数理解析研究所講究録
1174巻 2000年 138-145

半正定値計画問題に対するソフトウェア SDPA の広域並列計算システム

The SDPA (SemiDefinite Programming Algorithm) on the Ninf (A Network based Information Library for the Global Computing)

藤沢 克樹 (FUJISAWA Katsuki)
京都大学大学院 工学研究科 建築学専攻
Department of Architecture and Architectural Systems, Kyoto University
武田 朗子 (TAKEDA Akiko)　小島 政和 (KOJIMA Masakazu)
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
Department of Mathematical and Computing Sciences, Tokyo Institute of Technology
中田 和秀 (NAKATA Kazuhide)
東京大学大学院 工学系研究科 物理工学専攻
Department of Applied Physics, The University of Tokyo

Abstract: In resent years, semidefinite program (SDP) has been intensively studied both in theoretical and practical aspects of various fields including interior-point methods, combinatorial optimization and the control and systems theory. The SDPA (SemiDefinite Programming Algorithm) [1] is an optimization software, implemented by C++ language, of a Mehrotra-type primal-dual predictor-corrector interior-point method for solving the standard form semidefinite program. The SDPA incorporates special data structures for handling block diagonal matrices and an efficient method proposed by Fujisawa *et al.* [2] for computing search directions when problems to be solved are large scale and sparse. In this paper, we also discuss parallel execution of the SDPA on the Ninf [3], a global network-wide computing infrastructure which has been developed for high-performance numerical computation services. Ninf is intended not only to exploit high performance in global network parallel computing, but also to provide a simple programming interface similar to conventional function calls in existing languages. Therefore, the SDPA on the Ninf enjoys the following features: 1. Solving many SDPs simultaneously without complicated implementation. 2. Computational resources including hardware and software distributed across a wide area network are available. We report some numerical results on a parallel implementation of the successive convex relaxation method proposed by Kojima and Tuncel [4] applying the SDPA on the Ninf.

Key words: semidefinite programming, primal-dual interior-point method, optimization software, global computing

1　はじめに

半正定値計画問題（Semidefinite Programming : SDP）は, 組合せ最適化, 制御分野, 構造最適化及びデータマイニングなどに幅広い応用を持ち, 主双対内点法によって多項式時間で最適解を導き出すことができるので, 最近では注目度も高く頻繁に研究が行われている. 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あるが, 制約に半正定値制約という非線形制約を持っている. 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る. 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る [4]. また, 著者らが開発したソフトウェア SDPA (SemiDefinite Programming Algorithm) [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, インターネットより公開されている. ここ数年の間に多くの実験的解析が

行われると共に, それらの結果をフィードバックすることにより SDP のアルゴリズム自体も進歩を遂げた [2]. さらに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あって [3],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は現在開発中である. 本論文では SDP と SDP に対する主双対内点法について触れたあとに 現在開発中の大規模な非凸最適化問題を逐次半正定値計画緩和法 [4] を用いて解くための SDPA の広域計算システムについて説明を行なう.

2 半正定値計画問題 (Semidefinite Programming : SDP)

SDP に対するサーベイには Vandenberghe と Boyd [6] などがある. また SDP に関する文献やソフトウェアなどの情報を集めたホームページも存在するので参考にしていただきたい[1] [2]. SDP が期待されている主な工学的分野には構造最適化 [7], システムと制御 [8], 組合せ最適化 [9] などがある.

2.1 SDP に関する諸定義

次に SDP に関する諸定義を行なう. $\Re^{n\times n}$ 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する. 任意の $\boldsymbol{X}, \boldsymbol{Z} \in \Re^{n\times n}$ に対して, $\boldsymbol{X}\bullet\boldsymbol{Z}$ は $\boldsymbol{X}$ と $\boldsymbol{Z}$ の内積, すなわち, Tr $\boldsymbol{X}^T\boldsymbol{Z}$ ($\boldsymbol{X}^T\boldsymbol{Z}$ の trace : 固有和) を表す. $\boldsymbol{X}\succ\boldsymbol{O}$ 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る. 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る.

