

## 広域分散コンピューティング環境における数理計画ソフトウェア SDPA

藤沢 克樹

京都大学大学院 工学研究科 建築学専攻

武田 朗子 小島 政和

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

中田 和秀

東京大学大学院 工学系研究科 物理工学専攻

概要: 近年, 半正定値計画法は数理計画法の分野において理論的研究がさかんに行われている. また同時に組合せ最適化, システムと制御理論, データマイニングなどの応用分野も同時に研究が行われている. 著者らは, 半正定値計画問題を解くための主双対内点法を実現したソフトウェア SDPA [1] の開発を行い, 数値実験によって有効性の検証を行ってきた. 本研究では, 広域分散コンピューティング環境である Ninf [3] を用いて SDPA の並列化を行い, 数理計画問題の中でも難しい範疇に属する非凸最適化問題を解くことを試みる.

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

FUJISAWA Katsuki

Department of Architecture and Architectural Systems, Kyoto University

TAKEDA Akiko KOJIMA Masakazu

Department of Mathematical and Computing Sciences, Tokyo Institute of Technology

NAKATA Kazuhide

Department of Applied Physics, The University of Tokyo

**Abstract:** In recent 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. 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. 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, grid 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り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)

### 2.1 SDP に関する諸定義

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させていただきたい<sup>1 2</sup>. 次に SDP に関する諸定義を行なう.  $\mathbb{R}^{n \times n}$  を  $n \times n$  の実行列の集合,  $S^n$  を  $n \times n$  の実対称行列の集合とする. 任意の  $X, Z \in \mathbb{R}^{n \times n}$  に対して,  $X \bullet Z$  は  $X$  と  $Z$  の内積, すなわち,  $\text{Tr } X^T Z$  ( $X^T Z$  の trace : 固有和) を表す.  $X \succ O$  は  $X \in S^n$  が正定値, つまり任意の  $u (\neq 0) \in \mathbb{R}^n$  に対し  $u^T X u > 0$  であることを示している. また  $X \succeq O$  は  $X \in S^n$  が半正定値, つまり任意の  $u \in \mathbb{R}^n$  に対し  $u^T X u \geq 0$  であることを示している.

$C \in S^n, A_i \in S^n (1 \leq i \leq m), b_i, y_i \in \mathbb{R} (1 \leq i \leq m), X \in S^n, Z \in S^n$  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.

$$\left. \begin{array}{l}
 \text{主問題 :} \\
 \text{最小化} \quad C \bullet X \\
 \text{制約条件} \quad A_i \bullet X = b_i \quad (1 \leq i \leq m), \\
 \quad \quad \quad X \succeq O. \\
 \\
 \text{双対問題 :} \\
 \text{最大化} \quad \sum_{i=1}^m b_i y_i \\
 \text{制約条件} \quad \sum_{i=1}^m A_i y_i + Z = C, \\
 \quad \quad \quad Z \succeq O.
 \end{array} \right\} \quad (1)$$

<sup>1</sup><http://new-rutcor.rutgers.edu/~alizadeh/sdp.html>

<sup>2</sup><http://www.zib.de/helmberg/semidef.html>

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

## 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.

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

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 (図 1). ここで, 中心パス  $\mathcal{P}$  はパラメータ  $\mu$  を用いて以下のように定義される.

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

ただし  $I \in S^n$  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  $\mu \rightarrow 0$  のとき SDP の最適解に収束することも知られている [7]. SDP の主双対内点法では, 適当な初期解から探索を始め, 探索方向パラメータ  $\mu > 0$  を選び, 中心パス上の点  $(X, y, Z)$  をターゲットとし, その探索方向を計算して, 実行可能領域からはみ出さないようにステップサイズを選択する. この計算を繰り返し, 最適解に収束したら探索を終了する (図 1).

## 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 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やや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

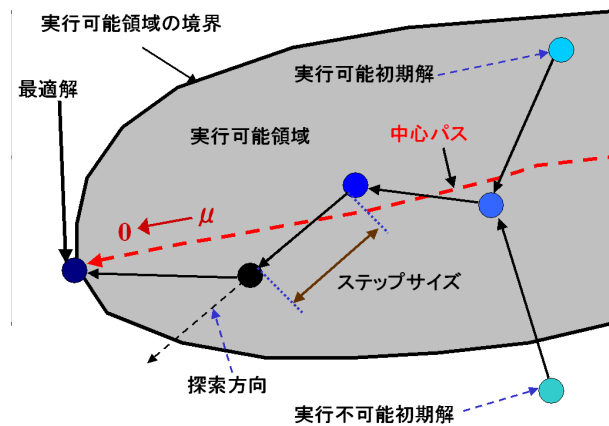


図 1: SDP に対する主双対内点法

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今回 Ninf を用いる主要な目的になっている。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。

