最適化問題に対する並列計算技術の適用

藤澤　克樹

数年前からクラスタやグリッドなどの並列計算技術が広く普及し，多くの分野に適用されて成功を収めている．最近ではマルチコアを搭載したプロセッサの登場によって，さらに簡単，安価に並列計算の適用が行えるようになった．本稿では最適化問題をめぐる並列計算技術の現状に触れた後，最適化問題として半正定値計画問題を取り上げ，並列計算の適用に関する実験結果と考察等を報告する

キーワード：最適化問題，半正定値計画問題，クラスタ計算，マルチコアプロセッサ

1. 最近の動向について

1.1 並列化の三つのレベル

最近では最適化分野も含めた多くの理工系の分野において並列計算技術の適用が行われている．これには様々な要因があるのだが，数年前にはなかった事情も含まれている．PC用のCPUのクロック周波数は最近5年間では3 GHzの周辺に留まっている．消費電力，発熱量また回路設計の難度などの問題から，現在よりも大きくクロック周波数を上げることは難しくなっている．そのため1クロックあたりの平均命令実行数（IPC）を増やしたり，CPU内のコア数を増やす（マルチコア）などの改良が行われているが，これらも並列化の一種である．もはや以前のようにクロック周波数の増大による高速化が望めない以上，並列化に向かわざるを得ない状況がある．しかし，並列化の新技術によって以前よりもさらに安価，簡単に並列計算を行えるようになったのも事実である．後述するように2,3年以内に1 CPUの性能が1 TFlops（1秒間に1兆回の浮動小数点演算）に達するようになり，並列計算もまた新たな時代に入ったといえるだろう．

並列計算技術の適用といっても様々なレベルで様々な手法が混在しているのが現状である．そこで図1のように，並列化を三つのレベルで分けて考えることにしよう．目的，規模，性能，予算等を考慮した適切な並列化の組合せを選択することが重要である．以下この節では項目別に最近の動向について見ていこう．なお本稿の2節はレベル2,3節はレベル1と2,4節はレベル1と2と3に関する話題である．

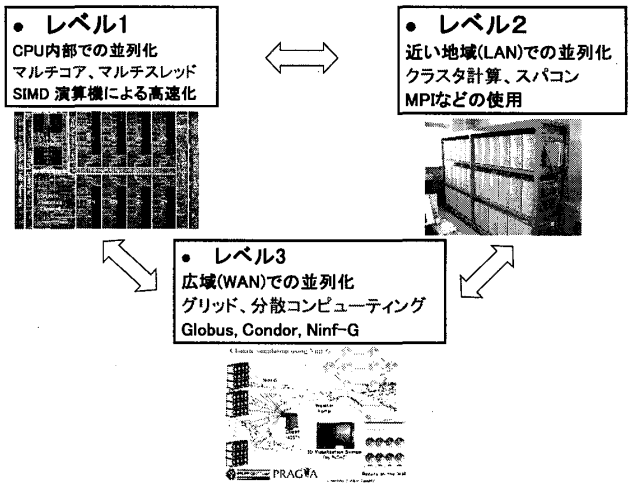

図1　並列化の三つのレベル

1. レベル1：CPU内部での並列化
   マルチコアの上でマルチスレッドのアプリケーションを実行する．IntelのSSE命令などのSIMD演算機による高速化も含まれる
2. レベル2：近い地域内（LAN）での並列化
   いわゆるクラスタ計算や地球シミュレータなどのスーパーコンピュータなどが含まれる
3. レベル3：広域（WAN）での並列化
   グリッドや分散コンピューティングなどの技術を用いる

1.2 PCの高性能化，低価格化

近年のPCの高速化，大容量化は大変目覚ましく，浮動小数点計算の性能で数GFlops（1秒間に数十億回の浮動小数点演算），主メモリも数Gbyteに達し，単体でも1990年代初等のスーパーコンピュータ並の性能をもっている（しかも数十万円以内で入手可能）．そのため単一のPCでも，優れたソフトウェアによっ