$\boldsymbol{C}\in\mathcal{S}^n$, $\boldsymbol{A}_i\in\mathcal{S}^n$ $(1\leq i\leq m)$, $b_i, y_i\in\Re$ $(1\leq i\leq m)$, $\boldsymbol{X}\in\mathcal{S}^n, \boldsymbol{Z}\in\mathcal{S}^n$ とする. このとき,SDP の主問題と双対問題は以下のように与えられる.

$$
\left.
\begin{array}{lll}
\textbf{主問題：} & & \\
\text{最小化} & \boldsymbol{C}\bullet\boldsymbol{X} & \\
\text{制約条件} & \boldsymbol{A}_i\bullet\boldsymbol{X}=b_i\ (1\leq i\leq m), & \\
 & \boldsymbol{X}\succeq\boldsymbol{O}. & \\
\textbf{双対問題：} & & \\
\text{最大化} & \displaystyle\sum_{i=1}^{m} b_i y_i & \\
\text{制約条件} & \displaystyle\sum_{i=1}^{m}\boldsymbol{A}_i y_i+\boldsymbol{Z}=\boldsymbol{C}, & \\
 & \boldsymbol{Z}\succeq\boldsymbol{O}. &
\end{array}
\right\} \qquad (1)
$$

本解説を通して, $\boldsymbol{A}_i\in\mathcal{S}^n(1\leq i\leq m)$ が線形独立であることを仮定する. $(\boldsymbol{X},\boldsymbol{y},\boldsymbol{Z})$ が SDP の実行可能解であるとは, $\boldsymbol{X}$ が主問題の実行可能解であり,$(\boldsymbol{y},\boldsymbol{Z})$ が双対問題の実行可能解であることを表す. また, $(\boldsymbol{X},\boldsymbol{y},\boldsymbol{Z})$ が SDP の実行可能内点解であるとは, $\boldsymbol{X}$ 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, $(\boldsymbol{y},\boldsymbol{Z})$ が双対問題の実行可能内点解 (つまり, $\boldsymbol{Z}\succ\boldsymbol{O}$ を満たす実行可能解) の場合である.

[1] http://new-rutcor.rutgers.edu/~alizadeh/sdp.html
[2] http://www.zib.de/helmberg/semidef.html

2.2 SDP に対する主双対内点法

以下では, SDP の主双対内点法の枠組みについて簡単に述べる. SDP (1) には内点実行可能解が存在することを仮定する. このとき双対定理により, SDP (1) の最適解は以下の式を満たす.

$$\left.\begin{array}{l} \boldsymbol{A}_i \bullet \boldsymbol{X} = b_i \ (1 \leq i \leq m), \\ \sum_{i=1}^{m} \boldsymbol{A}_i y_i + \boldsymbol{Z} = \boldsymbol{C}, \\ \boldsymbol{X} \succeq \boldsymbol{O},\ \boldsymbol{Z} \succeq \boldsymbol{O},\ \boldsymbol{XZ} = \boldsymbol{O}. \end{array}\right\} \tag{2}$$

SDP に対する主双対内点法は線形計画問題に対する主双対内点法と同様に, 中心パス $\mathcal{P}$ に沿って進みながら最適解に収束する反復解法である. ここで, 中心パス $\mathcal{P}$ はパラメータ μ を用いて以下のように定義される.

$$\mathcal{P} = \left\{ (\boldsymbol{X}, \boldsymbol{y}, \boldsymbol{Z}) \left| \begin{array}{l} \boldsymbol{A}_i \bullet \boldsymbol{X} = b_i \\ (1 \leq i \leq m), \\ \sum_{i=1}^{m} \boldsymbol{A}_i y_i + \boldsymbol{Z} = \boldsymbol{C}, \\ \boldsymbol{X} \succ \boldsymbol{O},\ \boldsymbol{Z} \succ \boldsymbol{O}, \\ \boldsymbol{XZ} = \mu \boldsymbol{I} \ (\mu > 0). \end{array} \right. \right\}. \tag{3}$$

ただし $\boldsymbol{I} \in \mathcal{S}^n$ は単位行列を表す. 任意の $\mu > 0$ に対し, 中心パス $\mathcal{P}$ 上の点が一意に存在することが知られている. また, 中心パス $\mathcal{P}$ は SDP の内点実行可能解の集合の内部の滑らかな曲線となっており $\mu \to 0$ のとき SDP の最適解に収束することも知られている [10].

SDP の主双対内点法では, 探索方向パラメータ $\mu > 0$ を選び下記の条件を満たす中心パス上の点 $(\boldsymbol{X}, \boldsymbol{y}, \boldsymbol{Z})$ をターゲットとし, その Newton 方向に進む.

$$\left.\begin{array}{l} \boldsymbol{A}_i \bullet \boldsymbol{X} = b_i \ (1 \leq i \leq m), \\ \sum_{i=1}^{m} \boldsymbol{A}_i y_i + \boldsymbol{Z} = \boldsymbol{C},\ \boldsymbol{XZ} = \mu \boldsymbol{I} \end{array}\right\} \tag{4}$$

次に SDP に対する一般的な主双対内点法の概要を以下に示す. なお Mehrotra 型 [14] の主双対内点法が存在するが, Mehrotra 型は説明が複雑になるのでこちらは SDPA [15] を参考にしていただきたい.

