SDPA（半正定値計画問題に対するソフトウェア）

藤沢　克樹

1. はじめに

本稿では, SDP (半正定値計画問題) という最近特に注目を集めている数理計画問題を取り上げて, SDPA (SDP に対する主双対内点法のソフトウエア) についての解説を行います. 今回は SDP を題材に取り上げますが, 解説していく数理・情報系の技術は他の最適化手法においてもすでに適用されていたり, 今後適用されていくと予想されるものがほとんどです. また数式や理論的な用語をあまり使用しないようにして, SDPA の紹介, 使用方法, 適用例及び将来の開発計画などについて解説していきます. なお SDP と SDP に対する主双対内点法に興味を持たれた方には, 土谷先生の書かれた日本語の解説 [1, 2] をお勧めします.

SDP に関する研究は意外と古くから行われていますが, SDP という名前が頻繁に使用されるようになって, 活発に研究が行われ始めたのは 1993 ～ 94 年ぐらいからです. 当時, SDP は組合せ最適化問題に対して非常に有効であると言われていました [3]. しかし, SDP に対するソフトウェア (SDPA も含む) が開発されるにつれて, 小さい規模の組合せ最適化問題に対してさえも SDP が高速に解けないことが判明してきました. 私自身も 1995 年から SDPA の開発を行い, 1995 年の 5 月ごろにはすでに SDPA の first version (fast version ではない) が完成していましたが, 非常に実行が遅く, 規模の小さい問題しか扱えなかったことをはっきりと記憶しています.

しかし最近 (特に 1998 年以降) SDP などの最適化手法が再び大きく注目を集めています. その理由には以下のようなものが考えられます. 例えば建築構造物や航空機などの構造設計においては, 単にデザイン的, 機能的に複雑な要求を満たす設計を見出すだけでなく, できるだけ速く, 安く完成させることも非常に重要です. そのためには前もって様々な要求を制約条件として記述して最も効率の良い解を求めていく最適設計を行なっていくことが好ましいわけです. しかし従来は大規模な構造物に対して最適設計を行なうことは, 最適化手法や計算機の能力からみても非常に困難でした. しかし最近の (SDP などの) 最適化手法と計算機環境の著しい進歩によって最適設計を目指す試みは加速されて, 現実味を帯びはじめています.

同時に SDP に対する理論的な研究も飛躍を遂げて, 新たな可能性も見え始めました. SDP は線形計画問題 (LP) や凸二次計画問題などを含んだ, より大きな凸計画問題の枠組ですが, 半正定値制約という非線形制約を持っています. 従って SDP として定式化できる最適化問題が解けるだけでなく, 非凸最適化問題に対する強力な緩和値を導き出すことができます. そのため SDP を繰り返して解くことによって (最適に解くことが極めて難しいが非常に実用上重要な) 非凸最適化問題を扱える可能性を持っています [4]. また, 著者らが開発したソフトウェア SDPA (SemiDefinite Programming Algorithm) [5][1] をはじめとして, 複数の研究グループによって SDP に対するソフトウェアが開発されて, インターネットより公開されています. さらに, ここ数年の間に多くの実験的解析が行われて, それらの結果をフィードバックすることにより SDPA のアルゴリズム自体も進歩を遂げました [6, 7].

また最適設計を目指して, 複雑で大規模な問題を解くためには, 理論的成果を随時組み入れると共に最新のコンピュータ技術 (並列, 広域計算) 等との融合も必要不可欠です [8]. そのため SDPA などの最適化手法を組み込んだ広域並列計算システムを現在開発中です (既に一部は稼働中). この SDPA を用いて大規模な SDP を解くための広域並列計算システムについても解説を行います.

ふじさわ　かつき　京都大学大学院工学研究科
〒 606-8501 京都市左京区吉田本町
e-mail: katsuki@archi.kyoto-u.ac.jp

[1] ftp://ftp.is.titech.ac.jp/pub/OpRes/software/SDPA/

2. 半正定値計画問題(SDP)の定義

なるべく数式などを使用せずにSDPの説明を行いたいのですが線形計画問題や整数計画問題のように適用例を見せて,読者に(何となく)納得させてしまうという技法はSDPでは非常に使いにくいのです.日本だけでなく,世界中でそのような例は見たことがありません.興味のある方は,少し面倒でもSDPの定式化を理解することをお勧めします.

