最適化ソルバー開発への最新の情報技術の適用について

藤澤　克樹

最適化ソフトウェアに関連性の高い情報技術には，マルチコア・プロセッサ，GPUコンピューティング，スーパーコンピュータ，クラウド・コンピューティングなどがある．これらの新技術が個別あるいは複合して最適化ソフトウェアとどのように絡んでくるのか，あるいはどのように活用すれば性能向上などの成果を上げることができるのかについては，最新の研究成果を含めてあまり知られていない．そこで本解説では，著者らのグループによる半正定値計画問題（SDP）に対するソフトウェア開発を題材にして，最先端の最適化アルゴリズムと最新の情報技術の有機的な融合方法等について触れていく．

キーワード：半正定値計画問題，最適化ソフトウェア，情報技術

1. 情報技術の進化と最適化ソフトウェア

計算機や通信技術はその登場以来，常に進化を遂げていることは良く知られている．例えば最適化ソフトウェアに関係が深いキーワードには，本解説で扱う**マルチコア・プロセッサ，GPUコンピューティング，スーパーコンピュータ，クラウド・コンピューティング**などがあり，インターネットやマスコミなどでこれらのキーワードに触れる機会は大変多くなっている．各キーワードの意味や内容についてはインターネット等で簡単に検索して調べることができるのだが，これらの新技術，新手法が個別あるいは複合して最適化ソフトウェアとどのように絡んでくるのか，あるいはどのように活用すれば性能向上などの成果を上げることができるのかについては，日本語だけでなく英語で検索しても情報の入手が困難である．そこで本解説では，著者らのグループによる半正定値計画問題（SDP）に対するソフトウェア開発を題材にして，最先端の最適化アルゴリズムと最新の情報技術の有機的な融合方法等について触れていく．SDPに対するソフトウェア（SDPA[1]やSDPARA[2]など）[1]の開発を中心とした構成になっているが，他の最適化問題に対しても広く普遍的に適用できる内容になっている．

近年，アルゴリズムサイエンス分野における基礎理論の探求は飛躍的に進んでおり，以前のような理論的計算量を重視する立場から，ソフトウェア実装面を意識して，実際にどのような構造，特性を持つ場合において高速かつ安定に解けるかといった研究に移行してきている．一方，コンピュータのハードウェア＆ソフトウェア面での研究の進展も著しく，近年ではスーパーコンピュータ等を中心とした並列計算技術だけでなく，汎用的なマルチコア・プロセッサ上の並列計算から大規模環境下でのクラスタやクラウドなどの新世代の実装方式が開発されている．多くの最適化問題において高速＆高性能化が達成されていることに疑いはないのだが，例えば

1. 具体的にコンピュータ内部（ハードウェア&ソフトウェア）でどのような現象が起きているのか？
2. どこまで性能を上げることができるのか？（理論的限界等）
3. 実用的な観点からハードウェアの仕様や実行時間に制限が付いたら対応は可能か？

といった質問に対して，現在の複雑かつ高度な数理科学と情報科学の発展の中で，説得力を持った答えを見つけることは困難である．そこで，数理科学分野と情報科学分野の最新の成果を持ち寄り，アルゴリズムサイエンスの立場における最適化ソフトウェアの実装方式について解明を行っていくことが必要とされている．具体的には以下の方針や技法，結果などを重視する方

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

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

針を取っている．

1．理論的限界等から性能目標を定める（ボトルネックとなる箇所の特定）．
2．トレードオフ関係の把握（数値演算能力とメモリバンド幅等）．
3．計算量とデータ移動量の正確な推定を行う．
4．データの特性（疎性，サイズ）と性能値の関係を見極める．

また最近の欧米などの海外では，以下のような研究の傾向が見られる．

1．理論的な研究であっても数値実験による評価，検証がほぼ必須となっている．
2．実用的な最適化問題を解くために理論から実装，パラメータ設定，大規模並列計算等の各分野の研究者が集結して総力戦を行っている．

