主双対内点法に対する高速化・並列化の技術

山下　真

ここでは，2010年現在のSDPソルバーによって，どの程度の規模のSDPをどの程度の時間で解くことができるかを紹介する．主双対内点法の発展に伴うSDPソルバーの開発の歴史は，すでに15年に達しており，SDPの性質によっては100倍以上の高速化が達成されている．また，並列計算の導入で，これまで求解できなかった制約が数万に達する超大規模SDPを実際に解くことが可能になってきている．特に，並列計算ソルバーSDPARAは，超大規模SDPを解く際に，主双対内点法の性質にあわせて複数の並列計算手法を組み合わせており，数理計画分野とHigh Performance Computingの融合として捉えても興味深い．

キーワード：半正定値計画問題，主双対内点法，ソフトウェア，High Performance Computing，並列計算

1. ソルバー開発によるSDP高速求解への挑戦

SDPソルバーは，福田光浩氏の記事にあるように主双対内点法を実装した代表的なものだけでも，CSDP，SDPA，SDPT3，SeDuMiなどがある．これらは，1995年のSDPAの開発以降，お互いに刺激を受けながら開発が進んでおり，15年以上におよぶ開発を経て，各SDPソルバーは大きな高速化を達成している．表1では，SDPAがバージョンアップに伴ってどれだけの計算時間短縮を実現してきたかをまとめた．数値実験環境としては，Xeon E5520 (2.27 GHz) を2個と24 GBの搭載メモリのLinux (RHEL5) である．なお，表1ではSDPAのみを扱っているが，これは著者らのグループがSDPAを開発してきたため，2000年以前の旧バージョンも数値実験に利用できたことによる．各SDPソルバーの最新バージョンが計算時間については，Mittelmannのベンチマークのページ[1]で詳細な結果を確認することもできる．

なお，control10, gpp500-1は，SDPの標準的なベンチマーク問題集であるSDPLIB[1]からの引用であるが，SDPLIBが整備されたのは1999年のことであり，近年のSDPソルバーではほとんどの問題が100秒程度で解けてしまう．ここ数年は，Mittelmannのページ[1]にあるより大規模なベンチマークが標準的なベンチマークとなりつつあり，NH$^+$，nonc_500，yalsdpは，ここからの引用である．

1.1 高速化達成の原動力

SDPソルバーの高速化を理解するには，アルゴリズム的な視点とハードウェアからの視点の両輪が同時に必要であるが，ここではアルゴリズムからの視点に重きを置く．ハードウェアからの視点は藤澤克樹氏の記事が詳しいので，そちらを参照してほしい．

SDPソルバーが表1にある大きな高速化を達成したのは，アルゴリズム面では以下の4つの要素によるところが大きい．

1. 入力行列 $F_1, \cdots, F_m$ の疎性を活用し，Schur補完行列の要素計算の高速化[Ver 2⇒3]
2. BLAS/LAPACKの導入による密行列演算の高速化[Ver 5⇒6]
3. Schur補完行列が疎行列となった場合に，疎コレスキー分解の採用[Ver 6⇒7]

表1　SDPAのバージョンと計算時間（秒）

Version	開発年	control10	gpp500-1	NH$^+$	nonc_500	yalsdp
2	1996	683.11	1199.03	3039.70	66891.09	3304.66
3	1997	328.21	1830.92	397.07	14369.86	1534.70
4	1998	118.77	72.54	310.42	30650.07	1233.32
5	1999	132.31	30.96	189.47	35328.96	1154.81
6	2002	48.26	2.77	92.24	789.92	315.45
7	2009	19.22	2.07	18.87	5.35	86.99
Ver 2から7での高速化		35倍	579倍	161倍	12503倍	37倍

やました　まこと
東京工業大学 大学院情報理工学研究科数理・計算科学専攻
〒152-8552 目黒区大岡山2-12-1

[1] http://plato.asu.edu/ftp/sdp/00README

4. Schur補完行列の要素計算のマルチスレッド化による並列計算[Ver 6⇒7]

この4つの要素は，いずれもSchur補完行列に関係しているが，これは一般にSchur補完行列の計算時間が主双対内点法の計算時間の80%以上に達するためである．中田和秀氏の記事に主双対内点法の枠組は記されているが，Schur補完行列は最適解に近づけるためにLagrange乗数をどれだけ変化させるか，を計算するための連立方程式の係数行列であるため，主双対内点法での核となり，計算時間も集中しやすい．