ここからSDPに関する諸定義を行います.$\Re^{n\times n}$を$n\times n$の実行列の集合,$\mathcal{S}^n$を$n\times n$の実対称行列の集合とします.また任意の$\boldsymbol{X},\boldsymbol{Y}\in\Re^{n\times n}$に対して,$\boldsymbol{X}\bullet\boldsymbol{Y}$は$\boldsymbol{X}$と$\boldsymbol{Y}$の内積,すなわち,Tr $\boldsymbol{X}^T\boldsymbol{Y}$($\boldsymbol{X}^T\boldsymbol{Y}$のtrace:固有和)を表します.$\boldsymbol{X}\succ\boldsymbol{O}$は$\boldsymbol{X}\in\mathcal{S}^n$が正定値,つまり任意の$\boldsymbol{u}(\neq 0)\in\Re^n$に対し$\boldsymbol{u}^T\boldsymbol{X}\boldsymbol{u}>0$であることを示しています.また$\boldsymbol{X}\succeq\boldsymbol{O}$は$\boldsymbol{X}\in\mathcal{S}^n$が半正定値,つまり任意の$\boldsymbol{u}\in\Re^n$に対し$\boldsymbol{u}^T\boldsymbol{X}\boldsymbol{u}\geq 0$であることを示しています.

$\boldsymbol{F}_i\in\mathcal{S}^n$ $(0\leq i\leq m)$, $c_i,x_i\in\Re$ $(1\leq i\leq m)$, $\boldsymbol{X}\in\mathcal{S}^n,\boldsymbol{Y}\in\mathcal{S}^n$とします.このときSDPの主問題と双対問題は以下のように定義することができます.

$$
\left.\begin{array}{ll}
\textbf{主問題:} & \\
\text{最小化} & \displaystyle\sum_{i=1}^{m} c_i x_i \\
\text{制約条件} & \boldsymbol{X}=\displaystyle\sum_{i=1}^{m}\boldsymbol{F}_i x_i-\boldsymbol{F}_0, \\
 & \boldsymbol{X}\succeq\boldsymbol{O}. \\
\textbf{双対問題:} & \\
\text{最大化} & \boldsymbol{F}_0\bullet\boldsymbol{Y} \\
\text{制約条件} & \boldsymbol{F}_i\bullet\boldsymbol{Y}=c_i\ (1\leq i\leq m), \\
 & \boldsymbol{Y}\succeq\boldsymbol{O}.
\end{array}\right\} \qquad (1)
$$

ここで$\boldsymbol{F}_i\in\mathcal{S}^n(1\leq i\leq m)$が線形独立であることを仮定します.$(\boldsymbol{X},\boldsymbol{x},\boldsymbol{Y})$がSDPの実行可能解であるとは,$(\boldsymbol{X},\boldsymbol{x})$が主問題の実行可能解であり,$\boldsymbol{Y}$が双対問題の実行可能解であることを意味します.また,$(\boldsymbol{X},\boldsymbol{x},\boldsymbol{Y})$がSDPの実行可能内点解であるとは,$(\boldsymbol{X},\boldsymbol{x})$が主問題の実行可能内点解(つまり,$\boldsymbol{X}\succ\boldsymbol{O}$を満たす実行可能解)であり,$\boldsymbol{Y}$が双対問題の実行可能内点解(つまり,$\boldsymbol{Y}\succ\boldsymbol{O}$を満たす実行可能解)であることを意味します.

感覚的に理解するのは難しいのですが,半正定値制約$(\boldsymbol{X}\succeq\boldsymbol{O},\boldsymbol{Y}\succeq\boldsymbol{O})$がついているので,半正定値計画問題と呼ぶというように覚えておいてください.半正定値制約のみが非線形であとは線形の制約ですが,この非線形制約が付加されているおかげで先程も述べましたように非凸最適化問題に対する強力な緩和値を導きだすことが可能になっています.

3. SDPの応用例:構造最適化

SDPの適用が期待されている工学的分野には構造最適化[9],システムと制御[10],組合せ最適化[3]などがあります.しかしSDPAのユーザーからの問い合わせを見ていますと,ファイナンスや量子化学などの分野にも広く適用されているようです.