ふじさわ　かつき
中央大学 理工学部経営システム工学科
〒112-8551 文京区春日1-13-27

て非常に大きな規模の最適化問題が高速に解けるようになっている．例えば，整数計画問題などに対してはCPLEXやXpress-MPなどの商用ソルバーの性能が極めて高いことが報告されている．ただ注意すべきことはこの高速化はアルゴリズムとデータ構造の進歩も大きな役割を担っており，計算機が高速化されたことが唯一の要因ではない．CPLEXにおいては初期版と最新版を同一の計算機で計算すると，はるかに最新版の方が高速であることが報告されている．

1.3 最適化分野における大規模並列計算

計算量の理論から考察すると$\mathcal{NP}$-困難な問題などは問題の大きさが増加するにつれて，必要な計算量が指数関数的に増大するので，ある程度大きな規模では最新の情報技術を用いても最適解を求めることは依然として極めて難しい．しかし近年では，主に以下の二つの方法やそれらの方法の組合せによって最適化問題を解く研究がさかんに行われており，アルゴリズムの発展や並列計算の適用などによって目覚ましい成果があげられている．

1. 超高速，大容量の計算機を集めて従来より提案されている手法に並列計算の技術を適用して問題を解く．ただし様々な理論的成果を採り入れて，計算時間を減らす工夫を採用している．
2. 近似解法を適用して実用的な時間で優れた近似解を得ることを目指す．

1は巡回セールスマン問題（TSP）に対する分枝カット法[2]や二次割当問題に対する分枝限定法[1]などが有名である．前者は100台近いPCを集めてSW 24978という24,978点をもつ対称TSPの最適解を求めることに成功している．後者は平均で653個，最大で1,007個のCPUを用いて約1週間で30次元のNUG 30という問題の最適解を求めた．また次節で解説するように著者らのグループが数万制約，数百万非零要素をもつ半正定値計画問題（SDP）を解くことに成功した[3]．これらの並列化はレベル2あるいはレベル2と3の融合型である．2に対しては配送計画問題などの組合せ最適化問題に対して近傍探索などを並列化したメタ解法を適用して問題を解く例などが最近では多く見られる．

また，数理計画問題を解くためのソフトウェアもアルゴリズムの改善や計算機能力の向上によって大幅に性能を向上させている．数理計画問題は問題の大きさnの多項式時間で解けるものも多いが，想定される数理計画問題が巨大であるときにはCPU単体のみの性能向上だけでは短時間に問題を解くことは困難である．それらの困難を克服するための並列計算の手法として高性能なスーパーコンピュータなどを用いて並列計算を行う手法は古くから行われていたが，最近ではクラスタ計算（Cluster Computing）とグリッド計算（Grid Computing）などの並列計算技術が注目を集めている．クラスタ計算は複数の独立した計算機をGigabit Ethernet（最近では10 Gb Ethernetも）やMyrinetやInfinibandなどの高速なネットワーク装置で結合し，単一システムのイメージを提供する並列計算技術である．特に一般のPC技術を活用するPCクラスタでは，近年のPCの高性能化と低価格化によって，従来のスーパーコンピュータをはるかに上回るコストパフォーマンスを達成している．また，並列計算を実現するためのソフトウェアではMPI[1]やOpenMP[2]などが有名である．

1. MPI：分散メモリ並列コンピュータ上でのメッセージパッシングを行うためのプログラミングモデルである．逐次プログラムを並列化する際には，データを分散配置させて，各コンピュータ上で行われる計算と通信コストのバランスをどのように取るかによって大きく性能が左右される．プログラミングにはMPI通信ライブラリを用いる．大規模な並列計算に適しているなどの特徴がある．
2. OpenMP：共有メモリマルチプロセッサ（あるいはマルチコア・プロセッサ）上のマルチスレッドプログラミングのためのツールである．OpenMPでの並列プログラミングは逐次のプログラムに指示文（コンパイラに対する）を加えて並列化を行う．逐次的，部分的に並列化を行うことができMPIよりも簡単に並列化作業を行うことができる．