なお，括弧書きしたのはSDPAのどのバージョンアップに対応しているかであり，これらの要素は他のSDPソルバーにも取り入れられている．例えば，1の疎性の活用については，SDPAに最初に実装された[2]が，その後CSDP，SDPT3にも導入されている．逆に3の疎コレスキー分解はSeDuMi，SDPT3がMatlabの関数として利用していたが，SDPAではC++から直接利用できるMUMPS[2]のライブラリにより導入した形である．また，4のマルチスレッド化はSDPAの最新技術でもあり，他のSDPソルバーが今後追随する可能性が高い．

以下では，この4つの要素について順次触れていきたい．

1.1.1 入力行列の疎性の活用

Schur補完行列 $\boldsymbol{B}$ の各要素は，

$$B_{ij}=(\boldsymbol{Y}\boldsymbol{F}_i\boldsymbol{X}^{-1})\cdot\boldsymbol{F}_j \tag{1}$$

で計算されるが，$\boldsymbol{F}_i$，$\boldsymbol{F}_j$ の要素に着目すると以下のように書き下すこともできる．

$$B_{ij}=\sum_{\gamma=1}^{n}\sum_{\epsilon=1}^{n}\left(\sum_{\alpha=1}^{n}\sum_{\beta=1}^{n}[\boldsymbol{Y}]_{\gamma\alpha}[\boldsymbol{F}_i]_{\alpha\beta}[\boldsymbol{X}^{-1}]_{\beta\epsilon}\right)[\boldsymbol{F}_j]_{\gamma\epsilon} \tag{2}$$

ただし，$[\boldsymbol{F}_i]_{\alpha\beta}$ は $\boldsymbol{F}_i$ の (α,β) 要素である．この式の場合，入力行列 $\boldsymbol{F}_i$ と $\boldsymbol{F}_j$ の要素のほとんどが0であった場合（つまり，疎行列であった場合），それらの計算を省略し，非ゼロ要素の計算だけを行えば計算時間を短縮できる．実際，組合せ最適化などに代表されるSDP緩和では，入力行列が疎行列になる場合が多く，大幅な計算時間短縮が実現される．

表1の NH^+ では，Ver 2からVer 3になるときに大幅な計算時間短縮となっている．これは，入力行列 $\boldsymbol{F}_1,\cdots,\boldsymbol{F}_{948}$ の平均非ゼロ要素が133個のみと，完全に密な状態（約24万）に対して極めて疎な状態であり，文献[2]による効果が著しく現れるためである．

[2] http://mumps.enseeiht.fr/

1.1.2 BLAS/LAPACKの導入

主双対内点法は，密行列の計算も多く含まれる．例えば，前述の $B_{ij}=(\boldsymbol{Y}\boldsymbol{F}_i\boldsymbol{X}^{-1})\cdot\boldsymbol{F}_j$ も $\boldsymbol{F}_i$, $\boldsymbol{F}_j$ が疎行列でない場合には，$\boldsymbol{Y}\boldsymbol{F}_i\boldsymbol{X}^{-1}$ に密行列同士の積が2回，さらに $\boldsymbol{F}_j$ との密行列同士の内積計算が1回必要である（一般に，主双対内点法では変数行列 $\boldsymbol{X}$，$\boldsymbol{Y}$ を密行列形式でメモリ上に格納している）．他にも，$\boldsymbol{X}$，$\boldsymbol{Y}$ のコレスキー分解などが必要である．

密行列の数値計算ライブラリとしては，行列積など基本的な演算を扱うBLAS（Basic Linear Algebra Sets）[3]やBLASを用いて連立方程式の求解などを行うLAPACK（Linear Algebra PACKage）[4]がある．これらのライブラリは，CPUキャッシュやメモリ構造などの性質を積極的に利用することで，CPUの理論ピークに近い性能を達成している．

