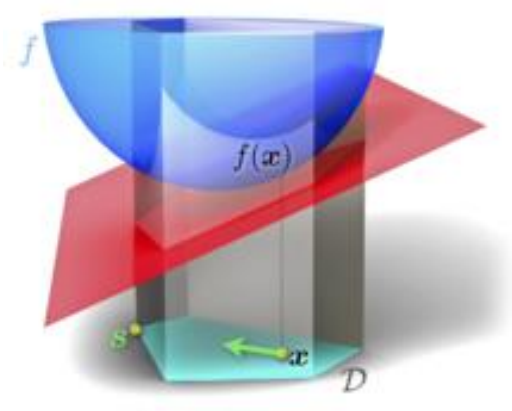
・dropping flow pattern を採用．



x_1, x_2 を結ぶ線上の任意の点が同一集合内にある

(参考) Frank-Wolfe法 その 1

- 数理的最適化問題を解く際のアルゴリズム



仮定： 関数$f: \mathrm{K} \to \mathbb{R}$の勾配$\nabla f(x)$が計算できる (K:凸集合)
K上での線形最適化問題$\arg min_{x \in \mathrm{K}} \boldsymbol{x} \cdot \boldsymbol{g}$が解ける

$initialize\ \boldsymbol{x}_0 \in \mathrm{K}$ 　$\boldsymbol{x}_0$を初期化

$for\ k = 0,1,2, \ldots$

収束するまで繰り返す

$\boldsymbol{s}_k = \arg min_{x \in \mathrm{K}}\ \boldsymbol{s} \cdot \boldsymbol{\nabla} f(\boldsymbol{x}_k)$ 　$\arg min f(x)$: f(x)を最小にするxの集合

$\gamma_k = \frac{2}{2+k}$ or $\gamma_k = \arg min_{\gamma \in [0,1]} f(\boldsymbol{x}_k + \gamma(\boldsymbol{s}_k - \boldsymbol{x}_k))$

$\boldsymbol{x}_{k+1} = (1 - \gamma_k)\boldsymbol{x}_k + \gamma_k \boldsymbol{s}_k$ 　γ_k: ステップ幅

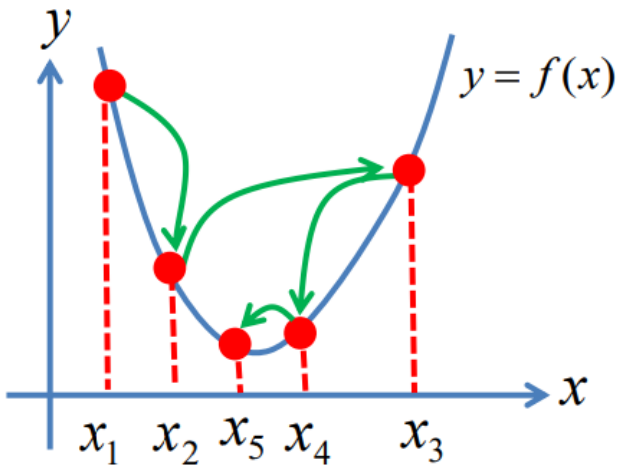
end for 　$\boldsymbol{s}_k$は勾配降下方向でKの端にいる

(参考) Frank-Wolfe法 その 2

簡単に言うと・・・

x_kごとにステップ幅と降下方向ベクトルを
定めて，解が収束するまで繰り返す.

→ $\nabla f(\boldsymbol{x}) \leq \kappa$　(κは微小な定数)
　となったら終わり.



2. 背景と既往研究

凸集合のプログラミング計算のためのSimplicialストラテジー

- Holloway (1974)
- Hohenbalken (1975, 1977)
- Sacher (1980)

対称問題の凸集合プログラミング計算のための列生成メソッド

- Leventhal et al. (1973)
- Cantor & Gerla (1974)

- Bertsekas & Gafni (1982)
- Pang & Yu (1982)

Simplicial Decomposition

Hohenbalken, V. (1977) = Smith (1983b)

2. 背景と既往研究

凸集合のプログラミング計算のためのSimplicialストラテジー

- Holloway (1974)
- Hohenbalken (1975, 1977)
- Sacher (1980)

