アルゴリズム 9－1
（1989. 7. 27）

平面ネットワーク最小費用流問題に対する内点法の高速化について

今井 浩

九州大学工学部情報工学科

平面ネットワークフロー問題に内点法を適用したときには、内点法の探索方向を求める際の線形方程式を解く部分で、有限要素法などで用いられていた線形計算における離散的手法を用いることができる。本稿では、この点を指摘し、実際に格子ネットワーク上の最小費用流問題を例に計算機実験を行なった。また、既存のネットワークシンプレックス法のプログラムを同じ問題に対して実行し、簡単な比較を行なった。

A Note on Implementing Interior Methods for Planar Minimum Cost Flow Problem

Hiroshi Imai

Department of Computer Science and Communication Engineering
Kyushu University 36, Fukuoka 812, Japan

A main issue in implementing interior methods for linear programming is to solve a system of linear equations repeatedly to find nice search directions. In solving this step of interior methods for the minimum cost flow problem on a planar network, which is a very special case of linear programming, a technique, called "nested dissection", which was originally developed to execute the finite element method efficiently, can be used. In this note, we point out this application, implement the dual affine scaling method with employing the nested dissection technique, and computationally investigate its efficiency. This interior method implemented with the nested dissection is partially compared with an existing code of the network simplex method.

1. はじめに

線形計画問題に対する内点法の実現では、線形方程式をコレスキー分解で解いて探索方向を求めるアプローチが多くとられている。このアプローチでは、線形方程式を解く部分を高速化することが重要である。線形方程式の高速解法は、数値計算の主要な問題の一つであり、これまでに多くの研究がなされている。従って、線形方程式の高速解法技法の多くがそのまま使え、実際にそのような技法を各所に用いた内点法のインプリメンテーションもされている (たとえば、 Alder, Karmarkar, Resende, Veiga [1,2])。

対称行列に関する線形方程式をコレスキー分解で解くときの中心的な問題の一つは、行と列の対称な入れ換えによりフィルイン (コレスキー分解の過程で行列の元々は 0 であった要素が 0 でなくなるところ) をできるだけ少なくするという離散的最適化問題である。線形計画問題を上述のようなアプローチで解くには、制約行列からまず解くべき線形方程式の行列を構成し、それを解くわけだが、ここで元の制約行列の疎構造が線形方程式の行列に反映され、さらにその疎構造がうまく少ないフィルインに結びついたとき、内点法の効率がたいへん良くなる。フィルインを最小にする問題は、 NP 困難であり [7]、フィルインをある程度少なくするための発見的解法が考えられている。 [1,2] では、代表的な二つの発見的解法を線形計画問題に適用し、比較している。一方、理論的に一般の線形計画問題を対象としたフィルイン最小化問題の議論を行なうことは、難しいと思われる。

本稿では、線形計画問題として平面ネットワーク上のフロー問題を考え、それに内点法を適用したとき、これまで有限要素法などに関連して考えられていたフィルイン最小化問題に関する線形計算技法 ([3,5]; nested dissection と呼ばれている) がそのまま利用できることを指摘する。このように線形計画問題を限定すると、対応するフィルイン最小化問題の (定数倍の違い以内での) 最適解がわかっており [3,5]、このネットワーク問題を、コレスキー分解を用いる内点法で解くアプローチの限界が明確になる。

以下では、2節で平面ネットワーク上での最小費用流問題について、3節で nested dissection についてまとめる。 4 節では、格子ネットワークを対象にして、内点法の双対アフィンスケーリング法を実現、計算機実験した結果について述べる。そこでは、一般の最小費用流問題に対して開発された既存のネットワークシンプレックス法のコード [4] と比較することも行なう。