SDPA，CSDPでは，ATLASやIntel MKLといった種類のBLASを利用できるようになっている．また，Matlab上のソフトウェアであるSDPT3，SeDuMiも，内部的にはMatlabが呼出しているIntel MKLで密行列演算を処理している．

表1のcontrol 10で，Ver 5からVer 6で計算時間が2.5分の1に短縮されるのは，BLAS/LAPACKを適切に利用できている効果である．

1.1.3 疎コレスキー分解の採用

Schur補完行列 $\boldsymbol{B}$ は，主双対内点法の枠組では必ず正定値行列となるため，下三角行列 $\boldsymbol{L}$ で $\boldsymbol{B}=\boldsymbol{L}\boldsymbol{L}^T$ とコレスキー分解できる．これにより，連立方程式 $\boldsymbol{B}d\boldsymbol{x}=\boldsymbol{r}$ を解く際に，$\boldsymbol{L}^T\widetilde{d\boldsymbol{x}}=\boldsymbol{r}$ の後進代入，$\boldsymbol{L}d\boldsymbol{x}=\widetilde{d\boldsymbol{x}}$ の前進代入となり，Gaussの消去法やLU分解よりも短時間で $d\boldsymbol{x}$ を求められる．

量子化学などから生じるSDPでは一般に，$\boldsymbol{B}$ がすべての要素で0ではない，完全な密行列となるため，LAPACKのルーチンを呼ぶことが可能である（図1）．その一方，センサーネットワーク問題のSDPでは，ほとんどの要素が0となり（図2），これに対応した疎コレスキー分解を適用する必要がある．なお，図1，2では，$\boldsymbol{B}$ が対称行列であることから，上三角のみ非零要素の位置を示している．

SDPAでは，Ver 7からMUMPSによる疎コレスキー分解が導入されたが，この効果は，nonc_500で見て取れる．nonc_500の場合，Schur補完行列の非

[3] http://www.netlib.org/blas/

[4] http://www.netlib.org/lapack/

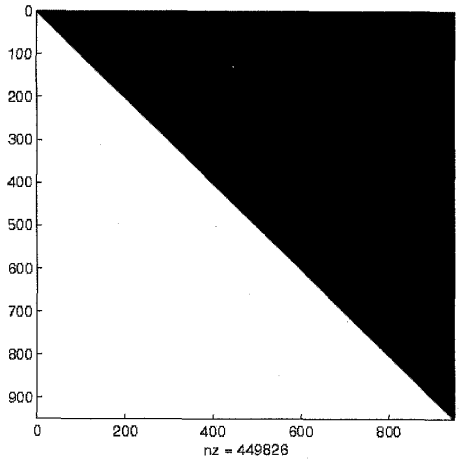

図1　量子化学から発生するSchur補完行列の非零要素の位置

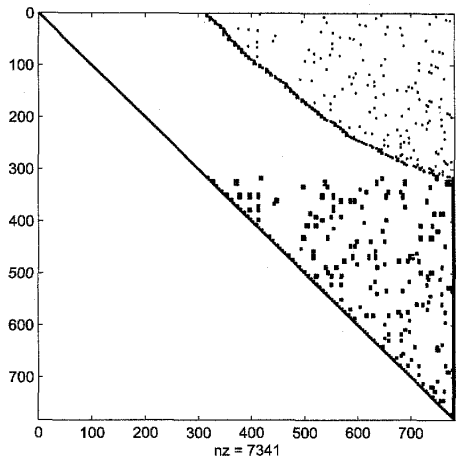

図2　センサーネットワーク問題から発生するSchur補完行列の非零要素の位置

ゼロ要素数は，89,732個しかなく，これは，m^2＝約2,500万に対して，0.36%である．

1.1.4　マルチスレッド化

近年のCPUは，ネットブックに搭載されているAtomでも一部マルチコア化が進んでおり，並列計算がより身近になっている．後述のSDPARAでは複数のPCからなるクラスタで並列計算を実行するように実装しているが，SDPAではVer 7より，この実装の一部を取り込み，Schur補完行列の計算時間がマルチスレッドでの並列計算で大幅に短縮されている．