SDP に対する主双対内点法

手順 0 . $k = 0$ として, $\boldsymbol{X}^0 \succ \boldsymbol{O}$, $\boldsymbol{Z}^0 \succ \boldsymbol{O}$ を満たす初期解 $(\boldsymbol{X}^0, \boldsymbol{y}^0, \boldsymbol{Z}^0)$ を選ぶ (実行可能解でなくてもよい).

手順 1 . 現在の解 $(\boldsymbol{X}^k, \boldsymbol{y}^k, \boldsymbol{Z}^k)$ が終了条件を満たすならば, アルゴリズムを終了する.

手順 2 . 以下のような条件を満たすような探索方向 $(d\boldsymbol{X}^k, d\boldsymbol{y}^k, d\boldsymbol{Z}^k)$ を計算する.

$$(\boldsymbol{X}^k + d\boldsymbol{X}^k, \boldsymbol{y}^k + d\boldsymbol{y}^k, \boldsymbol{Z}^k + d\boldsymbol{Z}^k) \in \mathcal{P}$$

具体的には探索方向を計算するために次の Newton 方程式を解く.

$$\begin{array}{l} \boldsymbol{A}_i \bullet (\boldsymbol{X}^k + d\boldsymbol{X}^k) = a_i \ (i = 1, 2, \ldots, m), \\ (\boldsymbol{Z}^k + d\boldsymbol{Z}^k) = \boldsymbol{C} - \sum_{i=1}^{m} \boldsymbol{A}_i (\boldsymbol{y}_i^k + d\boldsymbol{y}_i^k), \\ \boldsymbol{X}^k \boldsymbol{Z}^k + \boldsymbol{X}^k d\boldsymbol{Z}^k + d\boldsymbol{X}^k \boldsymbol{Z}^k = \mu \boldsymbol{I}, \\ d\boldsymbol{X}^k \in \mathcal{S}^n,\ d\boldsymbol{Z}^k \in \mathcal{S}^n,\ d\boldsymbol{y}^k \in R^m. \end{array}$$

手順3. 新しい反復点 $(\boldsymbol{X}^{k+1}, \boldsymbol{y}^{k+1}, \boldsymbol{Z}^{k+1})$ を次の条件

$$\boldsymbol{X}^{k+1} = \boldsymbol{X}^k + \alpha_p d\boldsymbol{X}^k \succ \boldsymbol{O},$$
$$\boldsymbol{Z}^{k+1} = \boldsymbol{Z}^k + \alpha_d d\boldsymbol{Z}^k \succ \boldsymbol{O},$$

を満たすように定める.

手順4. $k = k+1$ として, 手順1に戻る.

ただし, $\alpha_p, \alpha_d > 0$ はステップ長. (ステップ長をあまり大きく取ると新しい反復点が実行可能領域の境界に近づきすぎて, 以後の反復で困難を生ずる. また, 小さすぎると収束までに多くの反復回数を要する).

方向探索パラメータ μ とステップ長 α_p, α_d を適当に選ぶことにより, 大域的な収束性を保証することができる. また, ポテンシャル関数を評価関数として組み合わせて使うことも可能である.

3 SDPA と広域並列計算システム Ninf

3.1 Ninf 上における SDPA の並列実行

次に現在開発中の Ninf(広域並列計算システム) [3] 上で動作する新しい SDPA について説明を行う. ソフトウェアの並列化の技術では PVM や MPI などが有名だが, Ninf はこれらの技術とは異なった特徴を持っている. 私達の周辺を見回すと, 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る. 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なく, 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る. そこで, 遠隔地の計算機同士を高速なネットワークで接続して, 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っている. また, クラスタ計算機は複数の独立した計算機を高性能 LAN 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 (図 1) 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ーの数十倍から数百倍のコストパフォーマンスを達成している. 本論文では, これらの PC クラスタを用いた数値実験結果を報告する.