本稿ではSDPの応用例として一次固有振動数制約下でのトラス構造最適化問題を解説します.詳細はohsakiらの論文[9]を御覧下さい.この問題の目的,制約及び設計変数は以下の通りです.

目的:全部材体積(質量)を最小化
制約:トラスの全ての固有振動数が指定値以上
設計変数:部材断面積を求める

はじめに以下のような諸定義を行います.

L_i:i番目の部材の長さ(定数)
A_i:i番目の部材の断面積(設計変数)
Ω_r:r次の固有値
$\bar{\Omega}$:指定一次固有値
N^d:変位の自由度の総数
N^m:部材の総数
$\boldsymbol{K}_i$:i番目の部材の剛性行列への寄与
$\boldsymbol{M}_i$:i番目の部材の質量行列への寄与
$\boldsymbol{M}_0$:非構造質量に対する質量行列

このとき一次固有振動数制約下でのトラス構造最適化問題は以下のように定式化することができます.

$$
\begin{array}{ll}
\text{最小化} & \displaystyle\sum_{i=1}^{N^m} L_i A_i \\
\text{制約条件} & \Omega_r\geq\bar{\Omega},\ (r=1,2,\cdots,N^d) \\
 & A_i\geq 0,\ (i=1,2,\cdots,N^m)
\end{array} \qquad (2)
$$

目的関数は,全部材の体積を表しています.また,制約式$\Omega_r\geq\bar{\Omega}$は,トラスの全ての固有振動数が指定値以上という制約で,$A_i\geq 0$は部材の断面積が非負という制約になります.さらに(3)のようにSDPとして定式化することができます.ここで注意していただきたいのは,(3)のSDPの定式化は(2)の定式化の緩和ではないということです.つまりSDPAなどのソフトウェアを用いて(3)のSDPの最適解を求めれば,(2)の

構造最適化問題の最適解が求まることになります.

$$\left.\begin{array}{ll} \text{最小化} & \displaystyle\sum_{i=1}^{N^m} L_i A_i \\ \text{制約条件} & \boldsymbol{X} = \displaystyle\sum_{i=1}^{N^m} (\boldsymbol{K}_i - \bar{\Omega}\boldsymbol{M}_i) A_i - \bar{\Omega}\boldsymbol{M}_0 \\ & A_i \geq 0 \ \ (i = 1, 2, \cdots, N^m), \\ & \boldsymbol{X} \in \mathcal{S}^{N^d}, \boldsymbol{X} \succeq \boldsymbol{O} \\ \text{最大化} & \bar{\Omega}\boldsymbol{M}_0 \bullet \boldsymbol{Y} \\ \text{制約条件} & (\boldsymbol{K}_i - \bar{\Omega}\boldsymbol{M}_i) \bullet \boldsymbol{Y} = L_i \\ & (i = 1, 2, \cdots, N^m), \\ & \boldsymbol{Y} \in \mathcal{S}^{N^d}, \boldsymbol{Y} \succeq \boldsymbol{O} \end{array}\right\} (3)$$

次に図 1 のような平面アーチ格子のトラス構造最適化問題を SDPA を用いて解いた結果 (図 2) を示します. 図 1 で ● は重さ 2.100×10^4 kg の非構造質量を表しています. 構造質量とはその構造物を構成している部材等の質量のことですが, 非構造質量とは建物の中の人間や家具などの構造物以外の質量のことです. 図 1 からこの平面アーチが y 軸に対して対称である