Schur補完行列は，$B_{ij}=(\boldsymbol{Y}\boldsymbol{F}_i\boldsymbol{X}^{-1})\cdot\boldsymbol{F}_j$で評価されるが，各$i$行では$\boldsymbol{Y}\boldsymbol{F}_i\boldsymbol{X}^{-1}$の積計算のあとに$\boldsymbol{F}_j$との内積計算となる．つまり，各行の計算は完全に独立しており，行ごとに違うスレッドに計算させることが可能である．SDPARAでは，このような分割をrow-wise distributionと呼んでいる．例えば，図3は$\boldsymbol{B}$

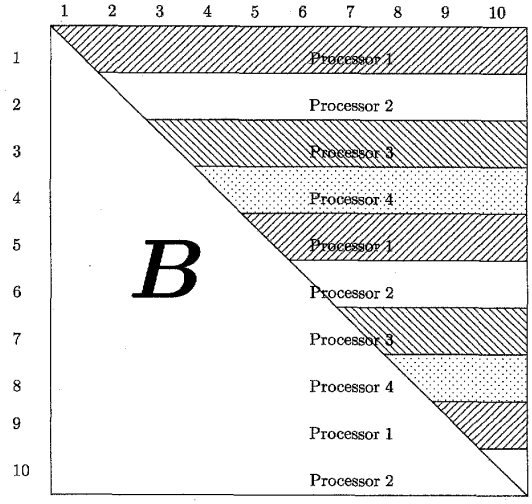

図3　row-wise distribution

表2　各ソルバーによる計算時間

問題名	CSDP	SDPA	SDPT3	SeDuMi
mater-6	10409.33	108.78	146.14	187.73
neu1g	385.11	103.29	374.69	211.77
shmup4	2409.69	343.36	1065.30	35774.77
t_texture	2183.78	124.39	280.31	35494.91

が10×10行列だったとして，4コア利用できるときを示している（ただし，$\boldsymbol{B}$は対称行列であるため，上三角だけを評価している）．

マルチスレッド化により，Schur補完行列の評価が94%を占めるNH^+のような問題をSDPA Ver 7では高速に解くことができる．

1.2　数値実験結果

表2では，現状のソルバー（SDPA，SeDuMi，CSDP，SDPT 3）でどれほどの規模の問題がどの程度の時間で解けるか，についてまとめてみた．表2で扱った問題の規模は，表3の通りである．SDPの問題の規模は，大まかにいうと，m（入力行列の個数），n（変数行列の次元），nnz（入力行列の非ゼロ要素の数の総和），sd（Schur補完行列の非ゼロ要素の密度）で表される．また，nの列に括弧書きしてある数字は，対角ブロック表現したときの最大ブロックの次元である．

なお，表1で扱った問題の規模は表4にまとめてある．表3と表4は，現状のSDPソルバーが標準的に求解できる規模の目安として見ることもできる．

2.　並列計算による大規模SDPへの挑戦

複数のPCを高速ネットワークで接続することで並列計算環境を構築するPCクラスタの導入が進み，大

表3　SDP問題の規模

問題名	m	n	nnz	sd
mater-6	20463	54628(11)	1325669	0.20%
neu1g	3002	252(252)	63503	100%
shmup4	800	4961(1681)	89520	100%
t_texture	1802	1801(1801)	14402	100%

表4　表1のSDP問題の規模

問題名	m	n	nnz	sd
control10	1326	150(100)	622700	100%
gpp500-1	501	500(500)	250500	100%
NH^+	948	1406(306)	126396	100%
nonc_500	4990	2998(6)	17479	0.36%
yalsdp	5051	300(100)	2000200	100%

規模な数理計算問題を実際に解く環境が近年整ってきた．並列SDPソルバーとしては，福田光浩氏の記事にまとめられているようにSDPARA[7]やPCSDP[3]がある．主双対内点法の場合，ほとんどの計算時間はSchur補完行列に費されるため，この2つの並列ソルバーの並列化はSchur補完行列の計算に集約されている．しかし，最新のSDPARAは1.1.3節にある疎なSchur補完行列に対しても並列計算を適用でき，この点でPCSDPよりも優れている．ここでは，SDPARAの並列化について解説した後，複数PCでどの程度の高速化が得られるのか，また，どの程度の規模まで解くことができるのかをまとめることとしたい．

