Research Reports on Mathematical and Computing Sciences
Series B : Operations Research

Department of Mathematical and Computing Sciences
Tokyo Institute of Technology
2-12-1 Oh-Okayama, Meguro-ku, Tokyo 152, Japan

Using the Conjugate Gradient Method in Interior-Points Methods for Semidefinite Programs (in Japanese)

Kazuhide Nakata[†], Katsuki Fujisawa[†] and Masakazu Kojima[†]

B-339 April 1998

Abstract. In recent years, Semidefinite Programming (SDP) has been intensively studied from both theoretical and practical aspects in various fields including interior-point methods, combinatorial optimization, and the control and systems theory. Besides the SDPA (Semidefinite Programming Algorithm) which we developed to solve SDPs using the primal-dual interior-point method, many computer programs for SDPs are now available through the Internet. Most of those programs utilize direct methods such as the Cholesky factorization or the LU factorization to solve a system of linear equations which arises at each iteration to generate a search direction. The system of equations to be solved at each iteration is full dense in general. Hence, as the size of the system of linear equations gets larger, the use of such direct methods becomes more difficult; even storing its coefficient matrix in the computer memory becomes impossible. To overcome this difficulty, we incorporate the conjugate graduate method, which is a popular iterative method for solving systems of linear equations, and various improvements into the SDPA. In particular, the resultant modified method, which we call the SDPA-CG, does not store the coefficient matrix of the system of equations to be solved at each iteration. This feature of the SDPA-CG makes it possible to handle much larger size of SDPs than the ones that the SDPA can do. Some numerical experiments show that when the SDPA-CG is applied to sparse problems having a large number of inequality constraints, the conjugate gradient method works very efficiently in earlier stages of the SDPA-CG until the SDPA-CG generates an approximate solution with a low accuracy. In particular, the SDPA-CG solves an SDP relaxation of the maximum clique problem which has 500 × 500 variable matrices and more than 110,000 inequality constraints.

Key words. Interior-Point Methods, Semidefinite Programming, Conjugate Gradient Method

[†] Department of Mathematical and Computing Sciences, Tokyo Institute of Technology, 2-12-1 Oh-Okayama, Meguro-ku, Tokyo 152, Japan.

B-339 半正定値計画問題に対する主双対内点法における共役勾配法の実装

中田 和秀[†]，藤沢 克樹[†]，小島 政和[†]

1998 年４月

概要． 半正定値計画（Semidefinite Programming, SDP と略）問題を主双対内点法により解くソフトウエアは、本研究が基づいた SDPA を含め幾つか公開されている。これらのソフトウエアでは、主双対内点法の各反復で解く線形方程式系に、コレスキー分解や LU 分解等を利用している。しかし、これらの直接解法では、計算時間やメモリの制約から、組合せ最適化問題への応用等に現れる大規模な問題を解くことが困難である。本研究では、この困難を克服するために本 SDPA の各反復で解く線形方程式系に反復解法である共役勾配法を用い、さらに、計算効率を高める種々の工夫を組み入れたソフトウエア SDPA-CG の開発を行った。記憶容量の面でも、SDPA-CG では線形方程式系の係数行列を保持せずに共役勾配法の計算を行うことにより、SDPA に比べメモリ使用量を大幅に減らすことができ、より大規模な問題を扱うことが可能となった。論文の後半では、最大クリーク問題の SDP 緩和、グラフ分割問題の SDP 緩和、ノルム最小化問題に対する SDPA-CG の数値実験結果を報告する。この実験を通して、組合せ最適化問題の SDP 緩和のように、要求される精度が小数点１～２桁程度の低い場合には SDPA-CG は有効に働くことを検証した、特に、超大規模な最大クリーク問題の SDP 緩和（行列変数サイズ $= 500$、制約条件数 $= 112{,}253$）を効率良く解くことができた。

Key words 内点法, 半正定値計画, 共役勾配法

† 郵便番号１５２ 目黒区大岡山２−１２−１
東京工業大学 情報理工学研究科 数理・計算科学専攻

1　はじめに

近年、半正定値計画 (Semidefinite Programming, SDP と略）問題に対して、その解法である主双対内点法に関わる理論的な解析、組合せ最適化問題への応用等の様々な研究が行われている。また、我々が で公開した SDPA (SemiDefinite Programming Algorithm、Fujisawa 他 (1996)）を始め、SDP を主双対内点法により解くソフトウェアが幾つか開発され、それをもとにした数値実験等の研究が盛んに行われている。Alizadeh 他 (1997)、Borchers(1997)、Potra 他 (1997)、Todd 他 (1996) 等参照。これらのソフトウェアでは、主双対内点法の各反復で解く線形方程式系に、コレスキー分解、修正コレスキー分解、LU 分解等を利用している。しかしこれらの直接法では、計算時間やメモリ使用量の制約から、組合せ最適化問題への応用等に現れる大規模な問題を解くことが困難である。そこで本研究では、この困難を克服するため、SDPA の各反復で解く線形方程式系に共役勾配法 (Conjugate gradient method) を適用したソフトウェア SDPA-CG の開発を行った。

共役勾配法は、係数行列が対称でかつ正定値である線形方程式系の反復解法であり、通常、係数行列が疎である線形方程式系の解法として利用されている。主双対内点法で SDP を解く際に現れる線形方程式系の係数行列は、一般に疎ではないが、この係数行列を保持せずに計算を行うことにより、SDPA に比べメモリ使用量を大幅に減らすことができる。一方、共役勾配法において計算時間は反復回数に大きく依存しており、より高い精度の近似解を生成するにつれて、この反復回数は増大することが予想される。本論文では SDPA-CG が生成する近似解の収束およびそれに要する計算時間等に関する実験的解析を報告する。また、共役勾配法の終了条件や、実行可能性の補正、効率的な計算方法等についても提案を行う。さらに、論文の後半では、大規模なグラフの最大クリーク問題の SDP 緩和、グラフ分割問題の SDP 緩和、および、ノルム最小化問題に対する SDPA-CG の有効性について調べる。

第 2 節では SDP の定義とその解法である主双対内点法について述べる。第 3 節では SDPA の実験結果等を基に疎で大規模な SDP を既存のソフトウェア SDP で解いたときの問題点を明らかにする。第 4 節では線形方程式系を解く反復解法である共役勾配法について簡単に説明する。第 5 節では共役勾配法を主双対内点法の各反復で解く線形方程式系に適用するにあたって、効率のよい計算方法を提案する。第 6 節では SDPA-CG の性能評価を行う。第 7 節では、大規模なグラフの最大クリーク問題の SDP 緩和、グラフ分割問題の SDP 緩和、および、ノルム最小化問題に対する SDPA-CG の数値実験結果とそれに対する考察を述べる。第 8 節ではまとめを述べる。

2　半正定値計画問題 (Semidefinite Programming)

$\Re^{n\times n}$ 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、$\mathrm{Tr}\ \boldsymbol{X}^T\boldsymbol{Z}$ ($\boldsymbol{X}^T\boldsymbol{Z}$のトレース）を表す。$\boldsymbol{X} \succ \boldsymbol{O}$ は$\boldsymbol{X} \in \mathcal{S}^n$ が正定値、つまりゼロでない任意の $\boldsymbol{u} \in \Re^n$ に対し$\boldsymbol{u}^T\boldsymbol{X}\boldsymbol{u} > 0$ であることを示す。$\boldsymbol{X} \succeq \boldsymbol{O}$ 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 $\boldsymbol{u}^T\boldsymbol{X}\boldsymbol{u} \geq 0$

であることを示す。

$\boldsymbol{C} \in \mathcal{S}^n$, $\boldsymbol{A}_i \in \mathcal{S}^n$ $(1 \le i \le m)$, $b_i \in \Re$ $(1 \le i \le m)$ とする。このとき、半正定値計画問題 (Semidefinite Programming–SDP) の主問題と双対問題は以下のように与えられる。

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

論文を通して、$\boldsymbol{A}_i \in \mathcal{S}^n (1 \le i \le m)$ が線形独立であることを仮定する。$\boldsymbol{X}$ が $\boldsymbol{X} \succ \boldsymbol{O}$ を満たす主問題の実行可能解であり、$(\boldsymbol{y}, \boldsymbol{Z})$ が $\boldsymbol{Z} \succ \boldsymbol{O}$を満たす双対問題の実行可能解であるとき、$(\boldsymbol{X}, \boldsymbol{y}, \boldsymbol{Z})$ を SDP (1) の内点実行可能解と呼ぶ。

以下では、SDP の主双対内点法の枠組みと探索方向の計算方法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。

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

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 $\mathcal{P}$ は以下のように定義される。

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

ただし、$\boldsymbol{I}$ は単位行列を表す。任意の μ 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。

SDP の主双対内点法では、探索方向パラメータ μ 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。

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

一般に、$\boldsymbol{X}\boldsymbol{Z}$ は対称行列でなく、系 (4) の左辺は $\mathcal{S}^n \times \Re^m \times \mathcal{S}^n$ から $\Re^m \times \mathcal{S}^n \times \Re^{n \times n}$ への写像となるため、直接 Newton 法を適用できない。このため、現在までに様々な探索方向