1.4 高速化によるスケールアップ

高速な計算機の上位500台を集めたウェブサイトTOP 500[3]によると，2007年6月現在で世界最高速コンピュータの浮動小数点演算の性能は280 TFlopsにも達している．さらに，2007年3月には理化学研究所などによる共同開発プロジェクトで通称“京速計算機”が神戸に設置されることが決まっている．この計

[1] http://www-unix.mcs.anl.gov/mpi/
[2] http://www.openmp.org/
[3] http://www.top500.org/

算機は名前の通りに10 PFlops（1秒間に1京回）の浮動小数点演算を行うことを目標にしていて，2010年頃の運用開始時には世界最高速になる見込みである．つまり，世界最高速レベルの計算機は近い将来テラフロップスからペタフロップスの単位になる．

また，最近ではマルチコア・プロセッサが主流になりつつあり，Intel Core 2系やPlayStation 3などに搭載されているCell（IBM/TOSHIBA/SCEI）などがある．Cellは2008年には34コア（PPE-2+SPE 32）を搭載して単精度演算で1 TFlops（倍精度演算で512 GFlops）に達する見込みである．他にもIntelのLarrabee, AMDのFUSION, NVIDIAのTeslaなどのCPUコア，GPUコアあるいは二つの統合などの形で新プロセッサが計画されている．初めは単精度演算が中心かもしれないが，こちらも近い将来1 TFlopsに達する見込みである．つまり，1台のPCでも近い将来ギガフロップからテラフロップスの単位に移行することになろう．

2. 超大規模SDPに対する並列計算

半正定値計画問題（SDP）は幅広い分野に応用をもち，線形計画問題を対称行列の空間へ拡張した構造をもっていることもあって21世紀の線形計画問題としての期待も大きい．SDPに対しては，すでに10年以上前から内点法アルゴリズムの適用が行われており，著者らのグループも1995年から高速かつ安定にSDPを解くためのアルゴリズムとソフトウェアの開発を行うSDPAプロジェクトを行っている[4]．

最近SDPの新たな応用として量子化学において分子の電子構造を求める研究がされている．分子の電子状態は固有値問題であるSchrödinger方程式を解くことによって求めることができる．特に分子の最も安定した基底状態に相当するのがSchrödinger方程式の最小固有値解であり，その最小固有値は基底状態エネルギーと呼ばれる．分子の基底状態エネルギーは化学反応モデル等の理論的なエネルギー予測に役立つ基本的な値である．波動関数を用いるアプローチの代用として1960年代にColemanやGarrod-Percusらによって提案されたのが縮約密度行列法と呼ばれる手法である．縮約密度行列法では分子の基底状態が2次の縮約密度行列で表され，その行列の線形制約式の元で最適化問題を解くことによって基底状態エネルギーの下界値が求まる仕組になっており，この最適化問題をSDPとして定式化することができる．このSDPの最適変数に相当するのが2次の縮約密度行列であり，目的関数では分子を構成する原子の種類や幾何構造等の情報をもった線形関数を最小化する．一方，縮約密度行列法からSDPとして定式化される問題は超大規模になりスーパーコンピュータまたは大型クラスタ計算機級のメモリ量が必要とされSDPに対するアルゴリズムも並列に処理されることが望まれる．今後SDPの効率的な解法の進歩もしくは高精度の基底状態エネルギーを要求する応用例の出現により，注目されるべき分野であろう．詳しくは文献[3]を参照のこと．

