日本オペレーションズ・リサーチ学会
2004年春季研究発表会

半正定値計画に対する行列補完型主双対内点法の並列化

01405430　東京工業大学　*中田 和秀　NAKATA Kazuhide
02701900　東京工業大学　山下 真　YAMASHITA Makoto
01507230　東京電機大学　藤沢 克樹　FUJISAWA Katsuki
01103520　東京工業大学　小島 政和　KOJIMA Masakazu

1　はじめに

本発表では、大規模な半正定値計画 (以下、SDP と略) を解くため、行列補完を利用して解くタイプの主双対内点法を並列に計算する方法を提案する。この方法は、[1, 2] で提案された行列補完型主双対内点法と、[5] で提案された並列主双対内点法を組み合わせた方法として捉えることができる。結果として、提案する方法に対し両者の長所を備え合わせ、互いの短所を補わせることに成功した。東京工業大学 松岡研究室の PC クラスタ Presto III 上での数値実験により、従来の解法と比べ非常に少ない計算時間とメモリ使用量しか必要としないことが確認できた。

2　半正定値計画とは

$\mathcal{S}^n$ を $n \times n$ の対称行列の空間とする。$A_p \in \mathcal{S}^n$ $(p = 0, 1, \ldots, m)$ と $b \in \mathbb{R}^m$ が与えられたとき、$X, Y \in \mathcal{S}^n$ と $z \in \mathbb{R}^m$ を変数とする次のような最適化問題を半正定値計画 (SDP) と呼ぶ。

主問題：

$$\begin{array}{ll} \text{最小化} & A_0 \circ X \\ \text{制約条件} & A_p \circ X = b_p \ (p = 1, 2, \ldots, m), \\ & X \in \mathcal{S}^n_+. \end{array}$$

双対問題：

$$\begin{array}{ll} \text{最大化} & \sum_{p=1}^m b_p z_p \\ \text{制約条件} & \sum_{p=1}^m A_p z_p + Y = A_0, \\ & Y \in \mathcal{S}^n_+. \end{array}$$

ただし $U, V \in \mathcal{S}^n$ に対し $U \circ V \equiv \sum_{i=1}^n \sum_{j=1}^n U_{ij} V_{ij}$ と定義する。また、$U \in \mathcal{S}^n_+$ は $U \in \mathcal{S}^n$ が半正定値であることを意味する。

3　既存の研究

SDP に対する主双対内点法 (SDPA)

SDP は主双対内点法により多項式時間で解けることが知られている。筆者らはこの主双対内点法を実装し、SDPA[4] というソフトウェアを開発した。SDPA は中程度の大きさの SDP を効率よく解くことができるが、大規模な SDP を解くことは困難である。SDPA の中で多くの計算時間を必要とする個所として次の 4 つの部分がある。

- C1: 線形方程式系の係数行列 $B \in \mathcal{S}^m$ の計算
- C2: 係数行列 $B \in \mathcal{S}^m$ の Cholesky 分解
- C3: 主問題の探索方向 dX の計算
- C4: 上記以外の $n \times n$ の密行列を扱う演算

また、SDPA の実行において、メモリを多く消費する部分として次の 2 つが挙げられる。

- M1: 線形方程式系の係数行列 $B \in \mathcal{S}^m$ の保持
- M2: $n \times n$ の密行列の保持

次ページにある表 1 の 1 列目は、ある SDP を SDPA により解いたときの、各部分でかかった計算時間とメモリ使用量を示している。計算時間の大部分が C1, C2, C3, C4、メモリ使用量の大部分が M1, M2 で消費されていることが確認できる。

行列補完理論の導入 (SDPA-C)

実務上、データ行列 $A_p (p = 0, 1, \ldots, m)$ が疎行列であることは多い。この場合、双対問題の行列変数 Y は疎になるものの、主問題の行列変数 X は一般に密となる。[1, 2] では、行列補完理論を導入することにより、主問題の特定の行列変数 $\widehat{X}$ が疎行列 M を使い $\widehat{X} = M^{-T} M^{-1}$ と疎分解できることを示している。よって、密行列 $\widehat{X}$ の代わりに疎行列 M を保持すればよい。このような疎性を利用することにより、C3 と C4 を高速に計算し、M2 を省メモリで保持する方法が提案されている。しかし、主問題の行列変数 X を直接保持しないため、C1 の計算時間が増大するという新たな問題が生じる。この方法を実装したソフトウェア SDPA-C の実験結果は表 1 の 2 列目である。

並列化 (SDPARA)