対称問題の凸集合プログラミング計算のための列生成メソッド

- Leventhal et al. (1973)
- Cantor & Gerla (1974)

- Bertsekas & Gafni (1982)
- Pang & Yu (1982)

Simplicial Decomposition

Hohenbalken, V. (1977) = Smith (1983b)

このへんを踏まえて
非対称問題に適用

3. アルゴリズムの紹介

Variational Inequality

・利用者均衡配分は変分不等式問題$VI(C,S)$で表せる

$$VI(C,S) = C(x^*)(x - x^*) \geq 0 \qquad \forall x \in \boldsymbol{S}$$

$\boldsymbol{S} \subset \mathbb{R}^{\boldsymbol{n}}$
($\boldsymbol{n}$はネットワーク上の弧の数)

$\boldsymbol{C(x)}$: $\boldsymbol{S} \subset \mathbb{R}^{\boldsymbol{n}} \to \mathbb{R}^{\boldsymbol{n}}$ で，$\forall x_1 \neq x_2$ において
$(C(x_1) - C(x_2))(x_1 - x_2) > 0$ を満たす連続関数を考える

↓ を満たすx^*を求める

Smith (1979) の方法

$$VI(C,S) = \boldsymbol{A}'C(\boldsymbol{A}\boldsymbol{\lambda}^*)(\lambda - \lambda^*) \geq 0 \qquad \forall \lambda \in \boldsymbol{\Lambda}$$

を満たすλ^*を求める

$\boldsymbol{A}$は$n \times m$行列
($\boldsymbol{m}$はextreme flow patternsの数)

$$\boldsymbol{\Lambda} = \left\{ \sum_{i-1}^{m} \lambda_i = 1, \lambda_i \geq 0 \right\}$$

下の式のアルゴリズム(SDVI)を考える

前提条件: 許容差δと，$\varepsilon_t > \varepsilon_{t+1} > 0$ かつ$t \to \infty$のとき$\varepsilon_t \to 0$
となる$\{\varepsilon_t\}$を仮定する.

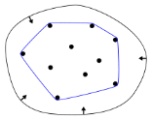
3. アルゴリズムの紹介

Step.0 a^1を凸集合**S**の極点とする．

$A^1 = \{a^1\}$, $\bar{G}^1 = \infty$, $t = 1$ と定義する． ($\bar{G}$: ギャップ)

Step.1 $\forall x \in H(A^t), C(x')(x - x') \geq -\varepsilon_t$ を満たす$x^t \in H(A^t)$を求める．

ここで， $H(A^t)$はA^tの列の凸包である．



与えられた点をすべて包含する最小の凸多角形(or凸多面体)のこと

$G(x^t)$：ギャップ関数

Step.2

- $G(x^t) \triangleq C(x^t)(x^t - y^t) = 0$ のとき，終わり．
- $G(x^t) \geq \bar{G}^t - \delta$ のとき， $A^{t+1} = A^t \cup y^t$
- $G(x^t) < \bar{G}^t - \delta$ のとき， $A^{t+1} = \frac{A^t}{D^t} \cup y^t$

$\bar{G}^{t+1} = min\{\bar{G}^t, G(x)\}$ とし，tを増やしていく．

→ Step.1 に戻る！

D^tはx^tの重みゼロのときのA^tの列とする

3. アルゴリズムの紹介

・もし

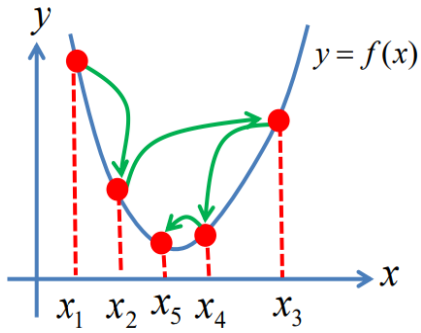
$$\forall x_1, x_2 \in S, \big(C(x_1) - C(x_2)\big)(x_1 - x_2) \geq \alpha \|x_1 - x_2\|^2$$