一方，日本では最適化アルゴリズムの実装技術とソフトウェア開発能力が低く，“優れた理論が提案されても実装してソフトウェアとして公開することができない”また“実用的な問題を解く際に海外の最適化ソフトウェアに極度に依存している”という状態になっている．日本の最適化研究の未来を考える際には現状で良いのか広く議論していく必要があろう．

2．メモリ階層構造とボトルネック箇所の特定

最近ではPC用のCPUでも理論的な最大計算性能は数十GFlops（1秒間に数百億回の浮動小数点演算を行う能力）に達しており，1990年代前半のスーパーコンピュータを凌駕する性能を持っている．しかし，この最大計算性能は特定の条件下でのみ達成されるものであり，最適化アルゴリズムでは数値演算だけを行うのではなく，比較演算やデータ移動なども頻繁に行われるので，最適化アルゴリズムの実行性能の評価に関してはまた別の方法が必要となってくる．この節では計算機内のメモリ階層構造を考慮し，最適化アルゴリズムの実行時にボトルネックとなる箇所の特定と性能を評価する簡単な方法を紹介する．この方法の活用によって最適化アルゴリズムの実装に際して，GPUを使うべきか，あるいはスーパーコンピュータで大規模並列化すべきかといった方針が決定しやすくなるであろう．

最近では計算機内部が図1のようなメモリ階層構造を持っている．このメモリ階層構造では，上位レベルになるほどアクセス速度が高速で小容量な記憶領域を，

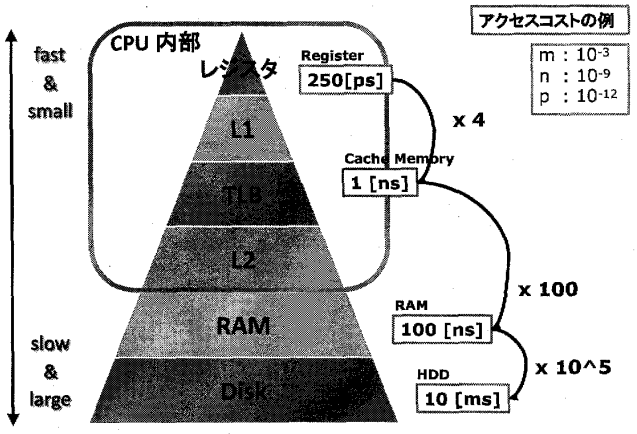


図1　メモリ階層構造の例

下位レベルになるほどアクセス速度が低速で大容量な記憶領域を保持している．特にCPU内部では，レジスタやキャッシュメモリ，TLB[2]など非常に高速な記憶領域が存在する．演算処理はレジスタ上でのみ行うことができるが，アクセス速度が非常に高速（250 ps）かつ容量が非常に小さい．また主記憶装置（RAM）は数Gbytes以上と非常に大容量であるがアクセス速度はレジスタと比較すると極めて低速である（100 ns）．そのため最適化分野に限らずソフトウェアの高速化のためには，演算量とデータ移動量の割合を考慮してデータを適切に配置し，レジスタと主記憶装置の中間に位置するキャッシュメモリを有効に利用することは非常に重要である．L2（あるいはL3）キャッシュメモリは，比較的高速で数Mbytesという大きな容量を保有しているので，かなり大きな1次元配列でも格納することができる．

例えばメモリ階層構造をロジスティックネットワークで考えるとCPUは工場，レジスタは工作機械，L2キャッシュはライン内の在庫置き場，RAMは工場内の倉庫，Diskは遠くにある大型の倉庫になる．しかも演算処理はレジスタでしか行うことができないので，頻繁にRAMやDiskにデータを取りにいくのでは，その間はレジスタでの計算が停止した状態が続いてしまうので大変効率が悪い．まとめるとCPUの性能を引き出すことができるアルゴリズムというのは，**データ移動量に対し演算量がある程度大きいこと（つまり$\frac{\text{演算量}}{\text{データ移動量}}$の比がある程度大きいこと）**という性質を持っていることがわかる．ただし，データの疎性を利