2.1　SDPARAにおける並列計算

Schur補完行列に関する計算は，その要素の評価とコレスキー分解であるが，SDPARAでは，これらのボトルネックをELEMENTSとCHOLESKYと呼んでいる．

Schur補完行列が完全に密行列となっている場合，ELEMENTSの計算は1.1.4節で取り上げたようにrow-wise distributionによって行う．つまり，N台のPCがあるときにp番目のPCは，$\boldsymbol{B}\in\mathbb{S}^m$のうち，$\mathcal{R}_p=\{i: 1\le i\le m, (i-1)\%N=p-1\}$の行を評価する．ただし，$a\%b$は$a$を$b$で割ったときの余りである．このように分割することで，p番目のPCがメモリ上に保存すべき$\boldsymbol{B}$の領域も明確になる．したがって，仮にmが数万を超えて大きくなり，$\boldsymbol{B}$が1台のPCのメモリ容量を超えたとしても，複数PCに分散保持できるため，さらに大きな$\boldsymbol{B}$を保持できるようになっている．

Schur補完行列が疎行列となる場合は，ELEMENTSの計算割り当ては複雑になる．図2からも簡単に予想がつくように，row-wise distributionでは各PCの計算負荷にバラツキが大きくなり，より多くの要素を割り当てられたPCの計算時間に全体の計算時間が律速されてしまう．

SDPARAでは，計算時間を計算式から予測することで，計算割り当てを算出する仕組みを導入している．要素B_{ij}は，入力行列$(\boldsymbol{F}_i, \boldsymbol{F}_j)$の疎性に応じて(1)式，あるいは(2)式で評価されるが，これらの計算式は積や和で構成されている．特に，$\boldsymbol{F}_i$と$\boldsymbol{F}_j$の非零要素の数をカウントすることで積の回数を見積もることができる．つまり，(1)式と(2)式にかかる計算コストは，$\boldsymbol{F}_i$と$\boldsymbol{F}_j$の情報によって，ある程度予測することができる．この計算コスト予測をすべてのB_{ij}に適用し，その結果を計算割り当てに用いる．B_{ij}にかかる計算コストをc_{ij}としたとき，全体のコストCは$C=\sum_{i=1}^{m}\sum_{j=i}^{m} c_{ij}$であり，$N$台のPCでは平均して各PCが$C/N$だけの計算を受け持つ．1台目のPCが担当する範囲$\varepsilon_1$は，$B_{11}$要素から始めて，行方向に計算コストの和が$C/N$を超えない最大範囲が基本となる．具体的に記せば，

$$\varepsilon_1=\{(i,j): 1\le i<\bar{i}, i\le j\le m\}\cup\{(\bar{i},j): \bar{i}\le j\le\bar{j}\}$$

となる．ただし，$\bar{i}, \bar{j}$は以下の式を満たすものである．

$$\sum_{i=1}^{\bar{i}-1}\sum_{j=i}^{m} c_{ij} < C/N \le \sum_{i=1}^{\bar{i}}\sum_{j=i}^{m} c_{ij}$$

$$\sum_{i=1}^{\bar{i}-1}\sum_{j=i}^{m} c_{ij} + \sum_{j=\bar{i}}^{\bar{j}-1} c_{\bar{i}j} < C/N \le \sum_{i=1}^{\bar{i}-1}\sum_{j=i}^{m} c_{ij} + \sum_{j=\bar{i}}^{\bar{j}} c_{\bar{i}j}$$

同様にして，$\varepsilon_2, \cdots, \varepsilon_N$を順次決定し，1台目から$N$台目までの計算割り当てとする．

次に，CHOLESKYについて見ていくこととする．Schur補完行列が密行列となる場合は，ScaLAPACK[5]が提供している並列コレスキー分解を利用する．これについては，PCSDPも同様である．ただし，ELEMENTSの計算が終了した時点でSDPARAのSchur補完行列はrow-wise distribtuionで各PCのメモリ上に分散している．SDPARAでは，2次元巡回ブロック分散に分散しなおすことで，ScaLAPACKの性能を引き出すことに成功している．図4は図3を2次元巡回ブロック分散に再分散した状態であり，B_{12}, B_{59}は1台目のPCが担当し，B_{43}, B_{78}は4台目のPCが担当する．