$(d\boldsymbol{X}, d\boldsymbol{y}, d\boldsymbol{Z})$ が提案されてきた。本論文では、まず MZ 方向族（Monteiro 他 (1996)）について述べ、その特殊な場合として HRVW/KSH/M 方向（Helmberg 他 (1996), Kojima 他 (1997), Monteiro (1997)）と NT 方向 (Nesterov 他 (1994,1995)) について述べる。SDPA-CG に実装した HRVW/KSH/M 方向と NT 方向の場合について探索方向 $(d\boldsymbol{X}, d\boldsymbol{y}, d\boldsymbol{Z})$ の具体的な計算方法についても述べる。

MZ 方向族：任意の正則行列 $\boldsymbol{P} \in \Re^{n\times n}$ に対し、$\boldsymbol{H}_p : \Re^{n\times n} \to \mathcal{S}^n$ を以下のように定義する。

$$\boldsymbol{H}_P(\boldsymbol{A}) = \frac{\boldsymbol{P}\boldsymbol{A}\boldsymbol{P}^{-1} + (\boldsymbol{P}\boldsymbol{A}\boldsymbol{P}^{-1})^T}{2}.$$

行列 $\boldsymbol{P}$ はスケーリング行列の役目をし、$\boldsymbol{H}_P(\boldsymbol{A})$ はスケーリングされた行列 $\boldsymbol{P}^{-1}\boldsymbol{A}\boldsymbol{P}$ の対称化に相当する。このとき、$\boldsymbol{H}_P(\boldsymbol{A})$ は $\boldsymbol{A}$ に対する線形操作であるこを注意しておく。

系 (4) の $\boldsymbol{X}\boldsymbol{Z} = \mu\boldsymbol{I}$ を $\boldsymbol{H}_P(\boldsymbol{X}\boldsymbol{Z}) = \boldsymbol{H}_P(\mu\boldsymbol{I})$ に置き換えると以下の新しい系を得る。

$$\left.\begin{array}{l} \boldsymbol{A}_i \bullet \boldsymbol{X} = b_i \ (1 \le i \le m), \\ \sum_{i=1}^{m} \boldsymbol{A}_i y_i + \boldsymbol{Z} = \boldsymbol{C}, \ \boldsymbol{H}_P(\boldsymbol{X}\boldsymbol{Z}) = \boldsymbol{H}_P(\mu\boldsymbol{I}). \end{array}\right\} \tag{5}$$

この系に Newton 法を適用し、探索方向 $(d\boldsymbol{X}, d\boldsymbol{y}, d\boldsymbol{Z})$ を以下の方程式系で与える。

$$\left.\begin{array}{l} \boldsymbol{A}_i \bullet d\boldsymbol{X} = p_i \ (1 \le i \le m), \\ \sum_{i=1}^{m} \boldsymbol{A}_i dy_i + d\boldsymbol{Z} = \boldsymbol{D}, \ \boldsymbol{H}_P(\boldsymbol{X}d\boldsymbol{Z} + d\boldsymbol{X}\boldsymbol{Z}) = \boldsymbol{H}_P(\boldsymbol{K}). \end{array}\right\} \tag{6}$$

ただし、$p_i = b_i - \boldsymbol{A}_i \bullet \boldsymbol{X}$, $\boldsymbol{D} = \boldsymbol{C} - \sum_{i=1}^m \boldsymbol{A}_i y_i - \boldsymbol{Z}$, $\boldsymbol{K} = \mu\boldsymbol{I} - \boldsymbol{X}\boldsymbol{Z}$ である。１つ目の等式が主問題の実行可能性、２つ目の等式が双対問題の実行可能性、３つ目の等式が相補性に相当している。この方程式系の一般的な解法についての詳細は Todd (1997) を参照されたい。

HRVW/KSH/M 方向：MZ 方向族の $\boldsymbol{P}$ として $\boldsymbol{Z}^{1/2}$ を採る場合に相当する。このとき、HRVW/KSH/M 方向 $(d\boldsymbol{X}, d\boldsymbol{y}, d\boldsymbol{Z})$ は以下のようにして求めることができる。

$$\left.\begin{array}{l} \boldsymbol{B}d\boldsymbol{y} = \boldsymbol{b}, \ d\boldsymbol{Z} = \boldsymbol{D} - \sum_{j=1}^{m} \boldsymbol{A}_j dy_j, \\ \widehat{d\boldsymbol{X}} = \boldsymbol{X}(\boldsymbol{X}^{-1}\boldsymbol{K} - d\boldsymbol{Z})\boldsymbol{Z}^{-1}, \ d\boldsymbol{X} = (\widehat{d\boldsymbol{X}} + \widehat{d\boldsymbol{X}}^T)/2. \end{array}\right\} \tag{7}$$

ただし、

$$\begin{array}{ll} B_{ij} = \boldsymbol{A}_i \bullet \boldsymbol{X}\boldsymbol{A}_j\boldsymbol{Z}^{-1} & (1 \le i \le m, \ 1 \le j \le m), \\ b_i = p_i - \boldsymbol{A}_i \bullet \boldsymbol{X}(\boldsymbol{X}^{-1}\boldsymbol{K} - \boldsymbol{D})\boldsymbol{Z}^{-1} & (1 \le i \le m). \end{array}$$

つまり、まず行列 $\boldsymbol{B}$ の各要素を計算し、その行列を係数行列とする線形方程式系を解き、$d\boldsymbol{y}$ を計算する。その後、$d\boldsymbol{Z}$、$\widehat{d\boldsymbol{X}}$、$d\boldsymbol{X}$ を順次計算する。

NT 方向 :行列 $\boldsymbol{W}$ を

$$\boldsymbol{W} = \boldsymbol{X}^{1/2}(\boldsymbol{X}^{1/2}\boldsymbol{Z}\boldsymbol{X}^{1/2})^{-1/2}\boldsymbol{X}^{1/2} = \boldsymbol{Z}^{-1/2}(\boldsymbol{Z}^{1/2}\boldsymbol{X}\boldsymbol{Z}^{1/2})^{1/2}\boldsymbol{Z}^{-1/2},$$

と定義する (MZ 方向族の $\boldsymbol{P}$ として $\boldsymbol{W}^{-1/2}$ を採る場合に相当)。このとき、NT 方向 $(d\boldsymbol{X}, d\boldsymbol{y}, d\boldsymbol{Z})$ は以下のようにして求めることができる。

$$\left.\begin{array}{l} \boldsymbol{B}d\boldsymbol{y} = \boldsymbol{b},\ d\boldsymbol{Z} = \boldsymbol{D} - \sum_{j=1}^{m} \boldsymbol{A}_j dy_j, \\ \widehat{d\boldsymbol{X}} = \boldsymbol{W}(\boldsymbol{X}^{-1}\boldsymbol{K} - d\boldsymbol{Z})\boldsymbol{W},\ d\boldsymbol{X} = (\widehat{d\boldsymbol{X}} + \widehat{d\boldsymbol{X}}^T)/2. \end{array}\right\} \tag{8}$$

ここで

$$\begin{array}{ll} B_{ij} = \boldsymbol{A}_i \bullet \boldsymbol{W}\boldsymbol{A}_j\boldsymbol{W} & (1 \le i \le m,\ 1 \le j \le m), \\ b_i = p_i - \boldsymbol{A}_i \bullet \boldsymbol{W}(\boldsymbol{X}^{-1}\boldsymbol{K} - \boldsymbol{D})\boldsymbol{W} & (1 \le i \le m). \end{array}$$

行列 $\boldsymbol{W}$ を求めた後は、NT 方向の計算方法は HRVW/KSH/M 方向の計算方法とほぼ同等であり、計算量やメモリ使用量はほとんど変わらないことを注意しておく。

この節の最後に SDP の一般的な主双対内点法の概要を以下に示す。

一般的な主双対内点法 :

Step 0: $\boldsymbol{X} \succ \boldsymbol{O}$ と $\boldsymbol{Z} \succ \boldsymbol{O}$ を満たすように初期点 $(\boldsymbol{X}, \boldsymbol{y}, \boldsymbol{Z})$ を決める。

Step 1: もし、現在の反復点 $(\boldsymbol{X}, \boldsymbol{y}, \boldsymbol{Z})$ が終了条件を満たしていたならば、反復を終了する。

Step 2: 探索方向パラメータ μ を定め、上述した方法で探索方向 $(d\boldsymbol{X}, d\boldsymbol{y}, d\boldsymbol{Z})$ を計算する。

Step 3: 主問題のステップ長 α_p と双対問題のステップ長 α_d を定め

$$\boldsymbol{X} = \boldsymbol{X} + \alpha_p d\boldsymbol{X} \succ \boldsymbol{O},\ (\boldsymbol{y}, \boldsymbol{Z}) = (\boldsymbol{y}, \boldsymbol{Z}) + \alpha_d(d\boldsymbol{y}, d\boldsymbol{Z}) \succ \boldsymbol{O}.$$

とする。

Step 4: Step 1 に戻る。

3 既存のソフトウェアの問題点

前節で述べた主双対内点法によって SDP を解くと、計算時間の大部分は式 (7) または (8) での探索方向の計算、その中でも特に線形方程式系 $\boldsymbol{B}d\boldsymbol{y} = \boldsymbol{b}$ の計算に使われていることが知られている。Fujisawa 他 (1997a) 参照。この線形方程式系 $\boldsymbol{B}d\boldsymbol{y} = \boldsymbol{b}$ の計算は、大きく分けて以下の 3 つの部分からなる。