今回，数値実験として表1のような系に対してSDP緩和を生成し，SDPARA 1.0.1[4]で解いた．SDPARAとはSDPに対する主双対内点法を実装したSDPA 6.2.1[5]のボトルネックとなっている探索方向の計算のための線形方程式系の計算をMPIやScaLAPACKなどを用いて並列化したソフトウェアである．この場合では$2K$は基底関数の数，Nは電子数，mはSDPの等式制約の本数，nはSDPの各行列の大きさ，#nonzeroはSDPに含まれる非零要素の総数を示している．また，nBLOCKは各行列のブロック対角の数，bLOCKsTRUCTはブロック対角の構成を示すベクトルである．ただし，表記スペースの問題からbLOCKsTRUCTは大きい順に並べて一部省略してある．例えば4,032×2とは4,032×4,032のブロックが二つあるという意味である．−430とは対角ブロックの大きさが430という意味である．表1で最も巨大なSDPはH_2Oに関するもので，図2のような巨大ではあるが疎なブロック対角構

表1 各問題の大きさに関するデータ

化学式	$2K$	N	m	n	#nonzero
LiOH	22	12	10,593	1,264	75,690
HF	24	10	15,018	1,498	105,694
NH	24	8	15,018	10,170	2,205,558
BH_3O	26	16	20,709	12,828	3,290,622
H_2O	28	10	27,888	15,914	4,766,902

化学式	nBLOCK	bLOCKsTRUCT
LiOH	14	(242,121×4,...,-274)
HF	14	(288,144×4,...,-322)
NH	22	(2532×2,792×4,...,-322)
BH_3O	22	(3224×2,1014×4,...,-374)
H_2O	22	(4032×2,1274×4,...,-430)

[4] http://sdpa.indsys.chuo-u.ac.jp/sdpa/

造を有している．15,914×15,914の大きさをもつ行列が問題内に27,888個存在していて，これらの行列の非零要素の合計数は4,766,902個に達する．これまでに解くことができた一般のSDPでは世界最大規模であると思われる．

表1の中のLiOH，HF，NH，BH_3Oに関しては産業技術総合研究所のAIST Super Cluster（ASC）のP32クラスタを，H_2Oに関してはM64クラスタを用いた．ASCは全体で14.6 TFlopsの総演算性能と9.78 TFlopsの実効性能をもっており，全部で三つのクラスタ（P32，M64，F32）と3,208個のプロセッサより構成されている．P32クラスタはメモリ量はそれほど多くないが同時に多くのCPUを必要とする場合に用いられる．また，M64クラスタは1CPUあたりのメモリ使用量が極めて大きいときに用いる．表2はCPU数を変化させたときのSDPARAの実行時間を表している（単位は秒）．P32クラスタの各PCはCPUとしてOpteron 246（2.0 GHz）を2個搭載し，メモリは6 GBである．また，M64クラスタの各PCはCPUとしてItanium 2（1.3 GHz）を4個搭載し，メモリは16 GBである．なおLiOH，HFは64台で十分高速に計算できたため128台以上では計算を行っていない．また，BH_3Oでは1CPUあたりのメモリ使用量が6.4 GBなので合計メモリ使用量は6.4×64台＝409.6 GBにも達する．また，H_2Oは1CPUあたりのメモリ使用量が11.2 GBなので合計メモリ使用量は11.2×8台＝89.6 GBにも達し，実行時間は約24日である．すべての数値実験において得られた解の精度も良く，この規模で精度の良い解が求まるのは珍しい現象である．SDPARAはこのような超大規模なSDPの計算においては他のソフトウェアの追随を許していない．

表2　大規模SDPに対するSDPARAの実験結果

問題名	8台	64台	128台	256台
LiOH		414		
HF		1.120		
NH		66.015	37.028	24.499
BH_3O		148.387		
H_2O	2.060.237			