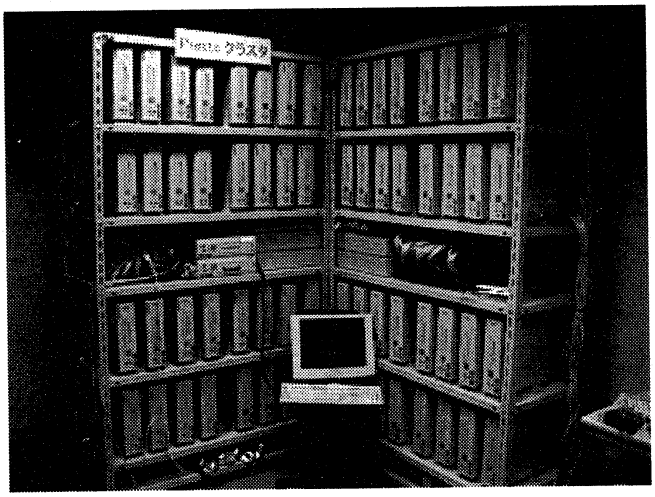

図 1: PC クラスタ (Pentium II 350MHz : 66 台) 東京工業大学 松岡研究室

図 2 は Ninf 上で動作する SDPA の仕組みを表している. 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) を登録する. Ninf Executable

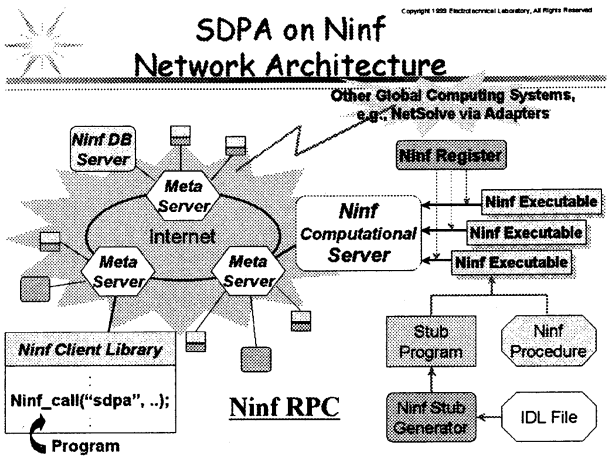

図 2: Ninf 上での SDPA

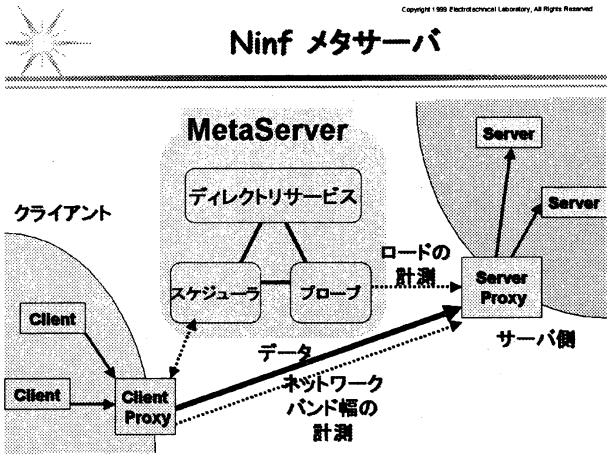

図 3: Ninf メタサーバー

は,UNIX 上などで C++, C, Fortran 等 (図 4) で記述されていれば容易に作成することが可能である. 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る. ここで図 2 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仮定する (Ninf_call("sdpa",...) を実行). その場合メタサーバーは, 登録されているサーバーのロードの計測を行って, 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う (図 3). その結果, 適当な Ninf Computational Server が指定されて計算が行われる. 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す仕組みになっている. この Ninf 上で並列動作する SDPA には以下のような特徴がある. 1: 通常の関数に似たインターフェイスを持っているので, 簡単な関数等の記述で並列動作する SDPA を実現することが可能である. 2: インターネット等によって広域に接続, 提供されているハードウェア, ソフトウェアの利用が可能である. 今回の数値実験では, PC クラスタ (CPU : Pentium III 500MHz 64 台 + Celeron 500MHz 64 台 = 128 台) を Ninf Computational Server として用いる.

図 4 は 図 2 の左下の Ninf Client Library を用いた Ninf の RPC API を簡単に説明したものである. Ninf では, 同期, 非同期の呼出しがサポートされており, 通常の関数呼出しに似た記述で実行することが可能になっている.