[5] http://www.netlib.org/scalapack/

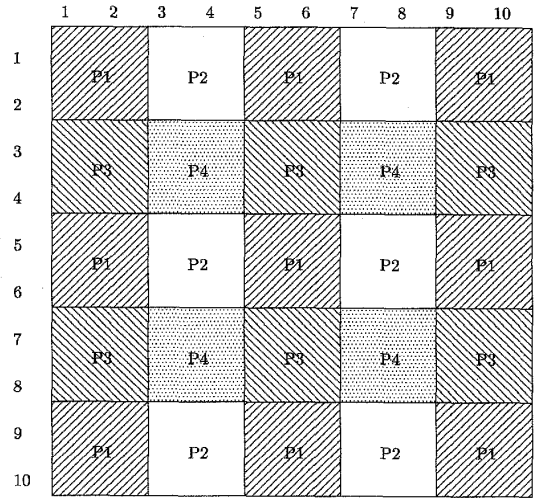

図4　2次元巡回ブロック分散

一方，Schur補完行列が疎行列となる場合は，MUMPSの並列疎コレスキー分解を適用する．MUMPSでは，上記の$\varepsilon_1, \varepsilon_2, \cdots, \varepsilon_N$のメモリ分散をそのまま利用できるため，メモリ上で再分散を行う必要はない．

ここまで見てきたように，SDPARAではSchur補完行列の計算方法をさまざまに変更することで，Schur補完行列が密行列であっても疎行列であっても，効率的な並列計算を実現している．

2.2　SDPARAによる数値実験

ここでは，まずPCの台数が増えたときにSDPARAがどれだけ高速にSDPを求解できるかを示した後，どれだけの規模のSDPまで現段階で解けているかを示す．

SDPARAの数値実験に用いた計算環境は以下である．

- 合計16ノードでMyrinet-10 G（10 Gbps）で接続
- 各ノードはXeon X5460（4 Core）を2個と48 GBメモリ，Cent OS 5.4で構成されている．

今回，SDPARAの計算実験で用いたSDPは，表5にある2つである．この表には，問題の出展も掲載している．表5の問題をPCの台数やスレッド数を変更しながら解いた時間をまとめたのが，表6である．

NH_3の問題は，合計時間で見ると1台/1スレッドの4,086秒から16台/8スレッドの80秒へ高速化されている．特に，ELEMENTSの部分は，3,941秒から41秒に短縮されており，合計128スレッドで96倍の高速化を達成している．

また，BroydenBand500の問題では，ELEMENTSの部分の高速化は，12,978秒から173秒と74倍の短縮に留まっている．しかしながら，SDPARAでは2.1節で述べた計算コストに基づく計算分散を行っており，疎行列に対する並列化として考えた場合には十分な並列効果を得ているともいうことができる．密行列のSchur補完行列のみを扱うPCSDPでは，計算時間以前にSchur補完行列を分散メモリに保持することさえできないため，SDPARAの大きなアドバンテージのひとつともいえる．

最後に現状のSDPARAでどれだけの規模の問題を解くことができるかを示したのが，表7である．ここでは，量子化学とセンサーネットワーク問題のSDPについて，合計16台をフルに用いてのSDPARAでの最大性能を示している．センサーネットワーク問題は，2次元平面にセンサーがばら撒かれている状態でセンサー間の距離だけが与えられたときに各センサーの座標を求める問題である．各センサーの座標からセンサー間の距離を求めることはピタゴラスの定理から容易であるが，この逆は極めて難しく，ここでは高精度な近似解を生成することにSDPを利用している．Sensor40000はセンサーネットワーク問題として40,000頂点を保持しており，72 GBのメモリを使用して文献[4]で生成できる最大規模の問題でもある．H_2O，Sensor40000はともに超大規模SDPであり，これだけの規模のSDPを解くことができるのは，SDPARAのみである．