2. ネットワークフロー問題と内点法

点集合 $\{v_1, v_2, \ldots, v_n\}$、有向枝集合 $\{a_1, a_2, \ldots, a_m\}$ からなる連結グラフ G で、容量ベクトル $\boldsymbol{c} = (c_j)$、費用ベクトル $\boldsymbol{d} = (d_j)$ が与えられ (枝 a_j の容量 c_j、費用 d_j)、流出・流入量ベクトル $\boldsymbol{b} = (b_i)$ (点 v_i で $b_i > 0$ のとき流出、$b_i < 0$ のとき $|b_i|$ 流入) が与えられたネットワーク N を考える。グラフ G の接続行列を A とする。すなわち、 $A = (a_{ij})$ は $n \times m$ 行列で、

$$a_{ij} = \begin{cases} 1 & \text{点} v_i \text{から枝} a_j \text{が出ている} \\ -1 & \text{点} v_i \text{に枝} a_j \text{が入っている} \\ 0 & \text{それ以外のとき} \end{cases}$$

流れベクトル $\boldsymbol{x} = (x_j)$ で枝 a_j 上の流れ x_j を表わすと、このネットワーク上での最小費用流問題は、次のように表わせる。

$$\begin{aligned} \text{minimize} \quad & \boldsymbol{d}^{\mathrm{T}}\boldsymbol{x} \\ \text{s.t.} \quad & A\boldsymbol{x} = \boldsymbol{b} \\ & 0 \leq \boldsymbol{x} \leq \boldsymbol{c} \end{aligned} \tag{P}$$

n 点よりなる連結グラフ G の接続行列の階数は $n-1$ である。このネットワークに 1 点人為的な点 v_0 と、その点 v_0 と各点 v_i を両方向に結ぶ人為的な有向枝各 2 本 (容量 $+\infty$ で、費用も十

分大きいとする) を付け加えたネットワークを考え、その接続行列の内、点 v_0 に関する行を除いた行列 $\widetilde{A}$ を用いて、次の等価な問題を考える。

$$\begin{aligned} \text{minimize} \quad & \widetilde{\boldsymbol{d}}^{\mathrm{T}}\widetilde{\boldsymbol{x}} \\ \text{s.t.} \quad & \widetilde{A}\widetilde{\boldsymbol{x}} = b \\ & 0 \le \widetilde{\boldsymbol{x}} \le \widetilde{\boldsymbol{c}} \end{aligned} \tag{P'}$$

但し、$\widetilde{A} = (A \mid I \mid -I)$ (I は $n \times n$ 単位行列)、$\widetilde{\boldsymbol{c}} = (c_1, \ldots, c_{m+2n})^{\mathrm{T}}$、$c_j = +\infty \quad (j = m+1, \ldots, m+2n)$、$\widetilde{\boldsymbol{d}} = (d_1, \ldots, d_{m+2n})^{\mathrm{T}}$、$d_j = M \quad (j = m+1, \ldots, m+2n)$、$M$ は十分大きな正の数とする。行列 $\widetilde{A}$ の階数は n である。

内点法の多くのアルゴリズムでは、$\widetilde{A}$ と $(m+2n) \times (m+2n)$ の対角行列 D を用いて

$$B = \widetilde{A} D \widetilde{A}$$

で定められる $n \times n$ の対称行列Bに関する線形方程式を解く。B の ij 要素は、元のグラフ G で点 v_i, v_j 間に枝がないときには、必ず 0 であり、枝があるときには通常 0 でない。すなわち、B の非零パタンは、グラフ G の隣接行列と通常同じであり、隣接行列で0のところは B で必ず 0 である。

3. 平面グラフの隣接行列と同じ非零パタンの行列のコレスキー分解

$n \times n$ 対称行列 B に関する線形方程式を解くとき、B の非零パタンが格子グラフ、平面グラフの隣接行列と（ある程度）同じなら、最小フィルイン数の高々定数倍のフィルイン数で線形方程式を解く方法が知られている [3,5]。この方法によれば、$O(n \log n)$ のフィルイン数で、$O(n^{1.5})$ の手間で方程式が解ける。この方法の必要とする記憶容量の大きさは、フィルイン数に比例し、また行・列のピボット順を計算する部分は、$O(n \log n)$ の手間で行える。また、この方程式をコレスキー分解で解くときには、$n^{1.5}$ に比例する以上の演算が必要であることも示されており、その意味で最適である。