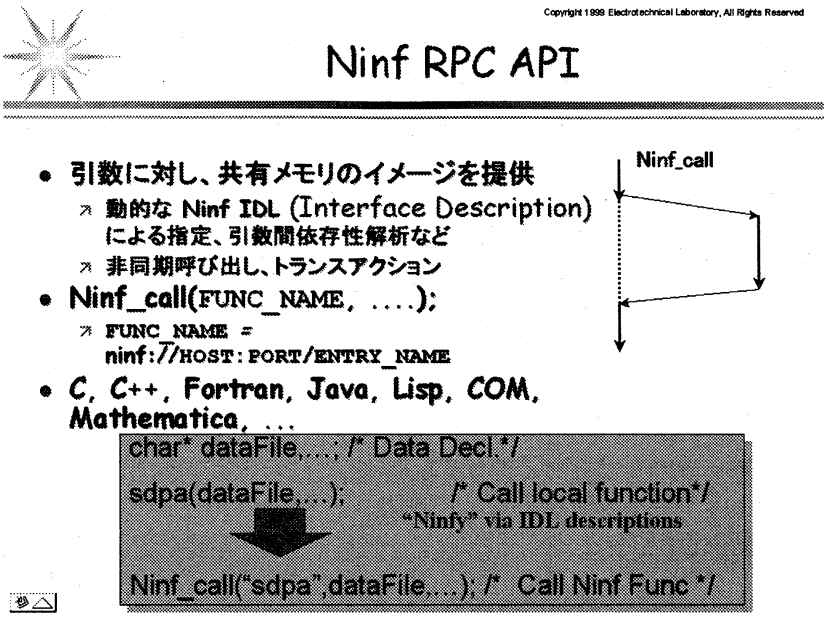

図 4: Ninf RPC API

3.2 逐次半正定値計画緩和法への適用

この節では, 以下のような非凸二次最適化問題 (NQP) を考える. ここで $\boldsymbol{c} \in \boldsymbol{R}^n, \boldsymbol{a}_i \in \boldsymbol{R}^n, \boldsymbol{q}_j \in \boldsymbol{R}^n$ は定数ベクトル, $b_i \in \boldsymbol{R}^1, \gamma_j \in \boldsymbol{R}^1$ は定数, $\boldsymbol{Q} \in \boldsymbol{R}^{n \times n}$ は $n \times n$ の定数行列である.

$$\text{(NQP)} \quad \left| \begin{array}{ll} \max & \boldsymbol{c}^T \boldsymbol{x} \\ \text{subj.to} & \boldsymbol{x} \in F \end{array} \right. \tag{5}$$

$$F \equiv \left\{ \boldsymbol{x} \in R^n : \begin{array}{r} \boldsymbol{a}_i^T \boldsymbol{x} + b_i \leq 0 \ \ (i = 1, \cdots, m_1) \\ \boldsymbol{x}^T \boldsymbol{Q}_j \boldsymbol{x} + \boldsymbol{q}_j^T \boldsymbol{x} + \gamma_j \leq 0 \ \ (j = 1, \cdots, m_2) \end{array} \right\}$$

ここで領域 F は有界であると仮定する. F は上記のように m_1 本の線形制約式と m_2 本の二次制約式から構成されている. ここで行列 $\boldsymbol{Q}$ は正定値でもなくてもよいので, LP や SDP だけでなく, 一般の非凸 (二次) 計画問題も含まれている. 武田ら [5] は NQP に対して, 逐次半正定値計画緩和法 [4] を適用して, Ninf 上で並列動作する SDPA を用いて計算を行っている. 逐次半正定値計画緩和法では, はじめに領域 F を完全に含む凸領域を定義する. そのあと LP 緩和や SDP 緩和などを繰り返し用いて, この領域が F の凸包に収束するまで計算を継続する. 目的関数は上記のように一般性を失うことなく線形関数 $(\boldsymbol{c}^T \boldsymbol{x})$ を仮定することができる. F 上での $\boldsymbol{c}^T \boldsymbol{x}$ の最大値と F の凸包上での $\boldsymbol{c}^T \boldsymbol{x}$ の最大値は一致するので, この性質を用いて NQP を解くのが大きな特徴である. このアルゴリズムでは 1 反復の中で複数の SDP 緩和問題を解かなければならない. これらの複数の SDP 緩和問題は前節で説明した Ninf を用いて, 異なる計算機上で非同期に解くことができる. その結果, 多くの計算機資源を利用することによって大幅な高速化が達成されている. 最後に前出の PC クラスタ (CPU : Pentium III 500MHz 64 台 + Celeron 500MHz 64 台 = 128 台) での実験結果を示す. 使用する問題は, 非凸二次計画問題 [5] で LC50-1296 ($n = 51$, $m_1 = 75$, $m_2 = 1$) と LC30-162 ($n = 31$, $m_1 = 45$, $m_2 = 1$) の二つである. 表 1 は PC クラスタでの CPU の台数と高速化の比率との関係を示している. 例えば CPU が 128 台のときに LC50-1296 では約 100 倍, LC30-162 では約 57 倍の高速化を達成している. 図 2 で紹介した Ninf Executable の起動に要するオーバーヘッドを考慮すると, 解く問題の規模が大きい方が