[2] 仮想メモリアドレスと物理メモリアドレスの変換のためのバッファメモリ．

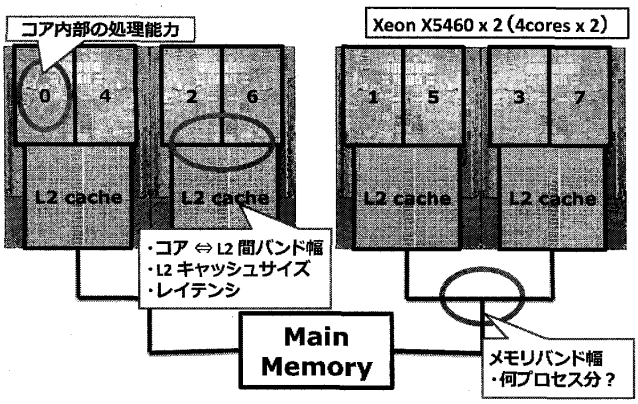


図2　ボトルネックとなる箇所の候補について

用した場合には$\frac{\text{演算量}}{\text{データ移動量}}$の比が小さくなることがあるが，全体の計算時間が減るのであれば問題はない．

ある最適化アルゴリズムを実装した際にボトルネック（律速）となる箇所は次の場合に分類することができる（図2参照）．

1. CPU内部の処理が律速の原因となる．
   (a) CPU内の浮動小数点能力に律速あるいはレジスタ⟺L2（あるいはL3）キャッシュ間のバンド幅に律速している．
   (b) 演算処理以外の部分に律速している（命令デコードやアドレス計算など）．
2. CPU⟺メインメモリ間のバンド幅に律速している．

例えば最適化された行列積の計算は1-(a)，ベクトル同士の内積計算では2の場合に相当する．またデータ量や演算量が少ない疎データの計算時には1-(b)に相当することもある．SDPAではこれらの処理をすべて含んでいるので，実行時には動的にボトルネックとなる箇所が変化していることになる．また最近のマルチコア・プロセッサを用いて並列計算を行う際には効率良く実行するためにメインメモリやキャッシュメモリなどの資源の競合を抑えるように設計する必要があるが，ある最適化アルゴリズムの実装が上記の2に相当する場合では，アルゴリズムを改善するか，メモリのバンド幅を上げない限り，並列計算を行っても高速化できないことがわかる．

組合せ最適化問題に対するアルゴリズム（例えばメタ戦略アルゴリズム等）では，数値演算の量が比較的小さく，多くの処理がデータ移動や比較演算などに費やされている．この場合ではメインメモリに対するデータアクセスの頻度が非常に大きく2の場合に相当するように思えるのだが，近傍探索などでメモリの局所参照性が保持されていることが多いので1-(a)，つまりレジスタ⟺L2（L3）キャッシュ間のバンド幅に律速している場合も多いようだ．

3. GPUコンピューティングと最適化ソフトウェア

GPUは汎用的なCPUとは異なり主にグラフィック関係などの特定の処理用に作られたプロセッサである（汎用性を強調してGPGPUと呼ばれることも多い）．ここ数年の間にGPUの大手メーカであるNVIDIAを中心にGPUをグラフィック以外の処理にも使用する試みが積極的に行われている．GPUの使用によって処理速度が数十倍，数百倍高速になったという報告が頻繁に行われているので，最適化分野でもGPUコンピューティングを行うことによって大きな成果を上げることができそうだが，実際には幾つかの問題点があるので最適化分野への適用は現在ではあまり進んでいない．反対にどのような性質を持つアルゴリズムであればGPUによって高速化できるのか考えてみよう．

1. アルゴリズムが単純で（比較や条件分岐などがない），数値演算中心で構成されていること．
2. 処理データが並列計算数に合わせてほぼ均等に分割できて独立に処理できること．
3. $\frac{\text{演算量}}{\text{データ移動量}}$の比が十分程度大きいこと．
4. 単精度演算で十分な数値精度が得られること．