$k \times k$ の格子グラフを例にすると、図3.1に示すようなピボット順をこの方法は用いる。この順番は、まず $k \times k$ 格子をまん中の行・列によりほぼ 1/4 の大きさの格子 4 つに分割し、今選んだ行・列の点の番号を他の点よりも大きくし、4 つの部分格子各々に対して再帰的に（かつ図ではバランスよく）この手続きを適用していくことにより得られる。実際のフィルイン数を表3.1に示す。平面グラフの場合には、いわゆる平面グラフ分割定理 [6] を用いて同様なことを行なう。

ここまでは、平面ネットワーク上での最小費用流問題のみを考え、それに対して nested dissection の方法が適用できることを述べてきたが、もう少し議論を続けていき、また適当な仮定をおくことにより、平面上での多品種流問題等にも同様のアプローチをすることができる。

4. 格子ネットワークでの計算機実験

2, 3 節の方法をプログラミングし、計算機実験を行なった。対象としては、$k \times k$ 無向格子ネットワーク (従って、点数 $n = k^2$) 上での容量制限無しの最小費用流問題を考え、内点法としては、問題 (P′) の双対問題に対するアフィンスケーリング法 (たとえば [1]) を考えた。無向ネットワークは、各無向枝を両方向の有向枝 2 本で置き換えることにより有向ネットワークの問題に変換できる。容量制限無しとしているので、双対問題は次のように表わせる。

$$\begin{aligned} \text{maximize} \quad & \boldsymbol{b}^{\mathrm{T}}\boldsymbol{y} \\ \text{s.t.} \quad & \widetilde{A}^{\mathrm{T}}\boldsymbol{y} \le \widetilde{\boldsymbol{d}} \end{aligned}$$

1	27	173	3	33	299	7	49	190	54	381	16	103	235	108	335	19	121	250	126
26	28	172	32	34	298	48	50	189	55	380	102	104	234	109	334	120	122	249	127
171	170	178	176	177	297	188	187	194	193	379	233	232	239	238	333	248	247	254	253
2	30	174	4	36	296	8	52	191	56	378	17	106	236	110	332	20	124	251	128
29	31	175	35	37	295	51	53	192	57	377	105	107	237	111	331	123	125	252	129
294	293	292	291	290	308	304	305	306	307	376	330	329	328	327	344	340	341	342	343
5	39	182	6	44	300	9	59	198	63	375	18	113	243	117	336	21	131	258	135
38	40	181	43	45	301	58	60	197	64	374	112	114	242	118	337	130	132	257	136
180	179	186	184	185	302	196	195	201	200	373	241	240	246	245	338	256	255	261	260
41	42	183	46	47	303	61	62	199	65	372	115	116	244	119	339	133	134	259	137
371	370	369	368	367	366	365	364	363	362	400	391	392	393	394	395	396	397	398	399
10	67	205	11	72	317	14	87	221	91	382	22	139	265	143	352	24	155	279	159
66	68	204	71	73	316	86	88	220	92	383	138	140	264	144	351	154	156	278	160
203	202	209	207	208	315	219	218	224	223	384	263	262	268	267	350	277	276	282	281
69	70	206	74	75	314	89	90	222	93	385	141	142	266	145	349	157	158	280	161
313	312	311	310	309	326	322	323	324	325	386	348	347	346	345	361	357	358	359	360
12	77	213	13	82	318	15	95	228	99	387	23	147	272	151	353	25	163	286	167
76	78	212	81	83	319	94	96	227	100	388	146	148	271	152	354	162	164	285	168
211	210	217	215	216	320	226	225	231	230	389	270	269	275	274	355	284	283	289	288
79	80	214	84	85	321	97	98	229	101	390	149	150	273	153	356	165	166	287	169

図 3.1. 20×20 格子ネットワークでの nested dissection によるピボット順

表 3.1. nested dissection によるピボット順でコレスキー分解した時の上 (下) 三角行列の非零数 (対角要素を含む)

k	20	30	40	50	60	70	80	90	100	110	120
n	400	900	1600	2500	3600	4900	6400	8100	10000	12100	14400
非零数	5008	14372	29195	50968	79789	115770	159164	211639	272009	342133	419592