を満たす$\alpha > 0$が存在し，$\hat{x}$が$VI(C,S)$の解であるとき，

$$\begin{aligned}
\alpha\|x^t - \hat{x}\|^2 &\leq \big(C(x^t) - C(\hat{x})\big)(x^t - \hat{x}) \\
&= C(x^t)(x^t - \hat{x}) - C(\hat{x})(x^t - \hat{x}) \\
&\leq C(x^t)(x^t - \hat{x}) \\
&\leq max_{|w \in H(A^t)}\, C(x^t)(x^t - \hat{x}) \\
&\leq \varepsilon_t \\
\therefore \|x^t - \hat{x}\|^2 &\leq \frac{\varepsilon_t}{\alpha}
\end{aligned}$$

よって，x^tが$\hat{x}$の近くにあると保証するためにε_tが用いられる．

4. 収束することの証明



major iteration

前提条件

・ギャップ関数$G(x^t)$が0になるまでの繰り返しを「大回り」と呼ぶ.

・まず，[T: 大回りの添え字集合] と定義する.

$$T = \{t: C(x^t)(x^t - y^t) < \bar{G}\,;\, t = 1,2,3,\dots\}$$

・もしk回で繰り返しが終わる場合，x^kは$VI(C,S)$の解である.
また，$k \in T$ である.

$$\because C(x^k)(x^k - y^k) = 0 < min\{C(x^t)(x^t - y^t)\,;\, t < k\}$$
かつ $\forall t < k,\ C(x^t)(x^t - y^t) > 0$

→よって，アルゴリズムは解き終わると大回りの解を出力する.

・また，アルゴリズムは数列$\{x^t\}$を出力し，全ての$\{x^t\}_{t\in T}$が解に収束することを証明する.

・パラメータδは最終的に列が消去されなくなることを保証し，これが収束の証明を導く.

4. 収束することの証明

定理 アルゴリズムが無限の数列$\{x^t\}$を出力し，Tが無限の集合であるとき，Tの任意の列が$VI(C,S)$の収束する解である．

証明

$x^t \in \boldsymbol{S}, \forall t \in T$で， $\boldsymbol{S}$がコンパクトなので，$\hat{T} \subseteq T$で収束する列$\{x^t\}_{t\in\hat{T}}$が必ず存在する．またx^*は列$\{x^t\}_{t\in T}$の極限点とする．

ギャップ関数$G(x)$が$\boldsymbol{S}$上の正の連続関数なので，

$$\forall t \in \hat{T}, t > K \ \ |G(x^t) - G(x^*)| < \delta$$

を満たす整数Kが存在する．

数列$\{G(x^t)\}_{t\in T}$の単調性から，次のことが言える．

(i) $\forall t \in \hat{T}, t > K \ \ G(x^t) - G(x^*) < \delta$

(ii) $t_1, t_2 \in \hat{T} \ | t_2 > t_1 > k, \ \ G(x^{t_1}) - G(x^{t_2}) < G(x^{t_1}) - G(x^*) < \delta$

(iii) $t \in T \ | t_1 < t < t_2, \ G(x^{t_1}) - G(x^t) < G(x^{t_1}) - G(x^{t_2}) < \delta$

(ii),(iii)と，$t \notin T$の繰り返しにおいて列消去がないことより．

K回目の繰り返しの後に列消去がなく，集合A^Tの数列で，

$t > k$は単調増加数列であるといえる．

$\boldsymbol{S}$が有限の極点を持っていることから，数列$\{A^t\}_{t<k}$がある集合$\hat{A}$に収束する．即ち$A^t = \hat{A}, \ \forall t > \hat{k}$を満たす整数$\hat{k}$が存在する．

ここで， step.1 より

$$\forall t, \quad C(x^t)(x - x^t) \geq -\varepsilon_t$$

しかし，$t > \hat{k}$で$A^t = \hat{A}$より，y^tは

$$C(x^t)(x^t - y^t) = max_{|w\in H(A^t)} \, C(x^t)(x^t - \hat{x})$$

を満たす$\hat{A}(= A^t)$の要素となる．

従って，$\forall t > \hat{k}$で，

$$G(x^t) = C(x^t)(x^t - y^t) = max_{|w\in H(A^t)} \, C(x^t)(x^t - \hat{x}) \leq \varepsilon_t$$