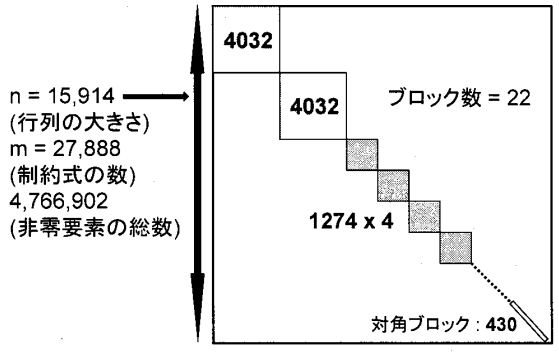

図2　超巨大SDPのブロック対角構造

3. マルチコア，マルチスレッドを用いたSDPに対する高速，安定計算

すでに述べたように単一のCPUでも高性能なマルチコアをもっているので2節のような大規模な設備が存在しなくても大規模な問題以外では十分な性能を得ることができる．SDPA 6.xでは行列積やベクトルの内積などの線形代数演算にATLAS[5]を用いているが，最新のSDPA 7.xにおいてはGotoBLAS[6]も選択可能になっている．GotoBLASはテキサス州立大学の後藤氏によって作成され，CPUの浮動小数点演算性能，キャッシュやメインメモリのバンド幅やレイテンシなどを考慮しながら高度に最適化されたライブラリである．表3はSDPA 6.2.1，SDPA 7.0.2とSDPT 3[7]を実行した結果である．GotoBLASはマルチスレッドに対応しているので1，2，4スレッドの場合で実行した．表中の187.0(31)とは実行時間が187.0秒で，反復回数が31回という意味である．並列計算では実時間で評価する必要があるので＊が付いているのは実

表3　数値実験結果（SDPA 6 & 7，SDPT3 4.0β）：CPU Opteron 850（2.4 GHz）×4：OS Linux 64 bit

	構造最適化	多項式最適化
SDPA 6.2.1(ATLAS)	187.0(31)	4998.0(20)
SDPA 7.0.2(ATLAS)	72.3(30)	3.2(20)
SDPA 7.0.2(Goto:1)	58.0(30)	2.9(20)
SDPA 7.0.2(Goto:2)	43.5*(30)	9.7*(20)
SDPA 7.0.2(Goto:4)	36.6*(30)	14.9*(20)
SDPT3 4.0β	79.7(40)	30.4(18)

	組合せ最適化	ノルム最小化
SDPA 6.2.1(ATLAS)	143.7(20)	5.3(16)
SDPA 7.0.2(ATLAS)	116.8(18)	5.3(16)
SDPA 7.0.2(Goto:1)	102.2(18)	4.2(16)
SDPA 7.0.2(Goto:2)	67.2*(20)	5.3*(16)
SDPA 7.0.2(Goto:4)	48.8*(19)	6.3*(16)
SDPT3 4.0β	234.0(23)	6.0(13)

[5] http://math-atlas.sourceforge.net/

[6] http://www.tacc.utexas.edu/resources/software/software.php

[7] http://www.math.nus.edu.sg/~mattohkc/sdpt3.html

表4　数値実験結果（SDPARA 1.0.1, SDPA 7.0.2）：CPU Xeon 5345（2.33 GHz）×8, OS Linux 64 bit

	組合せ最適化	量子化学
SDPA 7.0.2(Goto:4)	27.1(19)	3770.0(25)
SDPARA 1.0.1(4CPU)	148.8(23)	1089.0(25)

表5　数値実験結果（SDPA-GMP 1.0, SDPA 6.2.1）：CPU Opteron 270（2 GHz）×4：OS Linux 64 bit

	SDPA-GMP 1.0	SDPA 6.2.1
双対ギャップ	9.21575e-21	8.57085e-05
目的関数値（主）	**-7.3430762652**	**-7.343**0703498
目的関数値（双）	**-7.3430762652**	**-7.343**6997405
実行可能性（主）	3.59916-e76	2.34386e-12
実行可能性（双）	1.29434-e35	5.10099e-06
実行時間	773.2(96)	0.2(16)