初期実行可能解は、$y = 0$ を用いた (費用ベクトル $\tilde{d}$ を下記のように正にしていることから、これで実行可能な内点となる)。直線探索では、制約に当たるまでの 0.99 進むという方式を取った。また、停止条件は、後述するネットワークシンプレックス法のコードとの比較もあり、各ステップの目的関数値の増加がその時点での目的関数値の 10^{-5} 未満になれば停止するというものを用いた。

費用 d、流出・流入量ベクトル b に関しては、次の 2 つの場合を考えた。

(Case 1) 各枝の費用は、1 から 10^5 までのランダムな整数、$k \times k$ 格子の i 行、j 列の点の流出・流入量を i $(i \leq k/2)$、$i - k$ $(i > k/2)$ とした;

(Case 2) 各枝の費用は、$10^5 \pm 10^2$ の間の整数、$k \times k$ 格子の i 行、j 列の点の流出・流入量を j $(i = 1)$, i $(1 < i \leq k/2)$, $i - k$ $(k/2 < i < k)$, j $(i = k)$ とした;

また、M は 10^3 とした。

プログラムは FORTRAN で行なった。実数計算は、すべて倍精度で行なっている。アフィンスケーリング法のプログラムでも、一部格子ネットワークに特別なデータ構造などを用いており、一方コレスキー分解の部分のプログラムにはまだ少し改良の余地がある。計算機実験に用いた計算機は、VAX8800 (Ultrix V2.3) で、f77 コマンドで -O(最適化) を指定してコンパイルした後実行した。表 4.1, 4.2 に計算機実験の結果を示す (本来ならグラフ等を用いて、よりわかりやすく示すべきであるが、ここではまだ内点法としてアフィンスケーリング法だけ扱っ

表 4.1. $k \times k$ 格子ネットワーク最小費用流問題での計算実験結果: (Case 1)

問題の大きさ		アフィンスケーリング:A		シンプレックス: S		計算時間比
k	n	反復回数	時間 (s)	反復回数	時間 (s)	S/A
20	400	10	2.39	458	0.43	0.18
30	900	11	9.44	1007	1.60	0.17
40	1600	10	21.19	1788	4.18	0.20
50	2500	11	46.46	2816	9.28	0.20
60	3600	11	82.23	4094	17.71	0.22
70	4900	11	134.63	5510	29.94	0.22
80	6400	11	203.86	7345	49.63	0.24
90	8100	11	297.60	9098	74.95	0.25
100	10000	11	415.74	11528	111.38	0.27
110	12100	11	564.49	13933	162.27	0.29
120	14400	11	737.56	16987	237.15	0.32

表 4.2. $k \times k$ 格子ネットワーク最小費用流問題での計算実験結果: (Case 2)

問題の大きさ		アフィンスケーリング:A		シンプレックス: S		計算時間比
k	n	反復回数	時間 (s)	反復回数	時間 (s)	S/A
20	400	12	2.88	942	0.98	0.34
30	900	11	9.38	2482	4.83	0.51
40	1600	11	23.12	5186	16.12	0.70
50	2500	12	50.92	9298	37.15	0.73
60	3600	12	90.73	14412	78.23	0.86
70	4900	12	149.55	21304	146.48	0.98
80	6400	12	226.77	30532	245.02	1.08
90	8100	12	330.40	38923	382.85	1.16
100	10000	12	461.52	53161	593.52	1.29
110	12100	12	627.15	66526	902.77	1.44
120	14400	12	822.51	86740	1273.25	1.55

ていることでもあり、総合的な報告は別の機会にゆずることとする)。計算機時間には、問題の入出力の時間は含まれていない。(Case 1) については、費用の乱数を 2 タイプ考え、その平均を示している。

また、同時に Kennington, Helgason [4] による一般の最小費用流問題に対するネットワークシンプレックス法のコードを [4] より入力し、問題 (P) に対して適用し、参考のため比較した。このコードは、一般の最小費用流問題に対するものであり、汎用性がある。ただ、このコードがネットワークフローの種々のコードの中で、最高速であるというわけでもない [4]。また、比較の際には、アフィンスケーリング法のプログラムでの停止条件との兼ね合いなどを詳細に考えなければいけない。従って、ここでの比較では、あくまでもネットワークの点数を増やしていったときの両コードの要する計算機時間の増加のオーダなどの点に注目すべきである。また、費用ベクトルなどについて 2 つの場合を考えているが、この (Case 1) と (Case 2) では、ネットワークシンプレックス法のコードの方が、性能に大きな違いが生じる例となっており、これだけで一般的なことがいえるわけではない。