ε_tは0に収束するので，

$$\lim_{t\in\hat{T}} G(x^t) = 0 = G(x^*)$$

よって，x^*は$VI(C,S)$の収束する解である． ■

5. アルゴリズムの考察

(1)このアルゴリズムは射影法，ユーザー定義関数の最適化，線形近似法の考え方を含んでいる．

(2)アルゴリズムのstep.2の下位問題は，Frank-WolfeとCantor-Gerla法の最短経路探索法と似ている．

(3)ギャップ関数は部分問題を解くときの副産物であり，終了条件をつくる．

6. 計算結果

・その前に…

証明段階では$\varepsilon_t \to 0$ と，終了条件として $G(x^t) = 0$を仮定したが，計算過程ではεの解を求める方が実用的である．そこで

$$\varepsilon_t = 10^{-4} + 5 \times 10^{-4}/t, \qquad \delta = 10^{-4}$$

とする．

さらに，アルゴリズムは相対的なギャップ

$$C(x^t)(x^t - y^t)/C(x^t)y^t$$

が10^{-4} より小さいときに終了する．

今回の計算テストでは射影法で解くので，アルゴリズムは次のように表せる．

6. 計算結果

証明

Step.0

a^1を凸集合**S**の極点とする．

$A^1 = \{a^1\},\ \bar{G}^1 = \infty,\ t = 1$ と定義する．　($\bar{G}$: ギャップ)

Step.1

$\forall x \in H(A^t), C(x')(x - x') \geq -\varepsilon_t$ を満たす$x^t \in H(A^t)$を求める．

ここで，　$H(A^t)$はA^tの列の凸包である．

Step.2

- $G(x^t) \triangleq C(x^t)(x^t - y^t) = 0$ のとき，終わり．
- $G(x^t) \geq \bar{G}^t - \delta$ のとき，$A^{t+1} = A^t \cup y^t$
- $G(x^t) < \bar{G}^t - \delta$ のとき，$A^{t+1} = \frac{A^t}{D^t} \cup y^t$

  $\bar{G}^{t+1} = min\{\bar{G}^t, G(x)\}$ とし，tを増やしていく．

→ Step.1 に戻る！

6. 計算結果

計算

Step.0

a^1を凸集合**S**の極点とする．

$A^1 = \{a^1\},\ \bar{G}^1 = \infty,\ t = 1$ と定義する．　($\bar{G}$: ギャップ)

Step.1′

(1) $w^0 \in H(A^t)$を選び，k=0とする．

(2) w^{k+1}を$H(A^t)$上の$\{w^k - \alpha M^{-1} C(w^k)\}$ の射影とする．

(3) ① $C(w^{k+1})(x - w^{k+1}) \geq -\varepsilon_t,\ \forall x \in H(A^t)$ のとき，**Step.2** へ進む．

このとき，$x^t = w^{k+1}$, D^tはStep.1と同様に定義する．

②それ以外の場合，$k = k + 1$ として(2)へ戻る．

実計算ではλを考慮

Step.2

- $G(x^t) \triangleq C(x^t)(x^t - y^t) = 0$ のとき，終わり．
- $G(x^t) \geq \bar{G}^t - \delta$ のとき，$A^{t+1} = A^t \cup y^t$
- $G(x^t) < \bar{G}^t - \delta$ のとき，$A^{t+1} = \frac{A^t}{D^t} \cup y^t$

$\bar{G}^{t+1} = min\{\bar{G}^t, G(x)\}$ とし，tを増やしていく．

→ **Step.1′** に戻る！

6. 計算結果

・その前に…(part.2)

変分不等式の射影アルゴリズムが初めて交通量配分問題に使われたのはDafermos (1980). その次はBertsekas & Gafni (1982)である.

Step.1′ で変数λ を簡単に使いたいならBertsekas & Gafni (1982)を使う.

→この方法において与えられた発地からのフローを"Commodity"と呼び, この方法を"Commodity射影法"呼ぶことにする.

→ Simplicial Decomposition と Commodity射影法を比較する.