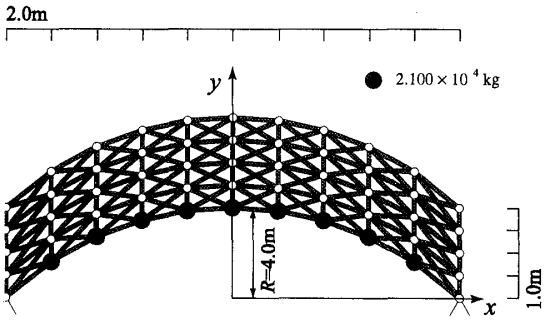

図 1: 平面アーチ格子, つまり SDPA 使用前

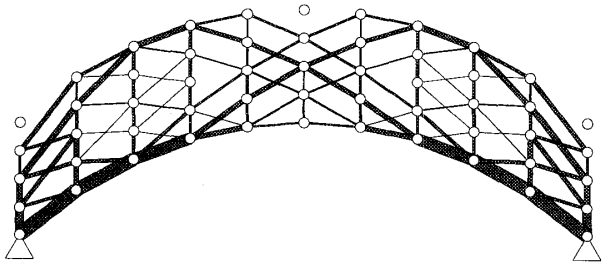

図 2: 平面アーチ格子の最適トポロジー, つまり SDPA 使用後

ことがわかりますが, 同時に図 2 より SDPA によって得られた最適解も y 軸に対して対称であることがわかります. 最近の研究によって対称な最適解を持つ, この種の構造最適化問題を, 主双対内点法のソフトウェア (SDPA など) で解いたときには必ず対称な最適解が得られることがわかってきました. 対称な構造物の方が景観や作りやすさなどの面で好ましいので, 理論的にも実用的にも非常に興味深いところです.

次に構造最適化の問題を用いて SDP のソフトウェア間での比較実験の結果を示します. 使用する問題は図 1 の問題と図 3 の問題です. この数値実験で使用するソフトウェアは SDPA(Ver5.0), CSDP(Ver 2.3) [2], SeDuMi(Ver 1.02) [3], SDPT3(Ver 2.1) [4] です.

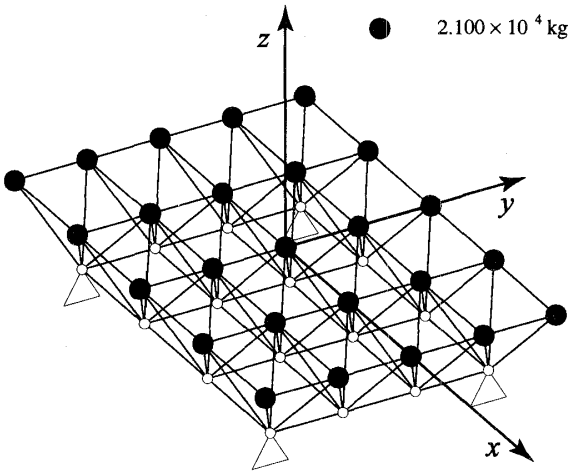

図 3: 2 層格子, これも SDPA 使用前

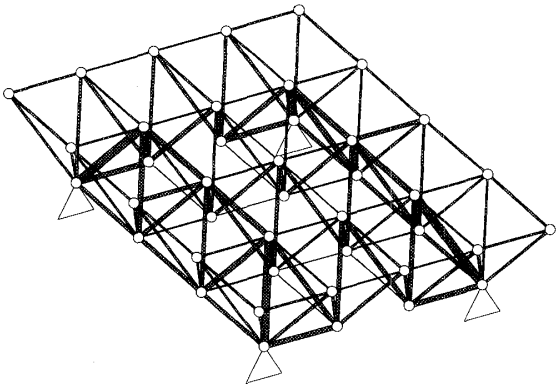

図 4: 2 層格子の最適トポロジー, これも SDPA 使用後

表 1 は比較実験結果です. 計算機は DEC ALPHA CPU 21164 (600MHz) を用いました. この表から SDPA が非常に高速に効率良く問題を解いていることがわかります.

[2] http://www.nmt.edu/~borchers/csdp.html
[3] http://www.unimaas.nl/~sturm/software/sedumi.html
[4] http://www.math.nus.sg/~mattohkc/index.html

表 1: 構造最適化問題 (一次固有振動数制約) に対する数値実験結果

問題	2 層格子	平面アーチ格子
大きさ	$N^d = 111, N^m = 128$	$N^d = 106, N^m = 174$
	実行時間 (反復回数)	実行時間 (反復回数)
SDPA	5.0(20)	5.4(23)
CSDP	26.0(24)	133.2(30)
SeDuMi	33.2(23)	124.6(36)
SDPT3	21.0(15)	22.2(22)

4. SDPA の使用方法 (入力ファイル編)

SDPA の使用方法に関してはマニュアル [5] に解説してあります. SDPA を使用するために複数の方法を用意していますが, この節では入力ファイルを用いて SDPA を使用する簡単な例を紹介します。詳細は SDPA のマニュアル [5] を参照下さい.