(I) 係数行列 $\boldsymbol{B}$ の各要素の計算。

(II) 係数行列 $\boldsymbol{B}$ のコレスキー分解あるいは修正コレスキー分解。

(III) $d\boldsymbol{y}$ を求めるための代入計算。

n を SDP の変数対称行列 $\boldsymbol{X}$, $\boldsymbol{Z}$ の大きさ、m を変数ベクトル $\boldsymbol{y}$ の大きさとすると、これら (I), (II), (III) の計算には、通常、それぞれ $O(n^3m+n^2m^2)$ flops, $m^3/3+O(m^2)$ flops, $O(m^2)$ flops の計算と、$m \times m$ の係数行列 $\boldsymbol{B}$ の確保分のメモリが必要である。なお、線形方程式系 $\boldsymbol{B}d\boldsymbol{y}=\boldsymbol{b}$ の計算以外の部分は、$O(n^3+n^2m)$ flops の計算と、いくつかの $n \times n$ の行列の確保分のメモリが必要である。しかし、筆者らが開発した SDPA (SemiDefinite Programming Algorithm) では、Fujisawa 他 (1997a) で報告したように疎で大規模な問題に対し (I) の部分をより効率的な方法で計算を行っている。ここで、疎で大規模な問題とは、n または m が大きく、制約行列 $\boldsymbol{A}_i$ $(1 \leq i \leq m)$ が疎であることを意味し、組合せ最適化問題への応用によく現れる。以下、議論を簡単にするため、f を定数として、全ての制約行列$\boldsymbol{A}_i$ $(1 \leq i \leq m)$ が、高々 f 個の非ゼロ要素しか持たないことを仮定する。このとき SDPA では、$O(m^2f^2)$ flops で (I) の部分の計算を行っている。これにより SDPA では既存のソフトウェアに比べ、疎で大規模な問題を高速に解くことを可能にしたが、n に対して m が極めて大きくなる問題は解くことが困難であることが 指摘されている (Fujisawa 他 (1997b) 参照)。実際、SDPA は n を固定すると、計算量は $O(m^3)$, メモリ使用量は $O(m^2)$ となる。

次に SDPA での実験例を示す。表 1 はいくつかの疎で大規模な問題を SDPA によって解いたときの、計算時間とメモリ消費量を表している。この問題では、1 つを除いた制約行列の非ゼロ要素は 2 個であり、双対ギャップが 0.1 以下になるまで SDPA の反復を繰り返した。

表 1: 疎で大規模な問題を SDPA によって解いたときの計算時間と使用メモリ

n	100	100	100
m	1024	2029	3992
$\boldsymbol{B}d\boldsymbol{y}=\boldsymbol{b}$の計算時間	71.1 秒	575.6 秒	5140.6 秒
その他の部分の計算時間	4.6 秒	4.9 秒	5.3 秒
総計算時間	75.7 秒	580.5 秒	5146.9 秒
使用メモリ	15.1MB	41.3MB	142.0MB

この表より、計算時間の大部分は線形方程式系 $\boldsymbol{B}d\boldsymbol{y}=\boldsymbol{b}$ の計算に費やされていることが分かる。その部分の計算時間は、m が大きくなるにつれて非常に増大しており、それにともない総計算時間も増大している。また、使用メモリの大部分は係数行列 $\boldsymbol{B}$ の確保分に費やされている。そして、その部分の使用メモリは、m が大きくなるにつれて非常に増

大しており、それにともない使用メモリも増大している。つまり、m が大きくなるにつれ、計算時間やメモリ使用量の観点から解くことが非常に困難になっている。

SDPA 以外の既存のソフトウェア (Alizadeh 他 (1997)、Borchers(1997)、Potra 他 (1997)、Todd 他 (1996) 等) でも、線形方程式系 $\boldsymbol{B}d\boldsymbol{y} = \boldsymbol{b}$ の解法としてコレスキー分解や修正コレスキー分解等の直接法を使用しているため、同様の問題が起っている。通常、このような困難を解決するために、反復解法が利用されている。本研究では、SDPA を基礎として SDPA の各反復で解く線形方程式系 $\boldsymbol{B}d\boldsymbol{y} = \boldsymbol{b}$ に反復解法である共役勾配法を適用したソフトウェア SDPA-CG の開発を行った。

4 共役勾配法とは

共役勾配法は、Hestenes and Stiefel (1952) が提案した方法で、係数行列が対称でかつ正定値である線形方程式系に対する反復解法である。すでに述べたように、本研究では探索方向として HRVW/KSH/M 方向と NT 方向 を用いる。このとき、線形方程式系 $\boldsymbol{B}d\boldsymbol{y} = \boldsymbol{b}$ の係数行列 $\boldsymbol{B}$は対称で正定値であることが知られており、共役勾配法を適用することが可能である。なお、探索方向として他の方向 (AHO 方向等) を使用する場合、行列 $\boldsymbol{B}$ は対称で正定値となるとは限らないので、他の反復解法 (SOR 法など) を適用しなければならない。

4.1 共役勾配法

以下のアルゴリズムは、Goulb 他 (1983) で述べられている線形方程式系 $\boldsymbol{B}d\boldsymbol{y} = \boldsymbol{b}$ を解く標準的な共役勾配法である。

Algorithm 4.1. 共役勾配法

```
$k = 1;\ d\boldsymbol{y} = \boldsymbol{0};\ \boldsymbol{r} = \boldsymbol{b}$
while $(\boldsymbol{r} \neq \boldsymbol{0})$
    $t_1 = \boldsymbol{r}^T\boldsymbol{r}$
    if $(k = 1)$  $\boldsymbol{p} = \boldsymbol{r}$
    else
        $\beta = t_1/t_2$ and $\boldsymbol{p} = \boldsymbol{r} + \beta\boldsymbol{p}$
    end
    $\boldsymbol{q} = \boldsymbol{B}\boldsymbol{p}$
    $\alpha = t_1/(\boldsymbol{p}^T\boldsymbol{q})$
    $d\boldsymbol{y} = d\boldsymbol{y} + \alpha\boldsymbol{p}$
    $\boldsymbol{r} = \boldsymbol{r} - \alpha\boldsymbol{q}$
    $t_2 = t_1;\ k = k + 1$
end
```

この共役勾配法の 1 反復の計算は、大きさ $m \times m$ の行列 $\boldsymbol{B}$ と大きさ m のベクトル $\boldsymbol{p}$ の積の計算の部分と、$O(m)$ flops で計算できる部分からなる。よって、共役勾配法により大きさが m の線形方程式系を解く場合における計算量は、

$$(\text{積 } \boldsymbol{Bp} \text{ の計算量 } + O(m)\text{flops}) \times \text{共役勾配法の反復回数}$$

となる。また、使用するメモリ量は、大きさ m の 2 つのベクトル $d\boldsymbol{y}$、$\boldsymbol{b}$ の他に大きさ m の 3 つの作業用ベクトル $\boldsymbol{r}, \boldsymbol{p}, \boldsymbol{q}$、大きさ $m \times m$ の行列 $\boldsymbol{B}$ と大きさ m のベクトル $\boldsymbol{p}$ の積の計算に必要なメモリ量となる。

SDPA の場合、たとえ制約行列が疎であっても、係数行列 $\boldsymbol{B}$ が密になる。つまり、コレスキー分解を行うときのように、まず係数行列 $\boldsymbol{B}$ の各要素を計算しそれを保持すると、m が大きくなった場合、計算量やメモリ使用量の観点から非常に効率が悪い。この $\boldsymbol{Bp}$ の計算方法は重要であり、この部分をより効率的に計算する方法について次節で述べる。

4.2 前処理つき共役勾配法

共役勾配法のアルゴリズムにおいて $\boldsymbol{Bp}$ の計算方法を定めると、線形方程式系を共役勾配法で解くのに必要な計算量は、共役勾配法の反復回数に比例することが分かる。共役勾配法の収束を速め反復回数を減らす方法として、前処理つき共役勾配法が知られている。これは、線形方程式系 $\boldsymbol{B}d\boldsymbol{y} = \boldsymbol{b}$ を解く代わりに、ある正則行列である前処理行列 $\boldsymbol{M}$ に対して、$(\boldsymbol{M}^{-1}\boldsymbol{B}\boldsymbol{M}^{-T})(\boldsymbol{M}^T d\boldsymbol{y}) = (\boldsymbol{M}^{-1}\boldsymbol{b})$ という線形方程式を解くことに相当する。このとき行列 $\boldsymbol{M}^{-1}\boldsymbol{B}\boldsymbol{M}^{-T}$ の固有値を密集させれば、共役勾配法の収束を加速させることができる。詳しくは、Goulb 他 (1983) 等参照。