6. 計算結果

・Commodity射影法

Step.0 各Commodity i に対して，$a_i{}^1$をS_iの極点とする．

$x_i{}^0 = a_i{}^1,\ x^0 = \sum_i x_i{}^0,\ A_i = \{a_i\},\ t = 1$ とする．

Step.1 各Commodity i に対して，$x_i{}^t$ を計算し，$\{x_i{}^{t-1} - \alpha M^{-1} C(x^{t-1})\}$ として$H(A_i{}^t)$に射影する．

また，$x^t = \sum_i x_i{}^t$ とする．

Step.2 各Commodity i に対して，$C(x^t)y_i{}^t = min\{C(x^t)y_i{}^t,\ y_i \in S_i\}$ を解く．

また，$y^t = \sum_i y_i{}^t$ とおく．

・ $G(x^t) = C(x^t)(x^t - y^t) = 0$ なら終わり．

・それ以外の場合，$A_i{}^{t+1} = A_i{}^t \cup y_i{}^t$ として，Step.1 に戻る．

→ 1回の繰り返しごと，1つのCommodityごとに1回射影を行う．

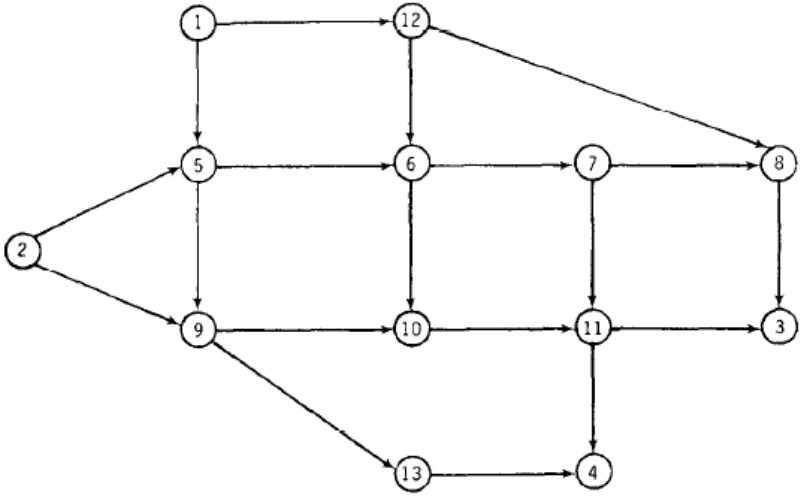
6. 計算結果

・今回は４つの問題を考える.

はじめの３つの問題(ND1～ND3)は右のネットワークについてアルゴリズムを解いていく.

２つの方法での解を出すまでの繰り返し回数を比較

α と Mの値は試行錯誤して定めた.
(その値は次ページからの表を参照.)



Network Topology

	3	4
1	400	800
2	600	200

Trip Matrix

Fig. 1. Data for problem ND1, ND2 and ND3.

6. 計算結果

・ND1

p_1: フローパターン数
p_2: 射影の回数

同じ繰り返し回数：
相対的なギャップを使うと
ギャップ関数より正確.

Table 1. Computational results for problem ND1

The Simplicial Algorithm: $\alpha = 0.8$ and $M = [1.2\, C(x^t)x^t] \cdot I$						
t	$F(x^t)$	Gap	Rel. Gap	p_1^1	p_2^2	Time[3] (CPU sec.)
1	94,252.81	$.1004\ 10^6$	$.1136\ 10^1$	2	19	.0910
2	86,560.56	$.2528\ 10^5$	$.2823\ 10^0$	3	26	.1265
3	85,894.68	$.4844\ 10^4$	$.4951\ 10^{-1}$	4	56	.1722
4	85,513.25	$.5183\ 10^4$	$.5343\ 10^{-1}$	5	42	.2164
5	85,214.62	$.1835\ 10^4$	$.1855\ 10^{-1}$	5	31	.2571
6	85,046.81	$.1849\ 10^4$	$.1873\ 10^{-1}$	6	106	.3318
7	85,027.75	$.6565\ 10^3$	$.6558\ 10^{-2}$	5	150[4]	.4263
8	85,027.62	$.6744\ 10^2$	$.6711\ 10^{-3}$	6	72	.4887
9	85,027.62	$.6750\ 10^1$	$.6713\ 10^{-4}$	6		.5178
					502	

Notes: 1: p_1 = # of active extreme aggregate flow patterns
2: p_2 = # of projections
3: Time on IBM 3033
4: 150 is the maximum number of projections allowed per iteration

6. 計算結果

・ND2

	Sim		Com
経路ツリー数	6	<	26
射影回数	109	>	26

しかし"Time"列を比べると
Com の方が多い.