例えば以下のような SDP を SDPA で解くとします.

$$
\begin{aligned}
\text{最大化} \quad & \begin{pmatrix} -11 & 0 \\ 0 & 23 \end{pmatrix} \bullet \boldsymbol{Y} \\
\text{制約条件} \quad & \begin{pmatrix} 10 & 4 \\ 4 & 0 \end{pmatrix} \bullet \boldsymbol{Y} = 48, \\
& \begin{pmatrix} 0 & 0 \\ 0 & -8 \end{pmatrix} \bullet \boldsymbol{Y} = -8 \\
& \begin{pmatrix} 0 & -8 \\ -8 & -2 \end{pmatrix} \bullet \boldsymbol{Y} = 20, \\
& \boldsymbol{Y} \succeq \boldsymbol{O}.
\end{aligned}
$$

このとき以下のように整理しますと, この問題は (1) の双対問題に相当します.

$$
m = 3,\ n = 2,\ \boldsymbol{c} = \begin{pmatrix} 48 \\ -8 \\ 20 \end{pmatrix},
$$

$$
\boldsymbol{F}_0 = \begin{pmatrix} -11 & 0 \\ 0 & 23 \end{pmatrix},\ \boldsymbol{F}_1 = \begin{pmatrix} 10 & 4 \\ 4 & 0 \end{pmatrix},
$$

$$
\boldsymbol{F}_2 = \begin{pmatrix} 0 & 0 \\ 0 & -8 \end{pmatrix},\ \boldsymbol{F}_3 = \begin{pmatrix} 0 & -8 \\ -8 & -2 \end{pmatrix}.
$$

また SDPA の入力ファイルは以下のようになります.

```
3
1
2
{48, -8, 20}
{ {-11,  0}, { 0, 23} }
{ { 10,  4}, { 4,  0} }
{ {  0,  0}, { 0, -8} }
{ {  0, -8}, {-8, -2} }
```

SDPA はブロック対角な行列を直接扱うためのデータ構造を持っています. 詳細は SDPA のマニュアル [5] を参照下さい. この場合は 2×2 の行列をブロック数が 1 で第 1 ブロックが 2×2 の行列であると考えます.

```
3
1
2
```

の部分は制約式が 3 本 ($m = 3$), ブロック数が 1 , 第 1 ブロックの行列の大きさが 2×2 であることを意味します. 後は, 右辺定数 (c) → 目的関数の行列 ($\boldsymbol{F}_0$) → 制約式の行列 ($\boldsymbol{F}_1, \ldots, \boldsymbol{F}_m$) の順で入力します.

次にこのファイルを保存して (ここでは example1.dat という名前にします),

```
sdpa example1.dat out
```

として実行します. 実行後に出力ファイル out を見ますと

出力ファイル out

```
            略
objValPrimal    = -4.190000e+01
objValDual      = -4.190000e+01
            中略
.yMat =
{ {+5.90000000E+00,-1.37500000E+00 },
  {-1.37500000E+00,+1.00000000E+00 }   }
```

なので, 最適目的関数値は -41.9 で 行列 Y は,

$$
\begin{pmatrix} 5.9 & -1.375 \\ -1.375 & 1.0 \end{pmatrix}
$$

であることがわかります.

5. SDPA の使用方法 (NEOS サーバー編)

この節で説明する SDPA の使用方法は, インターネットが使える環境は持っているが SDPA が実行できる環境を持っていない (Windows や Mac OS など). あるいは速い計算機を持っていない方などにお勧めで

す. 以下のアドレスを netscape 等のブラウザに入力しますと NEOS サーバーを介して SDPA を使用することができます.

http://www-neos.mcs.anl.gov/neos/solvers/SDP:SDPA/

NEOS サーバーは米国の Argonne 研究所が運用している計算サーバーです. SDPA の他にも LOQO, MINOS, XPRESS-MP などの有名な数理計画法のソフトウェアが登録されていて自由に使用することができます. 最初に netscape などのブラウザや e-mail を用いて入力ファイルを NEOS サーバーに転送します. NEOS サーバーは送られてきた問題を解いて実行結果をユーザーにインターネットを通じて送り返す仕組みになっています. 次に簡単な使用方法を解説します (2000 年 1 月現在).