この前処理つき共役勾配法を用いたとき、1 反復の計算量は通常の共役勾配法に比べ、線形方程式系 $\boldsymbol{M}\boldsymbol{M}^T\boldsymbol{z} = \boldsymbol{r}$ を解く分だけ増え、また使用メモリは前処理行列 $\boldsymbol{M}$ と大きさ m の作業用ベクトル 1 本を保持する分だけ増える。よって前処理行列は、行列 $\boldsymbol{M}^{-1}\boldsymbol{B}\boldsymbol{M}^{-T}$ の固有値をより密集させ、線形方程式系 $\boldsymbol{M}\boldsymbol{M}^T\boldsymbol{z} = \boldsymbol{r}$ をより簡単に解くことができ、前処理行列 $\boldsymbol{M}$ の保持にメモリをあまり使わないなどの条件を満たしているものが望ましい。これまで様々な前処理行列が提案されているが、1 番簡単な前処理としては、対角スケーリングがある。これは、行列 $\boldsymbol{D}$ を行列 $\boldsymbol{B}$ の対角部分（つまり、$\mathrm{diag}(B_{11}, B_{22}, \cdots, B_{mm})$）としたとき、前処理行列 $\boldsymbol{M}$ として $\boldsymbol{D}^{1/2}$ を使用するものである。また、よく使われる前処理行列として $\boldsymbol{B}$ の不完全コレスキー分解がある。これは、行列 $\boldsymbol{B}$ のゼロの部分が前処理行列 $\boldsymbol{M}$ でもゼロとなるように、近似的にコレスキー分解をする方法である。もし行列 $\boldsymbol{B}$ が疎であれば、前処理行列 $\boldsymbol{M}$ も同程度に疎な三角行列となり、さらに行列 $\boldsymbol{M}^{-1}\boldsymbol{B}\boldsymbol{M}^{-T}$ が単位行列に近ければ、収束は良くなることが予想される。しかし、SDPA-CG の場合、係数行列 $\boldsymbol{B}$ は一般に密な行列であり、このまま不完全コレスキー行列を適用することはできず、何らかの改良が必要となる。

良い前処理行列とは、解こうとする問題の性質や使用できるメモリの量などに依存すると思われる。2 次割当て問題の緩和問題である SDP を解くとき、線形方程式系 $\boldsymbol{B}d\boldsymbol{y} = \boldsymbol{b}$ に共役勾配法を適用した研究において、前処理として Zhao et al. (1996) は対角スケーリング、Lin 他 (1997, 1998) は、改良を加えた不完全コレスキー分解を使用している。本研

究では、n に対して m が大きい一般の SDP をできるだけ少ないメモリで解くことを目的とするため、特に断らない限り原則として対角スケーリングを使用している。

5　共役勾配法の実装

線形方程式系の解法として反復解法である共役勾配法を用いると、近似解しか得られないため、いくつかの問題が生ずる。本節では、n に対し m が大きな ($m = O(n^2)$) 疎で大規模な問題を解くことを仮定し、そのような問題、および、それらを解決するための提案と効率の良い計算方法について述べる。

5.1　行列とベクトルの積の計算

共役勾配法の各反復で計算を行う大きさ $m \times m$ の行列 $\boldsymbol{B}$ と大きさ m のベクトル $\boldsymbol{p}$ の積は、計算時間やメモリ使用量の点で SDPA-CG の効率に大きな影響を与えることは既に述べた。まず、行列 $\boldsymbol{B}$ の各要素を計算し保持しておき、各反復で任意の i について $\sum_{j=1}^{m} B_{ij} p_j$ を計算するのでは、$m \times m$ 行列 $\boldsymbol{B}$ を保持するためのメモリと 1 反復あたり $O(m^2)$ flops の計算量が必要であり、m が大きな疎で大規模な問題を解く場合に効率が悪い。

まず最初に、HRVW/KSH/M 方向の場合の計算方法について述べる。積 $\boldsymbol{Bp}$ で計算される大きさ m のベクトルを $\boldsymbol{q}$ とすると、任意の i に対して、

$$
\begin{aligned}
q_i &= \sum_{j=1}^{m} B_{ij} p_j = \sum_{j=1}^{m} \boldsymbol{A}_i \bullet \boldsymbol{X} \boldsymbol{A}_j \boldsymbol{Z}^{-1} p_j \\
&= \sum_{j=1}^{m} \mathbf{Tr}(\boldsymbol{A}_i \boldsymbol{X} \boldsymbol{A}_j \boldsymbol{Z}^{-1}) p_j \\
&= \mathbf{Tr}(\boldsymbol{A}_i \boldsymbol{X} \sum_{j=1}^{m} \boldsymbol{A}_j p_j \boldsymbol{Z}^{-1}) \\
&= \boldsymbol{A}_i \bullet \boldsymbol{X} \sum_{j=1}^{m} \boldsymbol{A}_j p_j \boldsymbol{Z}^{-1}
\end{aligned}
$$

が成り立つ。よって、以下のような手順で積 $\boldsymbol{Bp}$ の計算を行うことができる。

$$
\boldsymbol{H}_1 = \sum_{j=1}^{m} p_j \boldsymbol{A}_j,\ \boldsymbol{H}_2 = \boldsymbol{X} \boldsymbol{H}_1 \boldsymbol{Z}^{-1},\ q_i = \boldsymbol{A}_i \bullet \boldsymbol{H}_2\ (1 \leq i \leq m) \tag{9}
$$

この計算方法では、計算量としてほぼ大きさ $n \times n$ の行列同士の掛け算 2 回分 ($4n^3$flops) と見積もることができる。また、$m \times m$ の行列 $\boldsymbol{B}$ を保持せずにすみ、大きさ $n \times n$ の作業用行列を数個分の保持で HRVW/KSH/M 方向の計算が可能になる。

同様にして NT 方向の場合は、以下の計算方法で求めることができる。

$$
\boldsymbol{H}_1 = \sum_{j=1}^{m} p_j \boldsymbol{A}_j,\ \boldsymbol{H}_2 = \boldsymbol{W} \boldsymbol{H}_1 \boldsymbol{W},\ q_i = \boldsymbol{A}_i \bullet \boldsymbol{H}_2\ (1 \leq i \leq m) \tag{10}
$$

計算量と使用メモリは、HRVW/KSH/M 方向のときと変わらない。

5.2 主問題の実行可能性の補正

線形方程式系 $\boldsymbol{B}d\boldsymbol{y} = \boldsymbol{b}$ を共役勾配法で解き得られた近似解を $\tilde{d\boldsymbol{y}}$ とする。この $\tilde{d\boldsymbol{y}}$ から、式 (7) または式 (8) により、近似探索方向 $(\tilde{d\boldsymbol{X}}, \tilde{d\boldsymbol{y}}, \tilde{d\boldsymbol{Z}})$ を計算することができる。しかし、この近似探索方向は、(6) の全ての等式は満たさない。

一般に、近似探索方向 $(\tilde{d\boldsymbol{X}}, \tilde{d\boldsymbol{y}}, \tilde{d\boldsymbol{Z}})$ は、以下のような関係を満たす。

$$\left.\begin{array}{ll} \boldsymbol{A}_i \bullet d\boldsymbol{X} \neq p_i & (1 \le i \le m), \\ \displaystyle\sum_{i=1}^{m} \boldsymbol{A}_i dy_i + d\boldsymbol{Z} = \boldsymbol{D}, & \\ \boldsymbol{H}_P(\boldsymbol{X}d\boldsymbol{Z} + d\boldsymbol{X}\boldsymbol{Z}) = \boldsymbol{H}_P(\boldsymbol{K}). & \end{array}\right\} \tag{11}$$

実際、式 (7) または式 (8) により計算された近似探索方向 $(\tilde{d\boldsymbol{X}}, \tilde{d\boldsymbol{y}}, \tilde{d\boldsymbol{Z}})$ において、任意の i に対し、

$$p_i - \boldsymbol{A}_i \bullet \tilde{d\boldsymbol{X}} \;=\; b_i - \sum_{j=1}^{m} B_{ij}\tilde{dy}_j,$$

という関係があり、主問題の実行可能性における誤差 $p_i - \boldsymbol{A}_i \bullet \tilde{d\boldsymbol{X}}$ は、線形方程式系の誤差 $b_i - \sum_{j=1}^{m} B_{ij}\tilde{dy}_j$ と一致する。なお線形方程式系に共役勾配法を用いるとき、線形方程式系の誤差ベクトル $\boldsymbol{b} - \boldsymbol{B}\tilde{d\boldsymbol{y}}$ は共役勾配法のアルゴリズム中の $\boldsymbol{r}$ に等しく、線形方程式系の近似解 $\tilde{d\boldsymbol{y}}$ から派生する主問題の実行可能性に対する誤差 $p_i - \boldsymbol{A}_i \bullet \tilde{d\boldsymbol{X}}$ は改めて計算する必要がない。この事実は後に述べる計算方法において使われる。

しかし、これでは近似探索方向 $(\tilde{d\boldsymbol{X}}, \tilde{d\boldsymbol{y}}, \tilde{d\boldsymbol{Z}})$ が、主問題の実行可能性を満たしておらず、主双対内点法では都合が悪い。よって、以下の最小二乗問題の最適解 $\tilde{d\boldsymbol{X}}^*$ を$\tilde{d\boldsymbol{X}}$の代わりに使用する。

$$\text{minimize } \|d\boldsymbol{X} - \tilde{d\boldsymbol{X}}\|_F \;\; \text{subject to } \boldsymbol{A}_i \bullet d\boldsymbol{X} = p_i \; (1 \le i \le m). \tag{12}$$

これは、行列空間における線形部分空間への射影に相当しており、最適解$\tilde{d\boldsymbol{X}}^*$は以下の計算方法によって得ることができる。

$$\boldsymbol{G}\boldsymbol{w} = \boldsymbol{r}, \; \boldsymbol{J} = \sum_{j=1}^{m} \boldsymbol{A}_j w_j, \; \tilde{d\boldsymbol{X}}^* = \tilde{d\boldsymbol{X}} + \boldsymbol{J}. \tag{13}$$