表 1: PC クラスタ (CPU 128 台) での CPU の台数と高速化の比率との関係

	LC50-1296		LC30-162	
CPU(台)	実行時間 (秒)	比率	実行時間 (秒)	比率
1	31050	1.000	4647	1.000
2	5571	1.994	2349	1.978
4	7830	3.966	1194	3.892
8	4003	7.757	618	7.519
16	2042	15.206	345	13.470
32	1029	30.175	213	21.817
64	565	54.96	159	29.226
128	310	100.161	82	56.67

Server での Ninf Executable の実行時間の占める割合が大きくなって, 並列化の効率は向上することになる. 通常 CPU 128 台を使用した場合に 100 倍の並列化効率が得られることは珍しい. これらの結果のように Ninf による高速化は目覚しく, SDPA だけでなく他の最適化ソフトウェアとの組合せも今後は期待されるところである.

謝辞 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先生, 中田先生, 建部先生及び東京工業大学の松岡先生と研究室の皆様には深く感謝致します.

参考文献

[1] K. Fujisawa, M. Kojima and K. Nakata, "SDPA (Semidefinite Programming Algorithm) – User's Manual –," Technical Report B-308, Tokyo Institute of Technology, Oh-Okayama, Meguro, Tokyo 152, Japan, Revised on May 1999.

[2] K. Fujisawa, M. Kojima and K. Nakata, "Exploiting sparsity in primal-dual interior-point methods for semidefinite programming," *Mathematical Programming* **79**, pp. 235–253, 1997.

[3] M. Sato, H. Nakada, S. Sekiguchi, S. Matsuoka, U. Nagashima and H. Takagi, "Ninf: A Network based Information Library for a Global World-Wide Computing Infrastructure", HPCN'97 (LNCS-1225), pp. 491-502, 1997.

[4] M. Kojima and L. Tuncel, "Discretization and Localization in Successive Convex Relaxation for Nonconvex Quadratic Optimization Problems", July 1998. Revised May 2000. to appear in Mathematical Programming.

[5] A. Takeda, K. Fujisawa, Y. Fukaya and M. Kojima, "A Parallel Successive Convex Relaxation Algorithm for Quadratic Optimization Problems", 文部省統計数理研究所共同研究リポート "最適化:モデリングとアルゴリズム" に掲載予定.

[6] L. Vandenberghe and S. Boyd: Semidefinite programming; *SIAM Review*, Vol. 38, pp. 49–95 (1996)

[7] M. Ohsaki, K. Fujisawa, N. Katoh and Y. Kanno: Semi-Definite Programming for Topology Optimization of Truss under Multiple Eigenvalue Constraints; to appear in *Comput. Meth. Appl. Mech. Engng.*, (1999)

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

[9] 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)

[10] M. Kojima, S. Shindoh and S. Hara: Interior-point methods for the monotone semidefinite linear complementarity problems; *SIAM J. Optim.*, Vol. 7, pp. 86–125, (1997)

[11] C. Helmberg, F. Rendl, R.J. Vanderbei and H. Wolkowicz: An interior-point method for semidefinite programming; *SIAM J. Optim.*, Vol. 6, No. 2, pp. 342–361, (1996)

[12] R. D. C. Monteiro: Primal-dual path-following algorithms for semidefinite programming; *SIAM J. Optim.*, Vol. 7, No. 3, pp. 663–678, (1997)

[13] M. J. Todd, K. C. Toh and R. H. Tütüncü: On the Nesterov-Todd direction in semidefinite programming; *SIAM J. Optim.*, Vol. 8, No. 3, pp. 769–796, (1998)

[14] S. Mehrotra: On the implementation of a primal-dual interior point method; *SIAM J. Optim.*, Vol 2, pp. 575–601, (1992)

[15] K. Fujisawa, M. Fukuda, M. Kojima and K. Nakata: Numerical evaluation of SDPA (Semidefinite Programming Algorithm); The Proceedings of the Second Workshop on High Performance Optimization Techniques (2000)