これらは最適化アルゴリズムには厳しい条件であり，多重ループで構成される動的計画法のアルゴリズムを単純に分割して計算するなどの少数の例を除くとあまり成功した例は見られない（GPU内のメモリから溢れるような大規模な問題では高速化は限定的になる）．現状では，GPU内のビデオメモリのレイテンシとCPUとGPUをつなぐPCI-Expressバスのバンド幅が大きなボトルネックとなっている．ただし，今後登場するGPUではECCによる信頼性の確保，プログラミングの汎用性の向上，キャッシュメモリの大容量化，倍精度演算の高速化などが行われる予定であり，一部の処理だけを適切にGPUで行うようにすることよって最適化分野への適用事例も増えていくことだろう．SDPに対するソフトウェアの開発では倍精度演算の精度の低さが大きな問題となってきているので，例えば4倍精度演算をGPUを用いて高速化するといった応用も考えられる．

4. スーパーコンピュータ上での最適化問題の実行

スーパーコンピュータは2009年秋の事業仕分け作業によって注目を集めた．世界最高速（Linpackベンチマーク）を目指す意義や国産の維持などについて世間も含めて広く議論が行われた．しかし，アプリケーション側から見るとスーパーコンピュータを使いこなす（性能を引き出す）ためのソフトウェア技術の開発等にも尽力すべきであって，そもそもベンチマークではなく実アプリケーションで性能を引き出すことができないようなハードウェアであれば世界第一位でも第二位でも同じく意味が無いであろう．GPUコンピューティングや後述するようなクラウド・コンピューティングの普及によって，従来スーパーコンピュータで行われてきた数値実験等の一部が代用されるようになっている．それでは最適化分野ではどのような最適化問題がスーパーコンピュータでの実行に適しているのであろうか．以下に適した条件を列挙してみよう．

1．解くべき問題の規模が非常に大きいこと．
2．アルゴリズムの演算量がある程度大きいこと．
3．$\frac{\text{演算量}}{\text{データ移動量}}$ の比が十分程度大きいこと．
4．並列計算数に合わせて演算量やデータ移動量がほぼ均等になるように各プロセスに割り振ることができること．
5．並列に動作しているプロセス同士が高速かつ高い信頼性で頻繁に通信を行ったり，同期を取る必要がある場合．

1と2については言うまでもないが，3は並列計算に限らず性能効率を上げるための必要条件である．4について説明すると，スーパーコンピュータではGPUでの並列計算と異なり，演算量やデータ移動量がほぼ均等になるように分割することができれば，原則的には各プロセスは全く異なる処理を行っても良いということである．また5の条件が必要でない場合，例えば単なるパラメータサーチで最後に結果だけ集めれば良いのであれば，クラウド・コンピューティングの技術等で大量の計算機資源を集めても良いだろう．反対に並列分枝限定法などでは5の観点からはスーパーコンピュータ上の方が実装が容易で，安定して動作させることができる．最近では各ノードにGPUを搭載したスーパーコンピュータが登場しているので，組み合わせが複雑になる分だけ実装は困難となるが，適切な演算量とデータ移動量の分割によって，さらなる性能向上も期待できる．

以上を踏まえてスーパーコンピュータでの実行に適した最適化問題について考えてみよう．TSP（巡回セールスマン問題）に対する適用例[3]などは知られているが，例えば線形計画問題（LP）に対する内点法では，入力データに疎性があることが多く，またベクトル演算が中心ということもあって，データを分割して転送するコストの比率が大きくなり大規模な並列計算の適用は難しい．また最短路問題に対するダイクストラ法ではアルゴリズム内の作業間の依存性が高く，並列して実行できる部分が少ないのでこれもまた適用が難しい．著者らのグループではSDPに対する主双対内点法アルゴリズムを並列化したソフトウェアSDPARA[2]の開発，公開を行っている．超大規模なSDPをSDPARAを用いて解いた場合には，上記の5条件を満たすことがわかっている．現時点では超大規模なSDPを解くためにはスーパーコンピュータが必須となっている．SDP以外でもMIP（混合整数計画問題）に対して，スーパーコンピュータ上で大規模並列計算を行うソフトウェアParaSCIPの開発が独ZIB研究所の品野研究員らによって行われている．1節でも述べたように，日本の最適化ソフトウェア開発の能力は低く先進国の後塵を拝しているが，スーパーコンピュータ上での最適化問題の大規模並列計算に関しては世界的にも進んだ地位にあるといっても良いだろう．