→20個の経路ツリーを作る
より83回射影する方が
時間がかかる！

Table 2. Computational results for problem ND2

The Simplicial Algorithm $\alpha = 0.8$ and $M = (0.486 \cdot 10^6) \cdot I$					The Commodity Projection Method $\alpha = 0.8$ and $M = (0.486 \cdot 10^6) \cdot I$			
t	Rel. Gap	p_1^2	p_2^3	Time[1]	t	Rel. Gap	p_3^4	Time[1]
1	$.2677\ 10^{1}$	2	22	.0882	1	$.2677\ 10^{1}$	2,2	.0827
2	$.3446\ 10^{0}$	3	11	.1209	5	$.2035\ 10^{-1}$	4,2	.1847
3	$.1169\ 10^{0}$	3	15	.1534	10	$.5091\ 10^{-2}$	4,2	.2846
4	$.1852\ 10^{-1}$	4	61	.2038	15	$.1449\ 10^{-2}$	4,2	.4355
5	$.6524\ 10^{-4}$	4		.2329	20	$.4229\ 10^{-3}$	4,2	.5599
			109		25	$.1241\ 10^{-3}$	4,2	.6862
					26	$.9681\ 10^{-4}$	4,2	.7111

Notes: 1 : Time on IBM 3033
2 : p_1 = # of active extreme aggregate flow patterns
3 : p_2 = # of projections
4 : p_3 = # of active extreme commodity flows for commodities 1 and 2

6. 計算結果

・ND3

ND2と同様のことが言える.

Table 3. Computational results for problem ND3

The Simplicial Algorithm $\alpha = 0.8$ and $M = (0.486 \cdot 10^6) \cdot I$					The Commodity Projection Method $\alpha = 0.8$ and $M = (0.486 \cdot 10^6) \cdot I$			
t	Rel. Gap	p_1^2	p_2^3	Time[1]	t	Rel. Gap	p_3^4	Time[1]
1	$.1941\ 10^{1}$	2	1	0.0856	1	$.1941\ 10^{1}$	2,2	0.0807
2	$.4185\ 10^{0}$	2	10	0.1181	5	$.6743\ 10^{-1}$	3,2	0.1801
3	$.8564\ 10^{-1}$	3	33	0.1567	10	$.6768\ 10^{-2}$	3,2	0.3052
4	$.6111\ 10^{-1}$	4	30	0.1963	15	$.2409\ 10^{-2}$	3,2	0.4306
5	$.1637\ 10^{-3}$	4	15	0.2318	20	$.9164\ 10^{-3}$	3,2	0.5571
6	$.6401\ 10^{-4}$	4		0.2604	25	$.3479\ 10^{-3}$	3,2	0.6827
			89		30	$.1326\ 10^{-3}$	3,2	0.7820
					31	$.9026\ 10^{-4}$	3,2	0.8575