ここで、 計算方法中の$\boldsymbol{r}$は共役勾配法の計算に使用する$\boldsymbol{r}$であり、また、$m \times m$ 対称行列 $\boldsymbol{G}$ の各要素 G_{ij} は

$$G_{ij} \;=\; \boldsymbol{A}_i \bullet \boldsymbol{A}_j \; (1 \le i \le m, \; 1 \le j \le m),$$

で与えられる。ここでは、まず線形方程式系 $\boldsymbol{G}\boldsymbol{w} = \boldsymbol{r}$ を解く。これは線形方程式系 $\boldsymbol{B}d\boldsymbol{y} = \boldsymbol{b}$ と同じ大きさの問題であり、線形方程式系 $\boldsymbol{B}d\boldsymbol{y} = \boldsymbol{b}$ と同様に解くことが困難であるように思われるが、以下の様な違いがある。まず、行列 $\boldsymbol{G}$ は主双対内点法の各反復において不変のため、SDP を解く上で１度だけコレスキー分解をすればよい。また仮定をしているように $\boldsymbol{A}_i$ が十分に疎な行列ならば、$\boldsymbol{G}$ も疎になることが予想される。よって多くの場合、

コレスキー分解にかかる計算量、コレスキー分解をした行列の保持、コレスキー分解をした行列を使って線形方程式系 $\boldsymbol{G}\boldsymbol{w} = \boldsymbol{r}$ を解く計算量等は小さいことが期待できる。

以上で、計算された新しい近似探索方向 $(\widetilde{d\boldsymbol{X}}^*, \widetilde{d\boldsymbol{y}}, \widetilde{d\boldsymbol{Z}})$ は、以下の関係を満たす。

$$\left.\begin{array}{ll} \boldsymbol{A}_i \bullet d\boldsymbol{X} = p_i & (1 \leq i \leq m), \\ \displaystyle\sum_{i=1}^{m} \boldsymbol{A}_i dy_i + d\boldsymbol{Z} = \boldsymbol{D}, & \\ \boldsymbol{H}_P(\boldsymbol{X}d\boldsymbol{Z} + d\boldsymbol{X}\boldsymbol{Z}) \neq \boldsymbol{H}_P(\boldsymbol{K}). & \end{array}\right\} \tag{14}$$

つまり、主問題の実行可能性と双対問題の実行可能性が満たされている。相補性条件は満たされていないが、この条件は線形近似によって導かれた条件であるため、正確に満たされる必要はない。相補性条件における誤差の許容範囲は 5.3 節の中で述べる。

5.3 共役勾配法の終了条件

共役勾配法を線形方程式系 $\boldsymbol{B}d\boldsymbol{y} = \boldsymbol{b}$ に適用すると、理論的には m （線形方程式系の大きさ）回の反復で正確な解 $d\boldsymbol{y}$ が得られることが知られている。しかしながら、実際には誤差の影響を受けやすく有限回で正確な解 $d\boldsymbol{y}$ に収束する保証は無い。5.2 節で述べたように、近似探索方向は相補性条件を厳密に満たす必要は無く、SDPA-CG の収束を保証する精度が得られれば共役勾配法の反復を打ち切ってよい。この節では共役勾配法の終了条件について議論する。

Kojima 他 (1997) では、理論的な大域的収束の枠組みの中から *admissible direction* と呼ばれる以下の式を満たす近似探索方向を提案した。

$$\|\boldsymbol{H}_P(\boldsymbol{X}d\boldsymbol{Z} + d\boldsymbol{X}\boldsymbol{Z}) - \boldsymbol{H}_P(\mu\boldsymbol{I} - \boldsymbol{X}\boldsymbol{Z})\|_F \leq \kappa\|\boldsymbol{H}_P(\mu\boldsymbol{I} - \boldsymbol{X}\boldsymbol{Z})\|_F \quad . \tag{15}$$

ここで、$\kappa \in [0, 1)$ は近似精度を表すパラメータである。本研究では、計算された近似探索方向がこの admissible direction となるとき、共役勾配法の反復を終了することにした。そのためには、共役勾配法の各反復でこの条件 (15) の判定を行わなければならない。条件 (15) の左辺は、近似探索方向の $d\boldsymbol{Z}$ と $d\boldsymbol{Z}$ を含んでいる。 したがって、この条件を文字通りチェックしようとすると、共役勾配法によって求めた線形方程式系 $\boldsymbol{B}d\boldsymbol{y} = \boldsymbol{b}$ の近似解 $\widetilde{d\boldsymbol{y}}$ から、まず式 (7) または 式 (8) により $\widetilde{d\boldsymbol{X}}, \widetilde{d\boldsymbol{Z}}$ を計算し、次に式 (13) により主問題の実行可能性の補正を行い $\widetilde{d\boldsymbol{X}}^*$ を計算することによって、近似探索方向 $(\widetilde{d\boldsymbol{X}}^*, \widetilde{d\boldsymbol{y}}, \widetilde{d\boldsymbol{Z}})$ を導かなければならない。共役勾配法の 1 反復にこれだけの計算を行うと、全体の計算時間を大幅に増大させることになる。以下では、$\widetilde{d\boldsymbol{X}}^*$ および $\widetilde{d\boldsymbol{Z}}$ を計算せず、直接に条件 (15) の左辺 $\|\boldsymbol{H}_P(\boldsymbol{X}d\boldsymbol{Z} + d\boldsymbol{X}\boldsymbol{Z}) - \boldsymbol{H}_P(\mu\boldsymbol{I} - \boldsymbol{X}\boldsymbol{Z})\|_F$ を評価する方法について述べる。

近似探索方向は $(\widetilde{d\boldsymbol{X}}, \widetilde{d\boldsymbol{y}}, \widetilde{d\boldsymbol{Z}})$ は式 (11) を満たす。したがって、

$$\boldsymbol{H}_P(\boldsymbol{X}\widetilde{d\boldsymbol{Z}} + \widetilde{d\boldsymbol{X}}\boldsymbol{Z}) = \boldsymbol{H}_P(\mu\boldsymbol{I} - \boldsymbol{X}\boldsymbol{Z}).$$

が導かれる。よって、条件 (15) の左辺は

$$\begin{aligned}
&\|\boldsymbol{H}_P(\boldsymbol{X}d\tilde{\boldsymbol{Z}} + d\tilde{\boldsymbol{X}}^*\boldsymbol{Z}) - \boldsymbol{H}_P(\mu\boldsymbol{I} - \boldsymbol{X}\boldsymbol{Z})\|_F \\
&\quad = \|\boldsymbol{H}_P(\boldsymbol{X}d\tilde{\boldsymbol{Z}} + (d\tilde{\boldsymbol{X}} - \boldsymbol{J})\boldsymbol{Z}) - \boldsymbol{H}_P(\mu\boldsymbol{I} - \boldsymbol{X}\boldsymbol{Z})\|_F \\
&\quad = \|\boldsymbol{H}_P(\boldsymbol{X}d\tilde{\boldsymbol{Z}} + d\tilde{\boldsymbol{X}}\boldsymbol{Z}) - \boldsymbol{H}_P(\mu\boldsymbol{I} - \boldsymbol{X}\boldsymbol{Z}) - H_p(\boldsymbol{J}\boldsymbol{Z})\|_F \\
&\quad = \|\boldsymbol{H}_P(\boldsymbol{J}\boldsymbol{Z})\|_F
\end{aligned}$$

となる。つまり、5.2 節で述べた実行可能性の補正をする計算方法 (13) で使用する行列$\boldsymbol{J}$を使用することにより、$\|\boldsymbol{H}_P(\boldsymbol{X}d\tilde{\boldsymbol{Z}} + d\tilde{\boldsymbol{X}}^*\boldsymbol{Z}) - \boldsymbol{H}_P(\mu\boldsymbol{I} - \boldsymbol{X}\boldsymbol{Z})\|_F$ の計算が可能である。また、$\|\boldsymbol{H}_P(\boldsymbol{J}\boldsymbol{Z})\|_F$は、以下で述べるような計算方法で、計算される。

HRVW/KSH/M 方向の場合 ：

$$\begin{aligned}
\|\boldsymbol{H}_{Z^{1/2}}(\boldsymbol{J}\boldsymbol{Z})\|_F &= \|((\boldsymbol{Z}^{1/2}\boldsymbol{J}\boldsymbol{Z}^{1/2}) + (\boldsymbol{Z}^{1/2}\boldsymbol{J}\boldsymbol{Z}^{1/2})^T)/2\|_F \\
&= \|((\boldsymbol{Z}^{1/2}\boldsymbol{J}\boldsymbol{Z}^{1/2}) + (\boldsymbol{Z}^{1/2}\boldsymbol{J}\boldsymbol{Z}^{1/2})^T)/2\|_F \\
&= \|(\boldsymbol{Z}^{1/2}\boldsymbol{J}\boldsymbol{Z}^{1/2})\|_F \\
&= \mathbf{Tr}((\boldsymbol{Z}^{1/2}\boldsymbol{J}\boldsymbol{Z}^{1/2})(\boldsymbol{Z}^{1/2}\boldsymbol{J}\boldsymbol{Z}^{1/2})^T) \\
&= \mathbf{Tr}(\boldsymbol{J}\boldsymbol{Z}\boldsymbol{J}\boldsymbol{Z}) \\
&= \sum_{i=1}^{n}\sum_{j=1}^{n}(\boldsymbol{J}\boldsymbol{Z})_{ij}(\boldsymbol{J}\boldsymbol{Z})_{ji}.
\end{aligned}$$