[5] では SDPA の並列化について述べられている。そこでは、PC 間の通信に MPI を用い **C1** を効率よく並列計算する方法が提案されている。また、数値計算ライブラリ ScaLapack の並列 Cholesky 分解を用いて **C2** を並列計算する。このとき、係数行列 $\boldsymbol{B}$ は各 PC で分割して保持するため、**M1** に必要なメモリ量を減らすことができる。この方法を実装したソフトウェア SDPARA の実験結果は表 1 の 3 列目である。なお、実験では 64 台の PC を使い並列計算を行った。

4 提案する方法 (SDPARA-C)

本発表では、行列補完型主双対内点法である SDPA-C の並列化を提案する。以下で、各部分で行う計算法を簡単に述べる。**C1** は SDPARA で採用した並列化のアプローチを基に、ロードバランスを均等に保つような工夫を加える。**C2** は SDPARA と同様に ScaLapack の並列 Cholesky 分解を用いる。**C3** は SDPA-C で提案された計算法を並列化する。このとき、導出された疎性をうまく利用しデータの通信量を削減する。**C4** は SDPA-C と同様の計算を行う。メモリに関しては、SDPA-C と同様に行列補完理論による疎性を使うことと、SDPARA と同様に係数行列 $\boldsymbol{B}$ を各 PC に分割することにより、**M1** と **M2** 共に少ないメモリ量で保持することができる。

その結果、提案した方法は **C3**,**C4**,**M2** において SDPA-C のメリット、**C1**,**C2**,**M1** において SDPARA のメリットを受け継ぐことになる。この方法を実装したソフトウェア SDPARA-C の実験結果は表 1 の 4 列目である。SDPARA と同様に 64 台の PC を用いた。

表 1: 計算時間とメモリ使用量

	SDPA	SDPA-C	SDPA RA	SDPA RA-C
C1	173 秒	1269 秒	12 秒	20 秒
C2	77 秒	107 秒	6 秒	9 秒
C3	70 秒	38 秒	70 秒	3 秒
C4	129 秒	3 秒	128 秒	3 秒
全体時間	459 秒	1420 秒	228 秒	36 秒
M1	59MB	59MB	2MB	2MB
M2	237MB	9MB	237MB	9MB
全メモリ	311MB	71MB	268MB	44MB

5 スケーラビリティ

図 1 は [3] からの抜粋である。これは並列度を変えながら 10 個の SDP を SDPARA-C で解き、その計算時間をグラフにしたものである。ここで、横軸は PC の台数、縦軸は計算時間の対数である。SDPARA-C は非常に高いスケーラビリティを持っていることが確認できる。

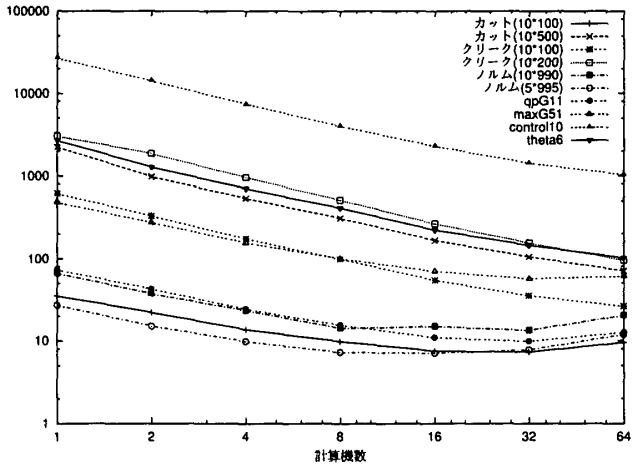

図 1: SDPARA-C のスケーラビリティ

参考文献

[1] M. Fukuda, M. Kojima, K. Murota and K. Nakata, Exploiting sparsity in semidefinite programming via matrix completion I: General framework, *SIAM Journal on Optimization* 11 (2000) 647–674.

[2] K. Nakata, K. Fujisawa, M. Fukuda, M. Kojima and K. Murota, Exploiting sparsity in semidefinite programming via matrix completion II: Implementation and numerical results, *Mathematical Programming*, Series B, 95 (2003) 303–327.

[3] K. Nakata, M. Yamashita, K. Fujisawa, and M. Kojima, A Parallel Solution Method for Semidefinite Programming Problems using Matrix Completion, テクニカルレポート B-387, 東京工業大学 数理・計算科学専攻 (2003).

[4] M. Yamashita, K. Fujisawa and M. Kojima, Implementation and Evaluation of SDPA6.0(SemiDefinite Programming Algorithm 6.0), *Optimization Methods and Software* 18 (2003) 491–505.

[5] M. Yamashita, K. Fujisawa and M. Kojima, SDPARA: SemiDefinte Programming Algorithm PARAllel version, *Parallel Computing* 29 (2003) 1053–1067.