Notes: 1: Time on IBM 3033

2: p_1 = # of active extreme aggregate flow patterns

3: p_2 = # of projections

4: p_3 = # of active extreme commodity flows for commodities 1 and 2

6. 計算結果

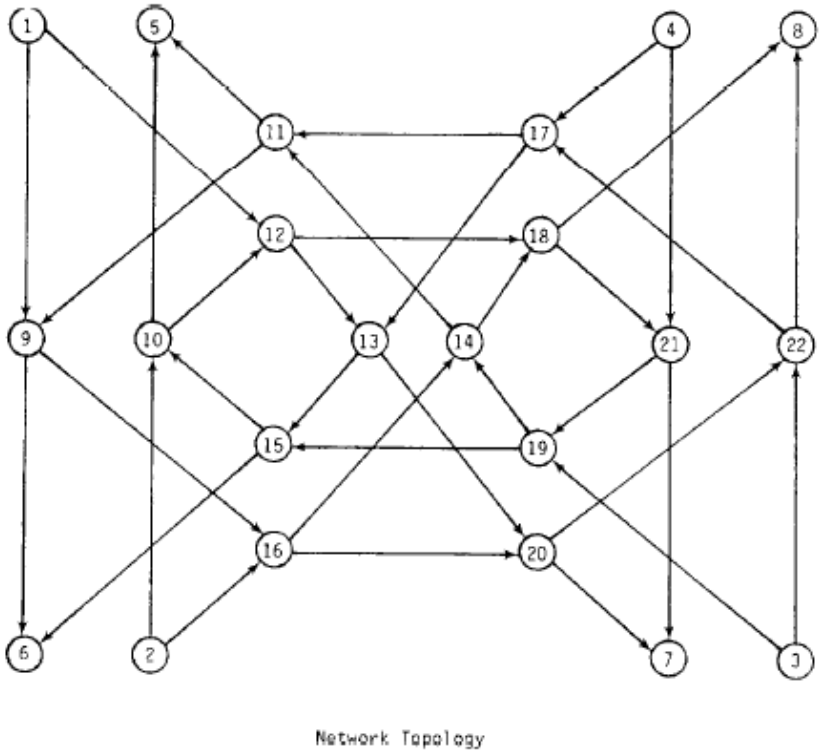
・問題 4

$4^{12} = 16{,}777{,}216$ パターン

	5	6	7	8
1	0	35	30	20
2	40	0	35	25
3	30	20	0	35
4	35	25	40	0

Trip Matrix

Fig. 2. Data for Problem 4.



Network Topology

6. 計算結果

Table 4. Cost vector for Problem 4

Arc	O	D	$C_i(x)$	Arc	O	D	$C_i(x)$
1	1	9	$f(x_1 + x_{14})$	19	14	11	$f(1/2\ x_{16} + x_{19} + x_{25} + 1/4\ x_{26})$
2	1	12	$f(x_2 + 1/2\ x_{11} + 1/4\ x_{12} + 1/4\ x_{14})$	20	14	18	$f(x_{16} + x_{20})$
3	2	10	$f(1/4\ x_{10} + x_3 + x_{21} + 1/4\ x_{22})$	21	15	10	$f(x_3 + x_{21})$
4	2	16	$f(x_{10} + x_4)$	22	15	16	$f(1/4\ x_{10} + 1/4\ x_3 + x_{22})$
5	3	19	$f(x_{34} + 1/4\ x_{33} + 1/4\ x_{32} + x_5)$	23	16	14	$f(x_{29} + 1/4\ x_{30} + 1/4\ x_{18} + x_{17} + x_{23})$
6	3	22	$f(x_{32} + x_6)$	24	16	20	$f(x_{18} + x_{24})$
7	4	17	$f(x_7 + x_{36})$	25	17	11	$f(x_{19} + x_{25})$
8	4	21	$f(x_8 + 1/2\ x_{27} + 1/4\ x_{28} + 1/4\ x_{36})$	26	17	13	$f(x_{15} + 1/4\ x_{16} + 1/4\ x_{19} + x_{20} + x_{26})$
9	9	6	$f(x_9 + x_{22})$	27	18	8	$f(1/4\ x_8 + x_{27} + x_{35} + 1/4\ x_{35})$
10	9	16	$f(x_{10} + 1/2\ x_3 + 1/4\ x_4 + 1/4\ x_{22})$	28	18	21	$f(x_8 + x_{28})$
11	10	5	$f(1/4\ x_2 + x_{11} + x_{13} + 1/4\ x_{14})$	29	19	14	$f(x_{29} + x_{23})$
12	10	12	$f(x_2 + x_{12})$	30	19	15	$f(x_{30} + 1/4\ x_{18} + 1/2\ x_{17} + 1/4\ x_{23})$
13	11	5	$f(x_{11} + x_{13})$	31	20	7	$f(x_{33} + x_{31})$
14	11	9	$f(x_1 + 1/4\ x_2 + 1/4\ x_{11} + x_{14})$	32	20	22	$f(1/4\ x_{33} + x_{32} + x_6 + 1/4\ x_5)$
15	12	13	$f(x_{15} + x_{26})$	33	21	7	$f(x_{33} + 1/2\ x_{32} + 1/4\ x_{31} + 1/4\ x_5)$
16	12	18	$f(x_{16} + 1/4\ x_{19} + 1/2\ x_{20} + 1/4\ x_{26})$	34	21	19	$f(x_{34} + x_5)$
17	13	15	$f(x_{30} + x_{17})$	35	22	8	$f(x_{27} + x_{35})$
18	13	20	$f(1/2\ x_{30} + x_{18} + x_{24} + 1/4\ x_{23})$	36	22	17	$f(x_7 + 1/4\ x_8 + 1/4\ x_{27} + x_{36})$

$$f(Z) = 3.0\ (1 + 0.15(Z/35)^4)$$

6. 計算結果

・パターン数が多い場合でもND2,ND3と同様のことが言える.