SDPは組合せ最適化，システムと制御，データ科学，金融工学など幅広い応用を持っているが[3]，今回は超大規模SDPの需要を多く持つ量子化学分野に対する数値実験を紹介しよう（ソフトウェアはSDPARAを用いる）．化学または物理分野では原理的にはシュレーディンガー方程式を解くことによって，ほぼすべての現象を理解することができるといわれている．例えば，水分子の挙動，蛋白質の性質，光合成，超電導の仕組みなどがある．しかし，シュレーディンガー方程式を解くのは様々な面で困難を抱えるため2次の縮約密度行列の直接変分法という手法の開発，研究が行われている．著者らのグループではSDPARA（本特集の山下真氏による解説を参照）やSDPA-GMP（本特集の中田真秀氏による解説を参照）を用いて，世界で初めて正確に2次の縮約密度行列の直接変分を行い，多くの原子・分子に適用することに成功

[3] http://www.tsp.gatech.edu/

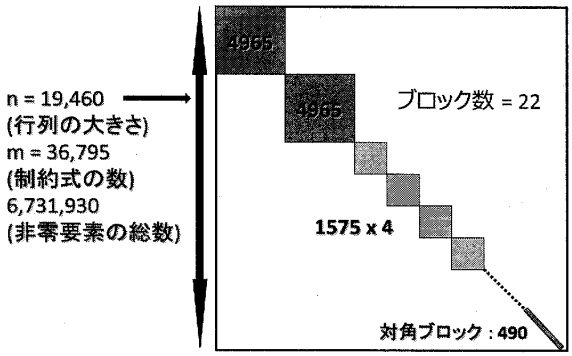


図3　超大規模SDPとブロック対角構造

表1　量子化学分野の超大規模SDPに対するSDPARAの実験結果

問題名 (分子名)	総計算時間 (秒)
H_2O	27,523.8
CH_3	68,593.4
NH_3	72,025.6
O_2	5,943.1

表2　京都大学T2Kスーパーコンピュータの構成

CPU	AMD Opteron 8356 2.3GHz x 4
メモリ	32 GByte
NIC	GbE x 2 & Infiniband x 4
今回利用した計算機資源数	128 ノード 512CPU(2048 コア)

した[1]．特に以下のCH_3，NH_3，O_2から生じる巨大なSDPに対しては，今回（2010年3月）世界で初めて京都大学T2Kスーパーコンピュータ[4]を用いてSDPARAによって最適解を求めることに成功し，世界最大規模のSDPを非常に正確に安定して解くことができた．今回の数値実験で解いた最も大きなSDPは図3のようなブロック対角構造を持っており（CH_3，NH_3から生じるSDP），各行列の大きさは19,460×19,460，制約式の数は36,795個，非零要素の総数は6,731,930個にも達する．

上記の京都大学T2Kスーパーコンピュータのように現在のスーパーコンピュータの主流はいわゆるスカラー型で各CPUは複数のCPUコアを有している．一昔前のスーパーコンピュータではベクトル型が主流であったために，過去にベクトル型用に作成されたソフトウェアは現在のスカラー型で性能を上げるためには大幅に書き換える必要もあるだろう．またMPIライブラリによって各プロセスに分割した後に複数のCPUコアを用いて，プロセス内をPthreadやOpenMPなどを用いてマルチスレッド化する2段階並列が現在のスーパーコンピュータのアーキテクチャでは有効である．しかしCPUの演算性能はコア数の増大とともに急激に上昇しているが，CPUとメモリ間のバンド幅などはあまり向上が見られず，ハードウェア本来の性能を引き出すのが難しくなっている．よって，**計算量とデータ移動量の正確な推定を行ったり，データの特性（疎性，サイズ）と性能値の関係を見極める**ことによって，神戸に建設中の通称京速計算機などの**超大規模スーパーコンピュータでも性能を発揮できるソフトウェアの作成**を目指していくことになろう．