1. 上記のアドレスを netscape などのブラウザに入力して, 図 5 の画面を開きます.

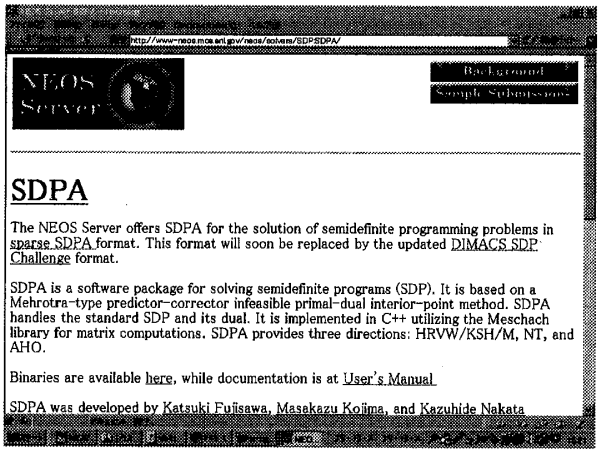

図 5: NEOS サーバー上での SDPA

2. このページの一番下の Interfaces to SDPA から World Wide Web を選択します.

3. WWW Interface SDPA のページの SDPA data の部分でハードディスク上にある入力データファイルを指定します.

4. このページの一番下の Submit to NEOS を選択して NEOS サーバーに問題を転送します.

5. NEOS サーバーでの実行が終了すると実行結果が送られてきます (図 6).

6. SDPA の Ninf(広域並列計算システム) への適用

最後に現在開発中の Ninf(広域並列計算システム) 上で動作する新しい SDPA について解説します. この

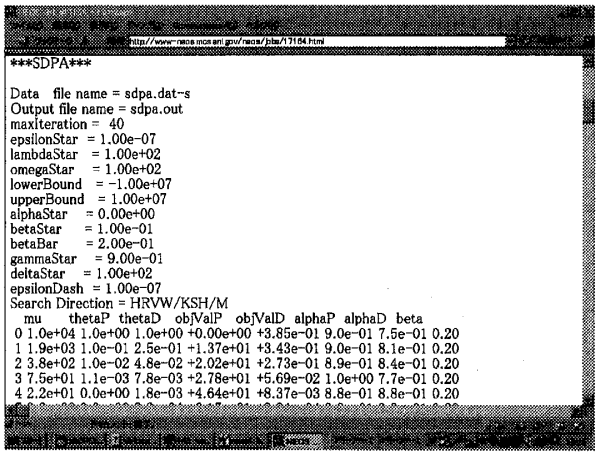

図 6: NEOS サーバー上での SDPA の実行結果の例

研究は, 東京工業大学 数理・計算科学専攻の小島 (政) 研究室と松岡研究室及び電子技術総合研究所の Ninf 開発チームとの共同研究です.

ソフトウェアの並列化の技術では PVM や MPI などが有名ですが, Ninf [11] はこれらの技術とは異なった特徴を持っています. 私達の周辺を見回しますと, やや古くなりつつある計算機などが使用されずに遊んでいるのを見掛けることがあります. しかしその遊休計算機資源の CPU パワーを他の計算機から利用するのは簡単ではありません. さらに離れた場所にある計算機を利用する場合にはネットワークの速度という問題も生じてきます.

そこで, 遠隔地の計算機同士を高速なネットワークて接続して, お互いの計算機資源 (特に CPU パワー) を有効に活用し, 大規模な問題を効率良く解くことが Ninf の主要な目的の一つになります. 今回は現在開発中の Ninf 上で並列に動作する SDPA を題材に取って解説を行います.