表6　数値実験結果1（SDPA-GMP 1.0＋OpenMP）

	組合せ最適化	ノルム最小化
SDPA-GMP(1CPU)	773.2(96)	106.4(28)
SDPA-GMP(2CPU)	486.8*(96)	83.6*(28)
SDPA-GMP(4CPU)	412.8*(96)	58.4*(28)

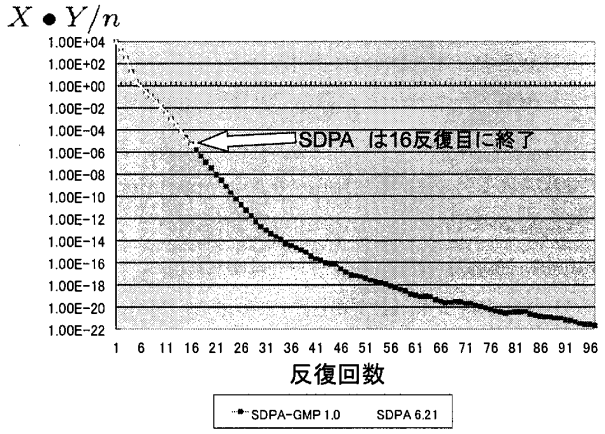

図3　SDPA-GMP と SDPA の収束状況の差

時間で，残りはCPU時間で計測している．SDPA 7.0.2＋GotoBLAS 1.1.5 の組合せは概ね良好で SDPA 7.0.2（SDPA 6.2.1）＋ATLAS 3.7.34 や SDPT 3 4.0β よりも高い性能を示している．表3の結果から実行時間がある程度大きな問題ではマルチスレッドによる並列計算は有効だが，反対に小さい問題では1スレッドの方が高速であることがわかる．

表4は SDPARA 1.0.1 と SDPA 7.0.2＋GotoBLAS 1.1.5 を比較したものである．SDPARA のように MPI を用いた処理（レベル2）では大きな行列の各要素を分散して計算するなどのアルゴリズムレベルでの大きな枠組での並列化が有効であり，GotoBLAS のようなマルチスレッド化（レベル1）は行列の積計算などの関数レベルでの並列化に適しているといえよう．今後はレベル1とレベル2の特性を活かしたハイブリッド型の並列計算も重要になるだろう．

また近年では，データ量の増大に伴って倍精度演算では十分な精度が得られないことも多い．そこで線形代数演算を行う変数を倍精度から任意精度の変数（GMP ライブラリを用いる[8]）に変更したソフトウェア SDPA-GMP の作成も行っている．表5は SDPA-GMP 1.0（200 bit）と SDPA 6.2.1（double：64 bit）の結果を比較したものであるが，双対ギャップや実行可能性などの値に多倍長計算の効果が顕著に表れている．また図3は縦軸に双対ギャップ，横軸に反復回数を取って両ソフトウェアの収束状況の差を示したものである．しかし当然ながら多倍長計算は膨大な実行時間を要するので，OpenMP などによるマルチスレッド化が有効である．OpenMP による簡単な作業（30分程度）で表6にあるように高速化を行うことができる．

4. おわりに

すでに2節で述べたように SDPA プロジェクトのソフトウェア群を公開するだけでなく SDPA Online Solver[9] というWebサービスを提供している．SDPA Online Solver は，レベル1,2,3の並列化技術が融合されたシステムになっている．SDPA 6.2.1 や 7.0.2，さらに SDPARA 1.0.1 による並列計算なども利用可能なので興味のある方は是非試していただきたい．

最後に並列化に関する他の話題について二つほど触れてみよう．行列同士の演算（BLAS Level 3）はキャッシュ上でのデータ再利用が行いやすいので理論計算値に近い値が得られるが，行列とベクトルの演算（BLAS Level 2）はデータ再利用が難しく GotoBLAS や ATLAS でもあまり性能が上がらない．そのため BLAS Level 2 の演算では今までの方法の延長ではマルチコアを活用した高速計算は難しいと考えられている．SDPA は表7のように多くの問題において行列積（BLAS Level 3）が全体の実行時間の中で大きな割合を占めている．また，1コアと4コアのと