[4] http://web.kudpc.kyoto-u.ac.jp/hpc/

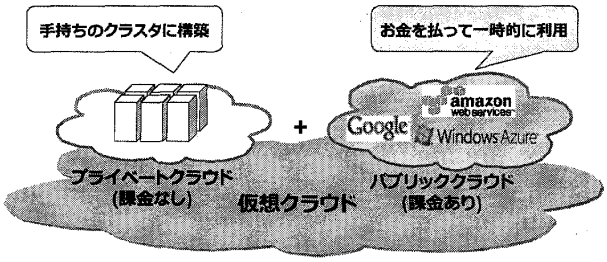
5.　クラウド・コンピューティングの適用

クラウド・コンピューティングとはクライアント側の端末はできるだけ軽く，安く抑えておいて，計算や検索などの処理およびデータの保存や管理はインターネット上のサーバで行うための技術の集合である．こういった思想は古くから存在していたが，近年のインターネットの高速化，安定化によってようやく実現ができるようになった．

クラウド・コンピューティングでは，クライアント端末はユーザの元にあり，サーバはインターネットを超えてユーザから遠くの場所に配置されている場合が多いのだが，最近では手持ちのサーバ上で仮想マシンを立ち上げて，ユーザからの要求（サーバの数や用途）に合わせて柔軟に対応していくプライベートクラウドなども注目を集めている．また，普段はプライベートクラウドを用いて，急に需要が増えた場合には外部のパブリッククラウドの資源を追加する使い方も提案，実施されている（仮想クラウド：図4参照）．

最適化問題に対するクラウド・コンピューティングの適用には様々な方法を考えることができるが，代表的な方法は以下の通りである．

1. サーバ上に最適化ソフトウェアをインストールした後に，Webアプリケーションとしてユーザにサービスだけを提供する（以前はASP，現在はSaaSと呼ばれる）．
2. 大規模最適化問題に対して数値実験等を行うために，必要なときに，必要な量だけ計算機資源をインターネット上から調達してくる（IaaSと呼ばれ，Amazon EC2，S3などのサービスが有名）．



- **クラウド上の大規模ジョブ実行環境**
  - プライベートクラウド：ローカルクラスタ上に構築
  - パブリッククラウド：クラウドベンダから有料でレンタル
- **ユーザからは一つの大きな実行環境として見える**

図4　仮想クラウド環境

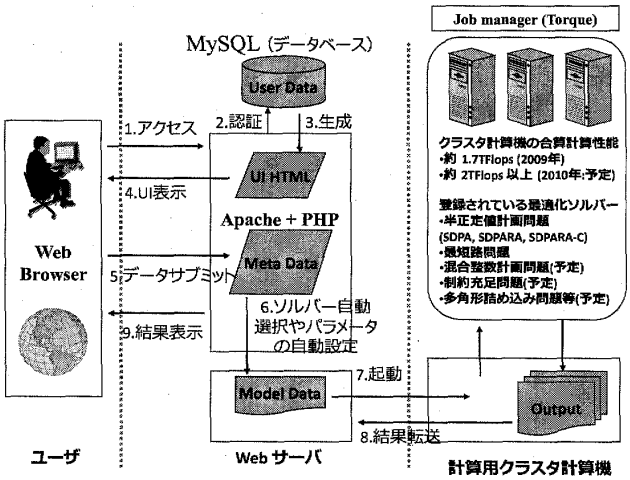


図5　最適化問題用オンライン・ソルバー

最近のノートパソコンでは中級以上の機種では高速なCPUと数Gbytesのメモリを搭載しているので，ノートパソコンでも大きな最適化問題を解くことは可能になってきた．ただしネットブックと呼ばれる数万円のノートパソコンやPDA（携帯情報端末）では，計算能力等が非力なので単独である程度大きな最適化問題を解くことは難しい．ところが上記の1のようなクラウド・コンピューティングの技術を使うことによって，端末側で必要な機能はサーバとインターネットを通じて通信を行うことと，データを入力したり結果を表示したりすることに限定することができる．著者らのグループではクラスタ・コンピューティングやクラウド・コンピューティングなどを技術を活用した最適化問題用のオンライン・ソルバーの開発と公開を行っている．現在SDP[5]および最短路問題[6]に対するオンライン・ソルバーが稼動しているが，今後は他の最適化問題に対するサービスも開始する予定である（図5）．