図 7 は今回の研究で用いるネットワーク資源を表しています. 当面の目標にしている非凸最適化問題を解くためには SDP を繰り返して解く反復解法を作成する必要があります. しかし 1 反復の中で解かなければならない複数の SDP は異なる計算機で非同期に解くことができます. つまり 1 反復中の複数の SDP は独立に解くことが可能で, 全ての SDP を解き終れば次の反復に移行します. そのため大量の計算機資源が利用できれば非常に高速化されることになります. 図 7 で京都大学 (Kyoto) から電子技術総合研究所 (ETL) 及び東京工業大学 (TIT) 等の高速な計算機を利用する場合には SINET などの回線を利用することに

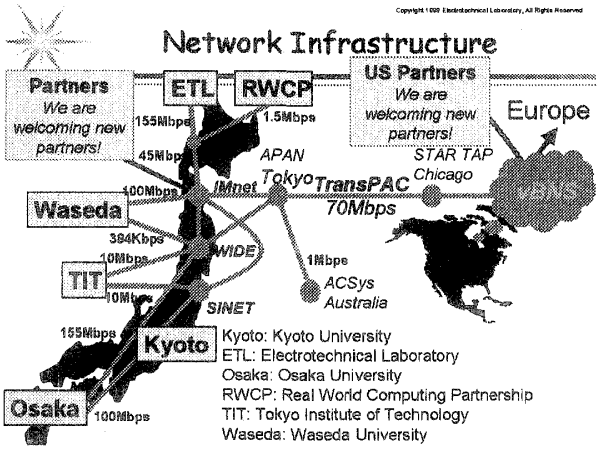

図 7: 広域計算システムのネットワーク資源

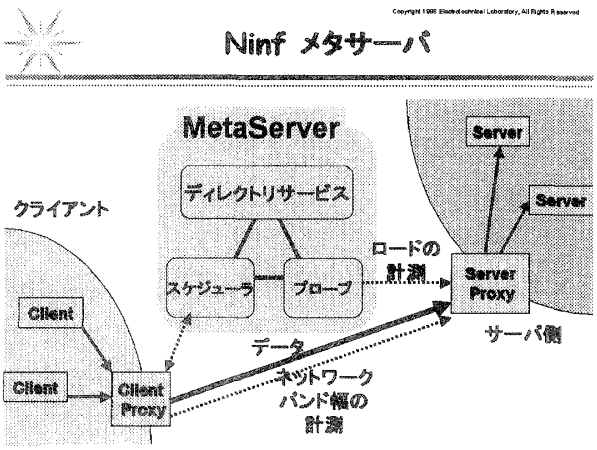

図 9: Ninf メタサーバー

なります. 現在開発中の SDPA では, データの転送量はあまり多くないので, この程度の速度 (100Mbps 以上) のネットワーク回線を用いることができるのならば, 全体でもかなりの高速化が期待できることが予備実験から判明しています.

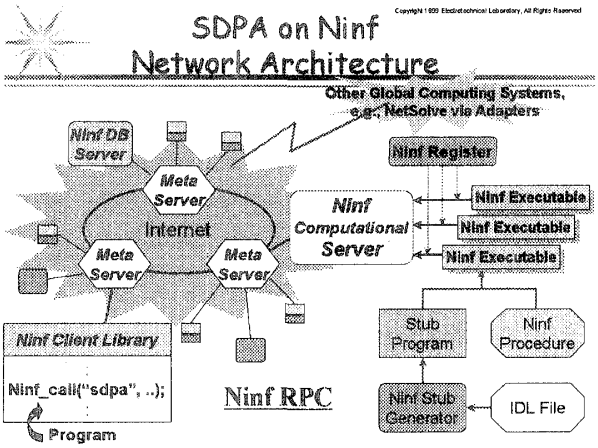

図 8: Ninf 上での SDPA

図 8 は Ninf 上で動作する SDPA の仕組みを表しています. 最初に複数の Ninf Computational Server に Ninf Executable (今回の場合は SDPA のライブラリ) を登録します. またこれら複数の Ninf Computational Server は, メタサーバー (Meta Server) によって管理されています. ここで図 8 の左下のようにユーザーが Ninf Client Library を用いて Meta Server に SDPA の計算要求を出したとします.