Table 5. Computational results for Problem 4

The Simplicial Algorithm: $\alpha = 0.8$ and $M = [1.2\ C(x^t)x^t] \cdot I$					The Commodity Projection Method: $\alpha = 0.8$ and $M = (0.6 \cdot 10^5) \cdot I$			
t	Rel. Gap.	p_1^2	p_2^3	Time[1]	t	Rel. Gap	p_3^4	Time[1]
1	$.3624\ 10^{0}$	2	23	0.1092	1	$.3624\ 10^{0}$	2,2,2,2	.1007
2	$.1018\ 10^{0}$	3	29	0.1543	10	$.2717\ 10^{0}$	2,3,2,2	.3587
3	$.4250\ 10^{-1}$	4	85	0.2440	20	$.1957\ 10^{0}$	2,3,2,2	.6445
4	$.2450\ 10^{-1}$	5	82	0.3409	30	$.1405\ 10^{0}$	2,3,3,3	.9286
5	$.1952\ 10^{-1}$	6	78	0.4434	40	$.1014\ 10^{0}$	2,3,3,3	1.2188
6	$.9912\ 10^{-2}$	5	161	0.6078	50	$.7753\ 10^{-1}$	2,4,3,3	1.5054
7	$.1489\ 10^{-2}$	5	208	0.8118	60	$.6118\ 10^{-1}$	2,4,3,3	1.7952
8	$.1082\ 10^{-2}$	6	300^5	1.1308	70	$.5441\ 10^{-1}$	2,5,3,3	2.0835
9	$.1222\ 10^{-2}$	7	174	1.3477	80	$.4896\ 10^{-1}$	2,5,3,3	2.3732
10	$.2853\ 10^{-3}$	7	300^5	1.7025	90	$.4351\ 10^{-1}$	2,5,3,3	2.6625
11	$.1133\ 10^{-2}$	8	300^5	2.0920	100	$.3908\ 10^{-1}$	2,5,3,3	2.9534
12	$.4488\ 10^{-3}$	9	300^5	2.5201	110	$.3539\ 10^{-1}$	2,5,3,3	3.2440
13	$.4447\ 10^{-3}$	10	193	2.8264	120	$.3227\ 10^{-1}$	2,5,3,3	3.5347
14	$.1071\ 10^{-3}$	8	300^5	3.2162	130	$.3008\ 10^{-1}$	2,6,3,3	3.8285
15	$.1116\ 10^{-2}$	9	169	3.4686	140	$.2829\ 10^{-1}$	2,6,3,3	4.1208
16	$.4216\ 10^{-3}$	10	249	3.8587	150	$.2657\ 10^{-1}$	2,6,3,3	4.4118
17	$.1822\ 10^{-3}$	11	186	4.1790	160	$.2522\ 10^{-1}$	2,6,3,3	4.4018
18	$.1498\ 10^{-3}$	12	180	4.5118	170	$.2390\ 10^{-1}$	2,6,3,3	4.9910
19	$.6774\ 10^{-4}$	12		4.5390	180	$.2269\ 10^{-1}$	2,6,3,3	5.2798
			3317		190	$.2159\ 10^{-1}$	2,6,3,3	5.5692
					200	$.2057\ 10^{-1}$	2,6,3,3	5.8599

Notes: 1: Time on IBM 3033
2: p_1 = # of active extreme aggregate flow patterns
3: p_2 = # of projections
4: p_3 = # of extreme commodity flows
5: 300 = max. allowable number of projections

6. 計算結果

・結論

非対称交通量配分問題の変分不等式計算において，従来の方法(Commodity射影法)と本論文で提案しているSimplicial Decompositionのアルゴリズムをそれぞれ用いて計算し比較した結果，ネットワークの複雑さ，つまりフローのパターン数の大小にかかわらず，Simplicial Decompositionの方が短い時間で最適解を得ることができる.