以上より、条件 (15) の左辺の値は以下の計算方法で計算される。

$$\left.\begin{array}{l}
\boldsymbol{G}\boldsymbol{w} = \boldsymbol{r},\ \boldsymbol{J} = \displaystyle\sum_{j=1}^{m}\boldsymbol{A}_j\boldsymbol{w}_j,\ \boldsymbol{H} = \boldsymbol{J}\boldsymbol{Z}, \\
\text{条件 (15) の左辺の値} = \displaystyle\sum_{i=1}^{n}\sum_{j=1}^{n}(\boldsymbol{H})_{ij}(\boldsymbol{H})_{ji},
\end{array}\right\} \tag{16}$$

ただし、$G_{ij} = \boldsymbol{A}_i \bullet \boldsymbol{A}_j\ (1 \le i \le m,\ 1 \le j \le m)$. この計算方法により、従来大きさ $n \times n$ の行列の掛け算 6 回分 ($12n^3$flops) ほどの計算量が必要だったのに対し、大きさ $n \times n$ の行列の掛け算 1 回分 ($2n^3$flops) ほどで条件 (15) の左辺の計算が可能になった。

なお条件 (15) の右辺は、以下のように計算できる。

$$\begin{aligned}
\|\boldsymbol{H}_{Z^{1/2}}(\mu\boldsymbol{I} - \boldsymbol{X}\boldsymbol{Z})\|_F &= \|((\mu\boldsymbol{I} - \boldsymbol{Z}^{1/2}\boldsymbol{X}\boldsymbol{Z}^{1/2}) + (\mu\boldsymbol{I} - \boldsymbol{Z}^{1/2}\boldsymbol{X}\boldsymbol{Z}^{1/2})^T)/2\|_F \\
&= \|(\mu\boldsymbol{I} - \boldsymbol{Z}^{1/2}\boldsymbol{X}\boldsymbol{Z}^{1/2})\|_F \\
&= \mathbf{Tr}((\mu\boldsymbol{I} - \boldsymbol{Z}^{1/2}\boldsymbol{X}\boldsymbol{Z}^{1/2})(\mu\boldsymbol{I} - \boldsymbol{Z}^{1/2}\boldsymbol{X}\boldsymbol{Z}^{1/2})^T) \\
&= \mathbf{Tr}((\mu\boldsymbol{I} - \boldsymbol{X}\boldsymbol{Z})(\mu\boldsymbol{I} - \boldsymbol{X}\boldsymbol{Z})) \\
&= \sum_{i=1}^{n}\sum_{j=1}^{n}(\mu\boldsymbol{I} - \boldsymbol{X}\boldsymbol{Z})_{ij}(\mu\boldsymbol{I} - \boldsymbol{X}\boldsymbol{Z})_{ji}.
\end{aligned}$$

NT 方向の場合 ：

$$\begin{aligned}
\|\boldsymbol{H}_{W^{-1/2}}(\boldsymbol{J}\boldsymbol{Z})\|_F &= \|((\boldsymbol{W}^{-1/2}\boldsymbol{J}\boldsymbol{Z}\boldsymbol{W}^{1/2}) + (\boldsymbol{W}^{-1/2}\boldsymbol{J}\boldsymbol{Z}\boldsymbol{W}^{1/2})^T)/2\|_F \\
&= \|((\boldsymbol{W}^{-1/2}\boldsymbol{J}\boldsymbol{Z}\boldsymbol{W}^{1/2}) + (\boldsymbol{W}^{-1/2}\boldsymbol{J}\boldsymbol{Z}\boldsymbol{W}^{1/2})^T)/2\|_F \\
&= \frac{1}{4}\sum_{i=1}^{n}\sum_{j=1}^{n}((\boldsymbol{W}^{-1/2}\boldsymbol{J}\boldsymbol{Z}\boldsymbol{W}^{1/2})_{ij} + (\boldsymbol{W}^{-1/2}\boldsymbol{J}\boldsymbol{Z}\boldsymbol{W}^{1/2})_{ji})^2.
\end{aligned}$$

これを用いると、条件 (15) の左辺の値は以下の計算方法で計算される。

$$\left.\begin{array}{l} \boldsymbol{G}\boldsymbol{w} = \boldsymbol{r},\ \boldsymbol{J} = \sum_{j=1}^{m} \boldsymbol{A}_j \boldsymbol{w}_j,\ \boldsymbol{H} = \boldsymbol{W}^{-1/2}\boldsymbol{J}\boldsymbol{Z}\boldsymbol{W}^{1/2}, \\ \text{条件 (15) の左辺の値} = \dfrac{1}{4}\sum_{i=1}^{n}\sum_{j=1}^{n}(\boldsymbol{H}_{ij} + \boldsymbol{H}_{ji})^2, \end{array}\right\} \tag{17}$$

この計算方法により、従来大きさ $n \times n$ の行列の掛け算 6 回分 ($12n^3$flops) ほどの計算量が必要だったのに対し、大きさ $n \times n$ の行列の掛け算 3 回分 ($6n^3$flops) ほどで条件 (15) の左辺の計算が可能になった。

なお条件 (15) の右辺は、以下のように計算される。

$$\begin{aligned} & \|\boldsymbol{H}_{W^{-1/2}}(\mu\boldsymbol{I} - \boldsymbol{X}\boldsymbol{Z})\|_F \\ = \ & \|((\mu\boldsymbol{I} - \boldsymbol{W}^{-1/2}\boldsymbol{X}\boldsymbol{Z}\boldsymbol{W}^{1/2}) + (\mu\boldsymbol{I} - \boldsymbol{W}^{-1/2}\boldsymbol{X}\boldsymbol{Z}\boldsymbol{W}^{1/2})^T)/2\|_F \\ = \ & \|(\mu\boldsymbol{I} - \boldsymbol{W}^{-1/2}\boldsymbol{X}\boldsymbol{W}\boldsymbol{X}\boldsymbol{W}^{-1/2})\|_F \\ = \ & \mathbf{Tr}((\mu\boldsymbol{I} - \boldsymbol{W}^{-1/2}\boldsymbol{X}\boldsymbol{W}\boldsymbol{X}W^{-1/2})(\mu\boldsymbol{I} - \boldsymbol{W}^{-1/2}\boldsymbol{X}\boldsymbol{W}\boldsymbol{X}W^{-1/2})^T) \\ = \ & \mathbf{Tr}((\mu\boldsymbol{I} - \boldsymbol{X}\boldsymbol{Z})(\mu\boldsymbol{I} - \boldsymbol{X}\boldsymbol{Z})) \\ = \ & \sum_{i=1}^{n}\sum_{j=1}^{n}(\mu\boldsymbol{I} - \boldsymbol{X}\boldsymbol{Z})_{ij}(\mu\boldsymbol{I} - \boldsymbol{X}\boldsymbol{Z})_{ji}. \end{aligned}$$

6 SDPA-CG の性能評価

最初に SDPA-CG の理論的計算量・使用メモリ量について述べ、SDPA のそれとの比較を行う。次に、SDPA-CG での計算実験を行い、SDPA と比較をすることで、SDPA-CG の特徴や SDPA に対する利点について考察を加える。なお、共役勾配法の前処理は原則として対角スケーリングを使用している。また SDPA が Mehrotra タイプの主双対内点法であるのに対し、SDPA-CG は単純なパス追跡タイプの主双対内点法 (Mehrotra タイプでは無い) であることを付け加えておく。

まず、線形方程式系を共役勾配法で解いたときの計算量について述べる。HRVW/KSH/M 方向の場合は、$\boldsymbol{B}\boldsymbol{p}$ の計算には大きさ n の行列の掛け算 2 回分の計算量が必要であり、終了条件の計算には大きさ n の行列の掛け算 1 回分の計算量が必要であるため、これらを合計して

大きさ n の行列同士の掛け算 3 回分 ($6n^3$flops) × 共役勾配法の反復回数

の計算量が必要であり、$n \times n$ の行列数個分のメモリが必要となる。また、NT 方向の場合は、$\boldsymbol{B}\boldsymbol{p}$ の計算に大きさ n の行列の掛け算 2 回分の計算量が必要であり、終了条件の計算には大きさ n の行列の掛け算 3 回分の計算量が必要であるため、これらを合計して

大きさ n の行列同士の掛け算 5 回分 ($10n^3$flops) × 共役勾配法の反復回数

の計算量が必要であり、$n \times n$ の行列数個分のメモリが必要となる。よって n に対し m が大きな SDP を解く場合、メモリ使用量の観点からは、SDPA-CG は SDPA に比べ $m \times m$ の係数行列 $\boldsymbol{B}$ を保持しないことが、非常に有利に働く。