[5] http://sdpa.indsys.chuo-u.ac.jp/portal/
[6] http://opt.indsys.chuo-u.ac.jp/portal/

表3　SDPA 7.3.1とSDPA 2.0.1の比較実験：mcp500-1. dat-s

SDPA 7.3.1(2009年)	1.3秒
SDPA 2.0.1(2009年)	504.7秒
SDPA 2.0.1(1996年)	133,892.5秒

1996年：SONY NEWS-5000WI, CPU MIPS R4400 133MHz, memory 128MB
2009年：Dell PowerEdge R710, CPU Intel Xeon X5550 2.66GHz, memory 72GB

6. ソフトウェアSDPAの実装方式について

著者らのグループではすでに述べたように代表的な数理計画問題であるSDPに対する内点法を記述したソフトウェアSDPAの開発・評価・公開を10年以上行っている[7]．最近では以下のような開発方針を採用している．

1. 近年のソフトウェアにおいては疎データの扱い，疎構造の記述方法が重要になっているが，入力問題に対して自動的に適切な前処理を行い，密データ構造と疎データ構造のどちらで処理するかについて判断を行う．
2. 通常，ソフトウェアのボトルネックはCPUコア内部（浮動小数点演算能力やL2キャッシュの帯域幅など）に存在するか，CPUとメモリ間の帯域幅によるものかのどちらかである場合が多い．この特性を用いて，アルゴリズムの各部分のボトルネック構造を解明することによって適切なアルゴリズムを選択することができる．
3. 数値精度が問題になる場合には，任意精度演算などを用いて得られる精度と計算時間とのトレードオフなどに注目しながら，適切な方法を選択していく．

表3はSDPを解くためのソフトウェアSDPAの比較実験の結果を示したものである．表3には1996年にリリースされたSDPA 2.0.1を用いてmcp500-1. dat-s（ベンチマーク問題集SDPLIB[8]に収録されている）というSDPを解いた場合の結果が含まれているが[9]，当時ワークステーション（SONY NEWS）で133,892.5秒（およそ37.2時間）の時間を要した．またSDPA 2.0.1を現在のワークステーション（Dell

[7] http://sdpa.indsys.chuo-u.ac.jp/sdpa/
[8] http://www.nmt.edu/~borchers/sdplib.html
[9] この問題の大きさはn(行列の大きさ)＝m(制約式の数)＝500である．

PowerEdge R710）で実行して mcp500-1. dat-s を解いた場合には 504.7 秒で解いている．よって粗く計算すれば 1996 年から 2009 年にかけて計算機の進歩によりソフトウェアが 133,892.5/504.7≈265.3 倍高速化されていることがわかる．また最新の SDPA 7.3.1 で mcp500-1. dat-s を解いた場合で 1.3 秒で解くことができるので，アルゴリズムによる高速化も同様に計算すると 504.7/1.3≈388.2 倍高速化されていることになるので計算機の高速化だけでなくアルゴリズム等の高速化も主因の一つであると考えることができる．このようにソフトウェアの高速化には計算機やアルゴリズムの高速化など様々な要因が関係していることがわかる．

謝辞　テキサス大学オースティン校研究員（現 米マイクロソフト）の後藤和茂さんや中央大学大学院の安井雄一郎さんには助言や協力等を頂きました．また最適化オンライン・ソルバーに関しては文部科学省科学研究費基盤研究（C）20510143 および中央大学特定課題研究費の助成を受けています．

参考文献

[1] 中田和秀, 藤澤克樹, 福田光浩, 山下真, 中田真秀, 小林和博, 最適化ソフトウェア SDPA, 応用数理, Vol. 18, No. 1, 2-14, (2008).

[2] K. Fujisawa, K. Nakata, M. Yamashita and M. Fukuda : SDPA Project : Solving Large-scale Semidenite Programs, *Journal of the Operations Research Society of Japan*, Vol. 50, No. 4, 278-298, (2007).

[3] 藤澤克樹, 梅谷俊治, 応用に役立つ 50 の最適化問題, 朝倉書店, (2009).