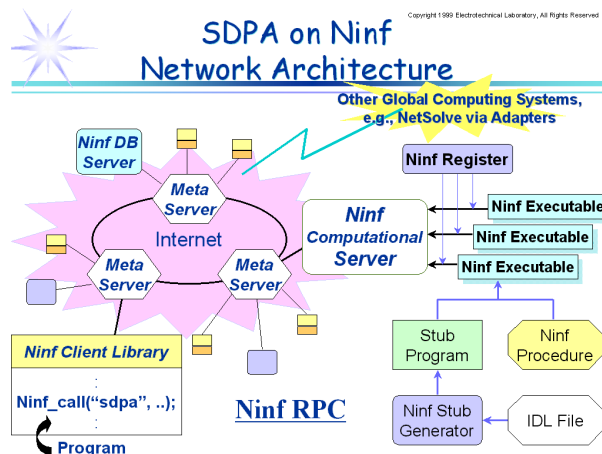


図 2: Ninf 上での SDPA

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 は、UNIX 上などで C++, C, Fortran 等 (図 3) で記述されていれば容易に作成することが可能である。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 800MHz 128 台, 東京工業大学松岡研究室) を Ninf Computational Server として用いる。



# Ninf RPC API

- 引数に対し、共有メモリのイメージを提供
  - 動的な Ninf IDL (Interface Description) による指定、引数間依存性解析など
  - 非同期呼び出し、トランザクション
- **Ninf\_call(FUNC\_NAME, ...);**
  - FUNC\_NAME = ninf://HOST:PORT/ENTRY\_NAME
- **C, C++, Fortran, Java, Lisp, COM, Mathematica, ...**

```
char* dataFile,...; /* Data Decl.*/
sdpa(dataFile,...); /* Call local function*/
                    /* "Ninfy" via IDL descriptions */
Ninf_call("sdpa",dataFile,...); /* Call Ninf Func */
```

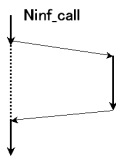


図 3: Ninf RPC API

図 3 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、非同期の呼び出しがサポートされており、通常の間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。

### 3.2 Ninf を用いた逐次半正定値計画緩和法の並列化

次に以下のような非凸二次最適化問題 (NQP) を考える。ここで  $c \in R^n, a_i \in R^n, q_j \in R^n$  は定数ベクトル,  $b_i \in R^1, \gamma_j \in R^1$  は定数,  $Q \in R^{n \times n}$  は  $n \times n$  の定数行列である。

$$(NQP) \quad \begin{cases} \max & c^T x \\ \text{subj.to} & x \in F \end{cases} \quad (4)$$

$$F \equiv \left\{ x \in R^n : \begin{cases} a_i^T x + b_i \leq 0 & (i = 1, \dots, m_1) \\ x^T Q_j x + q_j^T x + \gamma_j \leq 0 & (j = 1, \dots, m_2) \end{cases} \right\}$$

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  $Q$  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 ( $c^T x$ ) を仮定することができる。  $F$  上での  $c^T x$  の最大値と  $F$  の凸包上での  $c^T x$  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 800MHz 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 LC80-144 ( $n = 81, m_1 = 120,$

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

CPU(台)	LC80-144		Frac50-20	
	実行時間 (秒)	比率	実行時間 (秒)	比率
1	33125	1.00	289259	1.00
2	16473	2.01	145980	1.98
4	8238	4.02	72343	3.99
8	4145	7.99	36272	7.97
16	2099	15.78	18595	15.56
32	1118	29.62	9424	30.69
64	624	53.08	4822	60.00
128	361	91.76	3200	90.39

$m_2 = 1$ ) と Frac50-20 ( $n = 51, m_1 = 21, m_2 = 1$ ) 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 LC80-144 では約 92 倍, Frac50-20 では約 90 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 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 約 90 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産業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及び合田先生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. Tunçel, “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”, The Institute of Statistical Mathematics Cooperative Research Report, Vol 135, pp. 238–258, 2000.
- [6] L. Vandenberghe and S. Boyd: Semidefinite programming; *SIAM Review*, Vol. 38, pp. 49–95 (1996)
- [7] 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)