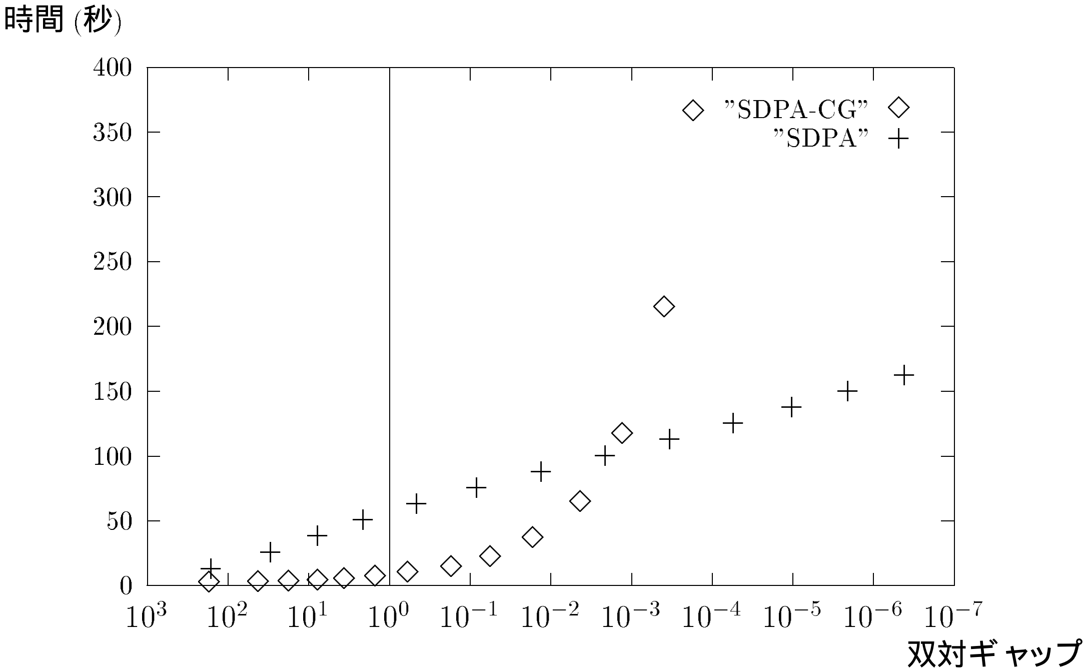
次に計算時間の観点から、SDPA-CG と SDPA を比べる。3 節で述べたように、SDPA 各主反復（主双対内点法の反復）でかかる時間の多くは、線形方程式系 $\boldsymbol{B}d\boldsymbol{y} = \boldsymbol{b}$ に費やされる。また共役勾配法の計算時間はその主反復回数に比例している。したがって、SDPA-CG の各主反復でかかる時間は、共役勾配法の反復回数に大きく依存しているといえる。表 2 は、疎で大規模な問題を解いたときの、SDPA の各主反復に関しての双対残差, 累積計算時間と、SDPA-CG の各主反復に関しての双対残差, 累積計算時間, 共役勾配法の反復回数を示している。また、この結果をグラフにしたものが図 1 であり、横軸に双対残差の対数、縦軸に累積計算時間を取っている。

表 2: 共役勾配法の反復回数

	SDPA		SDPA-CG		
内点法の主反復回数	累積計算時間 (秒)	双対残差	累積計算時間 (秒)	双対残差	共役勾配法の主反復回数
1	13.5	1.61e+02	1.5	1.73e+02	1
2	25.9	2.95e+01	2.0	4.26e+01	2
3	38.4	7.77e+00	2.6	1.77e+01	3
4	50.8	2.11e+00	3.3	7.78e+00	7
5	63.2	4.52e–01	4.4	3.57e+00	21
6	75.7	8.19e–02	6.3	1.49e+00	51
7	88.1	1.32e–02	9.2	5.85e–01	98
8	100.6	2.10e–03	13.8	1.72e–01	171
9	113.0	3.32e–04	21.9	5.57e–02	319
10	125.4	5.50e–05	36.3	1.65e–02	586
11	137.9	1.02e–05	63.5	4.32e–03	1124
12	150.3	2.06e–06	116.8	1.30e–03	2214
13	162.8	4.15e–07	214.2	3.97e–04	4084
14			365.8	1.69e–04	6392

表 2 より、SDPA ではその 1 主反復で双対残差はほぼ一定の割合で減り、計算時間もほぼ同じであるが、SDPA-CG では 1 主反復で双対残差はほぼ一定の割合で減るものの、各主反復での共役勾配法の反復回数は双対残差が小さくなるにつれて増えており、それにともない各反復での計算時間も増大していることが分かる。またこれらをグラフにした図 1 では、SDPA においては双対残差の対数と累積計算時間はほぼ比例しているが、SDPA-CG では双対残差の対数にたいして累積計算時間は指数的に増えている。これらにより、SDPA-CG では、双対残差が減るにしたがって、共役勾配法の反復回数が指数的に増え、それにとも

図 1: SDPA と SDPA-CG の双対ギャップ - 累積計算 グラフ



ない、１主反復にかかる計算時間も増えていることが分かる。

つまり SDPA-CG は SDPA に比べ、双対残差がある程度大きな精度の悪い近似最適解を高速に求めることができる。一方、SDPA-CG で高精度の近似最適解を求めることは、計算時間の点から困難である。よって、SDPA-CG は高精度の近似最適解ではなく上界や下界となる整数値を必要とする組合せ最適化問題の応用などに有効であると考えられる。

表３は、2.4 節で実験した疎で大規模な問題を SDPA-CG によって解いたときの、計算時間と使用メモリを示した。終了条件は 2.4 節と同様に、実行可能解の双対残差が 0.1 以下になったときである。

表 3: 疎で大規模な問題を SDPA-CG によって解いたときの計算時間と使用メモリ

n	100	100	100
m	1024	2029	3992
$\boldsymbol{B}d\boldsymbol{y} = \boldsymbol{b}$ の計算時間	19.7 秒	27.0 秒	47.4 秒
その他の部分の計算時間	4.5 秒	4.8 秒	5.7 秒
総計算時間	24.2 秒	31.8 秒	53.1 秒
使用メモリ	5.5MB	6.1MB	7.4MB

表３より、SDPA-CG は m が大きくなっても、総計算時間・メモリ使用量ともそれほど

増えていないことが分かる。また、表1と表3を比べることにより、SDPA-CG はこのような疎で n よりも m が 大きな大規模な問題に対し、総計算時間 ・使用メモリの両方において SDPA より優れていることが分かる。

7　数値実験

SDPA-CG を用いて、最大クリーク問題の SDP 緩和、グラフ分割問題の SDP 緩和、ノルム最小化問題の計算実験を行った。

最大クリーク問題の SDP 緩和は、m が $O(n^2)$ であり、n に対して m が非常に大きな問題である。また、制約行列が非常に疎であり、(13) や (17) に現れる行列$\boldsymbol{G}$が対角行列となる。グラフ分割問題の SDP 緩和は、SDP の主問題の内点実行可能解が存在しない問題である。一般にこのような問題は、既存のソフトウェアにおいても実行可能内点が存在する問題に比べ、主双対内点法の収束が悪くなる傾向がある。ノルム最小化問題は、上の2つの問題と違い、組合せ最適化問題ではない。この問題は、係数行列が密になる問題であり、既存のソフトウェアだけでなく SDPA においても、係数行列$\boldsymbol{B}$の各要素の計算に時間がかかる問題である。

すべての実験で探索方向は、HRVW/KSH/M 方向を使用し、共役勾配法の終了条件 (15) での κ は 0.1 とした。計算実験は、DEC Alpha (CPU 21164) 300MHz, 256MB メモリで行った。

7.1　最大クリーク問題の SDP 緩和

$G = (V, E)$ を無向グラフとする。ただし、$V = \{1, 2, \ldots, n\}$ は頂点集合、$E \subset \{(i, j) : i, j \in V,\ i < j\}$ は枝集合。V の部分集合である C に対して、異なる $i, j \in C (i < j)$ のすべてのペアに対して $(i, j) \in E$ であるとき、C はグラフ G のクリークであるという。最大クリーク問題とは、グラフ G の最大次数のクリークを見つける問題であり、NP-困難であることが知られている。次の SDP の最適値が、グラフ G の最大クリークの次数の上界を与えることが知られている。Helmberg 他 (1996) 等参照。

$$\left.\begin{array}{lll}
\text{主問題}: & \text{最大化} & \boldsymbol{E} \bullet \boldsymbol{X} \\
 & \text{制約条件} & \boldsymbol{E}_{ij} \bullet \boldsymbol{X} = 0 \ ((i,j) \notin E), \\
 & & \boldsymbol{I} \bullet \boldsymbol{X} = 1,\ \boldsymbol{X} \succeq \boldsymbol{O}. \\
\text{双対問題}: & \text{最小化} & y_0 \\
 & \text{制約条件} & y_0 \boldsymbol{I} + \displaystyle\sum_{(i,j) \notin E} y_{ij} \boldsymbol{E}_{ij} - \boldsymbol{Z} = \boldsymbol{E},\ \boldsymbol{Z} \succeq \boldsymbol{O}.
\end{array}\right\} \tag{18}$$

ただし、$\boldsymbol{E}$ は大きさ $n \times n$ で要素が全て1の行列、$\boldsymbol{E}_{ij}$は大きさ $n \times n$ で (i, j) 番目と (j, i) 番目の要素が1で残りの要素が0の行列を表す。

この SDP では、$|E|$ を枝の数とすると、$m = n(n-1)/2 - |E| + 1$ が成り立つ。したがって、m の大きさは $O(n^2)$ であり、n に比べ m が極めて大きくなり得る。また、全て