アフィンスケーリング法のコードは、問題の大きさに関わらずほぼ一定 (10 ~ 12) の反復回

数しか要さない。各反復が、線形方程式を解くところで$O(n^{1.5})$かかっていることから、全体でほぼ$n^{1.5}$に比例した時間がかかっている(一部、コレスキー分解のプログラムでの不備な点の影響がでているが)。また、この(Case 1), (Case 2)に関してはあまり差がない。一方、ネットワークシンプレックス法のコードは、$n^{1.8\sim 2}$に比例した時間がかかっており、nが莫大となれば、この停止基準の下では、アフィンスケーリング法のコードの方がいつか早くなることが見て取れる(また、この例に関しては(Case 2)で実際にそうなっている)。停止基準については、対象としているものがネットワーク問題であるので、その性質を用いて最適性の判定を行なった上での比較が本来は望まれる。記憶領域の大きさに関しては、アフィンスケーリング法のコードは2節で示したフィルイン数の配列を必要とするため、かなりシンプレックス法のコードに比べて多くを必要とする。

5. おわりに

内点法に関して報告されている計算機実験の結果では、コンピュータネットワークで入手可能なNETLIBの問題が多く用いられている(たとえば[1]で用いられている)。そこでの問題と比べると、格子ネットワーク上の最小費用流問題は、制約行列の各列の非零数が高々2というよい性質をもっている反面、フィルイン数がシンプレックス法等と比べて相対的にかなり多いという内点法にとって不都合な性質もある。4節の実験で、作業領域としてはかなり多くを使っても、(case 2)などでそれでもシンプレックス法より速くなる場合があるというのは興味深い。このように、この問題では、内点法は通常大規模になる程計算時間に関しては有利になるが、一方作業領域の大きさに関しては不利になるという面が顕著に現われている。

総合的には、最小費用流・最大流などの純粋なネットワークフロー問題に対しては内点法が他の(ネットワークシンプレックス法を含め)組合わせ的アルゴリズムに取って代わるのは難しいかもしれない。しかし、Adler, Karmarkar, Resende, Veiga [1]が、一般的なアフィンスケーリング法のコードで、多品種流問題専用のシンプレックスコードとほぼ同等の性能を得たとしていることからも、多品種流問題などに対しては、より有効であるかもしれない。

謝辞

本研究の一部は、平成元年度稲森財団研究助成金の援助を受けた。

参考文献

[1] I. Adler, N. Karmarkar, M. G. C. Resende and G. Veiga: An Implemetation of Karmarkar's Algorithm for Linear Programming. Manucript, May 1986, revised June 1987.

[2] I. Adler, N. Karmarkar, M. G. C. Resende and G. Veiga: Data Structures and Programming Techniques for the Implementation of Karmarkar's Algorithm. Manuscript, December 1987, revised November 1988.

[3] A. George: Nested Dissection of a Regular Finite Element Mesh. *SIAM Journal on Numerical Analysis*, Vol.10, No.2 (1973), pp.345–363.

[4] J. L. Kennington and R. V. Helgason: *Algorithms for Network Programming*. John Wiley & Sons, New York, 1980.

[5] R. J. Lipton, D. J. Rose and R. E. Tarjan: Generalized Nested Dissection. *SIAM Journal on Numerical Analysis*, Vol.16, No.2 (1979), pp.346–358.

[6] R. J. Lipton and R. E. Tarjan: A Separator Theorem for Planar Graphs. *SIAM Journal on Applied Mathematics*, Vol.36, No.2 (1979), pp.177-189.

[7] M. Yannakakis: Computing the Minimum Fill-In is NP-Complete. *SIAM Journal on Algebraic and Discrete Methods*, Vol.2, No.1 (1981), pp.77–79.