表5　大規模SDPの規模

問題名		m	n	nnz	sd	出典
量子化学	NH_3	2964	3178(744)	566720	100%	[5]
多項式計画問題	BroydenBand500	392171	59280(120)	7113106	0.67%	[6]

表6　SDPARAによる計算時間（単位：秒）

問題名 / スレッド数		PC台数	1	2	4	8	16
NH_3	/1	ELEMENTS	3941.65	1974.94	993.71	500.80	252.37
		CHOLESKY	47.26	27.88	16.45	12.98	8.31
		合計計算時間	4086.48	2098.32	1101.09	602.43	347.22
NH_3	/8	ELEMENTS	568.23	288.10	147.16	76.61	41.04
		CHOLESKY	21.48	11.42	9.20	8.83	6.79
		合計計算時間	632.51	341.28	193.03	119.57	80.07
BroydenBand500	/1	ELEMENTS	12978.29	6763.48	3441.37	1850.83	995.64
		CHOLESKY	21953.57	12807.86	7479.62	4648.30	3013.93
		合計計算時間	35988.86	20394.17	11571.03	7100.83	4666.87
BroydenBand500	/8	ELEMENTS	2941.40	1302.34	585.40	321.99	173.43
		CHOLESKY	13134.59	7983.12	4997.47	3388.00	2456.02
		合計計算時間	17054.42	10055.84	6190.81	4275.33	3263.54

表7　超大規模SDPの規模

問題名		m	n	nnz	sd	出典	計算時間（秒）
量子化学	H_2O	27888	15194(4032)	9438380	100%	[5]	51046.45
センサー	Sensor40000	602484	918468(50)	3903224	5.92×10^{-3}%	[4]	209.67

3. まとめ

表7で示したのが，どれだけの規模のSDPをどれだけの時間で解くことができるか，に対する現状のSDPソルバーの目安であるともいえる．最初の表1で示したとおり，SDPソルバーの性能はこの15年間で飛躍的に上昇しているが，藤澤克樹氏の記事にも一部あるように，アルゴリズム中のパラメータの設定，並列計算やCPUキャッシュの効率化によって，一層の性能向上の余地は十分にある．これらの改良によって，さらに大規模なSDPをより短時間で求解することを今後も目指していきたい．

参考文献

[1] Borchers, B.: "SDPLIB 1.2, a library of semidefinite programming test problems," *Optimization Methods and Software*, Vol. 11 & 12, pp. 683-690 (1999).

[2] Fujisawa, K., Kojima, M. and Nakata, K.: "Exploiting Sparsity in Primal-Dual Interior-Point Methods for Semidefinite Programming," *Mathematical Programming*, Vol. 79, pp. 235-253 (1997).

[3] Ivanov, I. D. and de Klerk, E.: "Parallel Implementation of a Semidefinite Programming Solver based on CSDP in a distributed memory cluster," CentER Discussion Paper, Vol. 2007-20, pp. 1-22 (2007).

[4] Kim, S., Kojima, M., Waki, H. and Yamashita, M.: "SFSDP: a Sparse version of Full SemiDefinite Programming relaxation for sensor network localization problems," Research Report B-457, Dept. of Math. and Comp. Sciences, Tokyo Institute of Technology, 152-8552, July (2009).

[5] Nakata, M., Braams, B. J., Fujisawa, K., Fukuda, M., Percus, J. K., Yamashita, M. and Zhao, Z.: "Variational calculation of second-order reduced density matrices by strong N-representability conditions and an accurate semidefinite programming solver," *Journal on Chemical Physics*, Vol. 128, 164113 (2008).

[6] Waki, H., Kim, S., Kojima, M., Muramatsu, M. and Sugimoto, H.: "SparsePOP: a Sparse Semidefinite Programming Relaxation of Polynomial Optimization Problems," *ACM Transactions on Mathematical Software*, Vol. 35 (2), 883 (2008).

[7] Yamashita, M., Fujisawa, K. and Kojima, M.: "SDPARA: SemiDefinite Programming Algorithm paRAllel version," *Parallel Computing*, Vol. 29, pp. 1053-1067 (2003).