その場合メタサーバーは, 登録されているサーバー (Ninf Computational Server) のロードの計測を行ないます (図 9 参照). さらに大事なことは, ユーザー (クライアント) の計算機とサーバーとの間のネットワークのバンド幅の計測も行ないます. つまり, メタサーバーは自動的に負荷分散を行ない, 高速かつ高スループット計算を実現するサーバーを見つけ出して計算要求を出します. この場合ユーザーはサーバーの状態を直接知る必要が無いので大変便利です.

現在開発中のシステムなので, まだ実験データ等も少ないのですが, 開発の進展に合わせて徐々に発表, 公開していこうと考えています.

7. まとめ

SDPA の開発を始めてから約 4 年半ほどが過ぎました. この間, 既に述べましたようにアルゴリズムの改良や計算機の高速化などで SDPA も劇的な高速化を遂げました. 現時点でも大きな問題が扱えるようになりましたので, 今回解説しましたような構造最適化への応用などにも SDPA を使用しています. 今回は触れませんでしたが, 今年 (2000 年) は, さらに SDP のアルゴリズムを大きく見直して [12, 13], SDPA の大改造を行なう予定になっています (Ninf への適用も含む).

最後に Ninf に関して貴重な資料や情報等を提供してくださいました電子技術総合研究所の関口さん, 中田さん, 建部さん及び東京工業大学の松岡先生と中川さんには深く感謝致します.

参考文献

[1] 土谷 隆: 最適化アルゴリズムの新展開 - 内点法とその周辺 III 半正定値計画問題 I; システム/制御/情報, Vol. 42, No. 8, pp. 460–469, (1998).

[2] 土谷 隆: 最適化アルゴリズムの新展開 - 内点法とその周辺 IV 半正定値計画問題 II; システム/制

御/情報, Vol. 42, No. 10, pp. 550–559, (1998).

[3] M. X. Goemans and D. P. Williamson: Improved Approximation Algorithms for Maximum Cut and Satisfiability Problems Using Semidefinite Programming; *J. ACM*, 42, pp. 1115-1145, (1995).

[4] 小島政和: 半正定値計画緩和と大域的最適化; Research Report B-342, Dept. of Mathematical and Computing sciences, Tokyo Institute of Technology, Meguro, Tokyo, Japan, (1998).

[5] K. Fujisawa, M. Kojima and K. Nakata: SDPA(Semidefinite Programming Algortithm) User's Manual.; Research Report B-308, Dept. of Mathematical and Computing sciences, Tokyo Institute of Technology, Meguro, Tokyo, Japan, (1999).

[6] K. Fujisawa, M. Fukuda, M. Kojima and K. Nakata: Numerical evaluation of SDPA (Semidefinite Programming Algorithm); High Performance Optimization, Kluwer Academic Publishers, pp. 267–301, (2000).

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

[8] 鈴村 豊太郎, 中川 貴之, 松岡 聡, 中田 秀基: クライアント・サーバ型のグローバルコンピューティングシステムの比較 — Ninf, NetSolve, CORBA, Ninf-on-Globus の性能評価 —; 情報処理学会研究会報告 99-HPC-34, (1999).

[9] M. Ohsaki, K. Fujisawa, N. Katoh and Y. Kanno: Semi-Definite Programming for Topology Optimization of Truss under Multiple Eigenvalue Constraints; *Comput. Meth. Appl. Mech. Engng.*, Vol. 180, pp. 203–217, (1999).

[10] S. Boyd et al.: Linear Matrix Inequalities in Systems and Control Theory; *SIAM books* (1994).

[11] 中田秀基, 高木浩光, 松岡 聡, 長嶋雲兵, 佐藤 三久, 関口 智嗣: Ninf による広域分散並列計算; 情報処理学会論文誌, Vol. 39, No. 6, pp. 1818-1826, (1998).

[12] M. Fukuda, M. Kojima, K. Murota and K. Nakata: "Exploiting Sparsity in Semidefinite Programming via Matrix Completion I: General Framework, Research Report B-358, Dept. of Mathematical and Computing sciences, Tokyo Institute of Technology, Meguro, Tokyo, Japan, (1999).

[13] K. Nakata, K. Fujisawa, M. Fukuda, M. Kojima and K. Murota: "Exploiting Sparsity in Semidefinite Programming via Matrix Completion II: Implementation and Numerical Results, in preparation.