[8] http://gmplib.org/
[9] http://sdpa.indsys.chuo-u.ac.jp/portal/

表7　SDPA 7.0.2+GotoBLAS 1.1.5における行列積演算の実行時間に占める比率と順位，および並列化の効果

	比率 (%)	順位	1コア → 4コア
組合せ最適化	74.2	1位	2.41倍
構造最適化	41.9	1位	1.50倍
多項式最適化	4.2	7位	0.14倍
ノルム最小化	24.0	2位	0.95倍

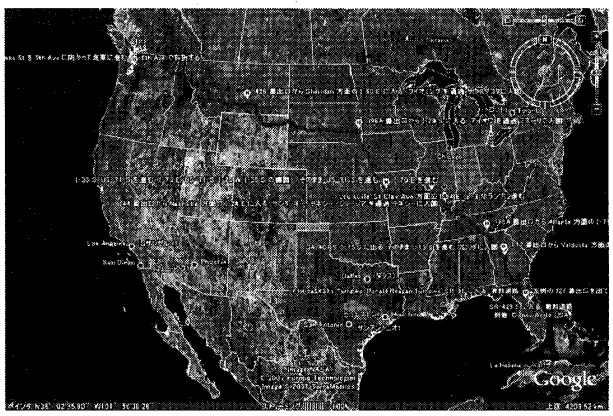

図4　Google Earth：シアトルからディズニーワールドまで

きの高速化の倍数を見ると，BLAS Level 3演算が多くを占めている場合には並列計算の恩恵を受けやすくなることがわかる．

最後に別の最適化問題（最短路問題）について触れる．図4はGoogle Earthを用いてアメリカ大陸横断の最短路を求めた結果であるが，query（問い合わせ）は数秒以内で終了する．最近では前処理に時間をかけてネットワークを簡略化して，二点間の最短距離を求める問い合わせに対してはms（ミリセカンド）程度の時間で反応するようになっている[10]．そのため前処理には並列化が有効であるが，問い合わせに対しては簡略化したネットワーク上での簡単な探索に留めておく必要がある．

謝辞　SDPAの開発に対してテキサス州立大学の後藤和茂先生から貴重な情報や助言をいただきました．また，独立行政法人産業技術総合研究所からは計算機資源としてAIST Super Clusterの提供を受けました．SDPAについてはSDPA Projectのメンバー[11]の共同研究による成果であることを明記しておきます．

参考文献

[1] K. M. Anstreicher, "Recent Advances in the Solution of Quadratic Assignment Problems," Mathematical Programming, Series B, **97** (2003) 27-42.

[2] D. Applegate, R. Bixby, V. Chvátal and W. Cook, "Implementing the Dantzig-Fulkerson-Johnson Algorithm for Large Traveling Salesman Problems," Mathematical Programming, Series B, **97** (2003) 91-153.

[3] K. Fujisawa, K. Nakata, M. Yamashita and M. Fukuda, "SDPA Project: Solving Large-scale Semidefinite Programs," Journal of the Operations Research Society of Japan, **50** (4) (2007) に掲載予定.

[4] M. Yamashita, K. Fujisawa and M. Kojima, "SDPARA : SemiDefinite Programming Algorithm PARAllel Version," Journal of Parallel Computing, **29** (2003) 1053-1067.

[5] M. Yamashita, K. Fujisawa and M. Kojima, "Implementation and Evaluation of SDPA 6.0 (SemiDefinite Programming Algorithm 6.0)," Optimization Methods and Software, **18** (4) (2003) 491-505.

[10] http://algo2.iti.uni-karlsruhe.de/schultes/hwy/

[11] http://sdpa.indsys.chuo-u.ac.jp/sdpa/contact.html