の制約行列は疎であり、(13)および(17)に現れる行列$\boldsymbol{G}$は対角行列となる。計算実験ではこの性質を有効に利用している。

数値実験は、ランダムグラフに対するSDP (18)の計算を行った。このランダムグラフは2つのパラメータn, pからなる。パラメータnはグラフの頂点の数、パラメータp $(0 < p < 1)$は枝の生成確率である。実験結果は表4に示す。表4には、SDPAおよびSDPA-CGの主反復回数、計算時間、メモリ使用量、ならびにSDPA-CGの最終主反復での共役勾配法の反復回数が示してある。なお、表中の空白の部分は、主記憶上に全データを保持できず、解くことが不可能あったたことを表す。また、SDPAおよびSDPA-CGの終了条件は双対ギャップが0.1以下とした。初期点$(\boldsymbol{X}^0, \boldsymbol{y}^0, \boldsymbol{Z}^0)$として、以下のような点を使用した。

$$\boldsymbol{X}^0 = \frac{1}{n}\boldsymbol{I},\ y_0^0 = 10n,\ y_{ij}^0 = 0\ (\forall(i,j) \notin E),\ \boldsymbol{Z}^0 = 10n\boldsymbol{I} - \boldsymbol{E}.$$

実験結果より、SDPA-CGはSDPAに比べ計算時間とメモリ使用量が大幅に改善されていることが分かる。

7.2 グラフ分割問題のSDP緩和

$G = (V, E)$を完全無向グラフとする。ただし、$V = \{1, 2, \ldots, n\}$は頂点集合、$E \subset \{(i,j) : i, j \in V,\ i < j\}$は枝集合。重み$C_{ij} = C_{ji}$ $((i,j) \in E)$も与えられている。nは偶数と仮定する。グラフ分割問題は、$|L| = |R| = n/2$となるようなVの分割(L, R)の中で、LとRにまたがる枝の重みの総和$\sum_{i \in L, j \in R} C_{ij}$を最小とする分割$(L, R)$を求める問題であり、NP-困難であることが知られている。以下のSDPの最適値が、最小分割の値の下界となる。Helmberg 他 (1996) 参照。

$$\left.\begin{array}{lll}
\text{主問題 ：} & \text{最小化} & \boldsymbol{A}_0 \bullet \boldsymbol{X} \\
 & \text{制約条件} & \boldsymbol{E}_{ii} \bullet \boldsymbol{X} = 1/4\ (1 \le i \le n), \\
 & & \boldsymbol{E} \bullet \boldsymbol{X} = 0,\ \boldsymbol{X} \succeq \boldsymbol{O}. \\
\text{双対問題 ：} & \text{最大化} & \displaystyle\sum_{i=1}^{n} \frac{1}{4} y_i \\
 & \text{制約条件} & y_0 \boldsymbol{I} + \displaystyle\sum_{i=1}^{n} y_i \boldsymbol{E}_{ii} + \boldsymbol{Z} = \boldsymbol{A}_0,\ \boldsymbol{Z} \succeq \boldsymbol{O}.
\end{array}\right\} \tag{19}$$

ここで、$\boldsymbol{A}_0 = \mathrm{diag}(\boldsymbol{Ce}) - \boldsymbol{C}$、また、$\boldsymbol{E}_{ii}$は大きさが$n \times n$で$(i,i)$番目の要素が1で、残りの要素が0である行列を示している。

SDP (19)の主問題には、内点実行可能解が存在しない。一般にこのような問題は、SDPAなどの既存のソフトウェアにおいて、内点実行可能解が存在する同程度の問題に比べ、主双対内点法の収束が悪くなり、計算時間がかかる傾向がある。Fujisawa 他 (1997,1998) 等参照。

数値実験では、ランダムグラフに対するSDP(19)の計算を行った。初期点$(\boldsymbol{X}^0, \boldsymbol{y}^0, \boldsymbol{Z}^0)$として、$(100\boldsymbol{I}, \boldsymbol{0}, 100\boldsymbol{I})$を使用した。SDPA-CGでの計算の途中で共役勾配法によって、条件(15)を満たす近似探索方向を計算することができなくなることが判明した。表5には、

表 4: 最大クリーク問題の SDP 緩和での実験結果

			SDPA			SDPA-CG			
(n, p)	n	m	反復回数	時間 (秒)	メモリ (MB)	反復回数	時間 (秒)	メモリ (MB)	CG 法の反復回数
(100,0.9)	100	539	7	13.3	8.5	10	24.7	5.1	336
(100,0.7)	100	1530	6	237.3	26.1	8	16.8	5.6	222
(100,0.5)	100	2513	6	1116.1	60.0	8	17.7	6.2	208
(100,0.3)	100	3528	6	3078.2	102.0	9	25.8	6.7	259
(100,0.1)	100	4476	7	7249.7	175.0	10	19.9	7.2	80
(200,0.9)	200	1977	7	662.6	49.0	10	209.8	9.8	385
(200,0.7)	200	6054				9	223.6	14.9	450
(200,0.5)	200	9957				9	258.0	16.9	488
(200,0.3)	200	14032				9	179.6	18.9	277
(200,0.1)	200	17932				12	275.0	21.0	155
(300,0.9)	300	4568	8	8988.2	193.0	11	1331.9	25.4	778
(300,0.7)	300	13669				10	1442.3	30.2	833
(300,0.5)	300	22572				10	1289.0	34.9	697
(300,0.3)	300	31580				9	724.0	39.7	340
(300,0.1)	300	40388				12	1169.4	44.3	411
(400,0.9)	400	8116				11	3668.7	43.1	849
(400,0.7)	400	24079				10	4322.4	51.6	1114
(400,0.5)	400	40094				10	2994.6	60.0	641
(400,0.3)	400	56073				10	2483.7	68.4	516
(400,0.1)	400	71895				11	1996.6	77.3	304
(500,0.9)	500	12694				11	8102.6	65.9	939
(500,0.7)	500	37596				10	6973.3	79.0	838
(500,0.5)	500	62499				10	6152.9	92.1	694
(500,0.3)	500	87591				10	5053.4	105.0	530
(500,0.1)	500	112253				12	4662.9	118.0	358

その時点での SDPA-CG の主反復回数、計算時間、双対ギャップ、最終主反復での共役勾配法の反復回数を示した。実験結果より、SDPA-CG は内点実行可能解が存在しない問題に対しては、あまり有効でないことが分かる。

表 5: グラフ分割問題の SDP 緩和での実験結果

(n,p)	n	m	反復回数	計算時間	双対ギャップ	共役勾配法の反復回数
(100,0.1)	100	101	13	16.2	3.65e-01	121
(100,0.5)	100	101	14	22.8	1.96e-01	195
(100,0.9)	100	101	15	25.0	1.12e-01	206
(200,0.1)	200	201	15	225.0	4.22e-01	277
(200,0.5)	200	201	16	307.6	1.70e-01	366
(200,0.9)	200	201	15	207.6	4.55e-01	220
(300,0.1)	300	301	16	1088.5	4.92e-01	381
(300,0.5)	300	301	16	1415.5	4.21e-01	554
(300,0.9)	300	301	16	1134.1	3.69e-01	372
(400,0.1)	400	401	17	3469.4	5.82e-01	538
(400,0.5)	400	401	16	3075.7	8.88e-01	434
(400,0.9)	400	401	17	3679.2	4.69e-01	540
(500,0.1)	500	501	17	6939.9	6.41e-01	491
(500,0.5)	500	501	17	8983.4	6.96e-01	701
(500,0.9)	500	501	17	7916.8	6.77e-01	648

7.3 ノルム最小化問題

$\boldsymbol{F}_i \in R^{q \times r}$ $(0 \leq i \leq p)$ が与えられているとする。このとき、ノルム最小化問題は以下のように定義される。

$$\text{minimize} \quad \left\| \boldsymbol{F}_0 + \sum_{i=1}^{p} \boldsymbol{F}_i y_i \right\| \quad \text{subject to} \quad y_i \in \Re \ (1 \leq i \leq p).$$

ここで $\|C\|$ は C のスペクトルノルム、つまり $\|C\| = \boldsymbol{C}^T\boldsymbol{C}$ の最大固有値の平方根である。この問題は、次の SDP に変換できる。Todd 他 (1996) 等参照。

双対問題 ：

$$\begin{array}{ll} \text{最小化} & y_{p+1} \\ \text{制約条件} & \displaystyle\sum_{i=1}^{p} \begin{pmatrix} \boldsymbol{O} & \boldsymbol{F}_i^T \\ \boldsymbol{F}_i & \boldsymbol{O} \end{pmatrix} y_i + \begin{pmatrix} \boldsymbol{I} & \boldsymbol{O} \\ \boldsymbol{O} & \boldsymbol{I} \end{pmatrix} y_{p+1} + \begin{pmatrix} \boldsymbol{O} & \boldsymbol{F}_0^T \\ \boldsymbol{F}_0 & \boldsymbol{O} \end{pmatrix} = \boldsymbol{Z}, \\ & \boldsymbol{Z} \succeq \boldsymbol{O}. \end{array}$$

主問題 ：

$$\begin{array}{ll} \text{最大化} & -\begin{pmatrix} \boldsymbol{O} & \boldsymbol{F}_0^T \\ \boldsymbol{F}_0 & \boldsymbol{O} \end{pmatrix} \bullet \boldsymbol{X} \\ \text{制約条件} & \begin{pmatrix} \boldsymbol{O} & \boldsymbol{F}_i^T \\ \boldsymbol{F}_i & \boldsymbol{O} \end{pmatrix} \bullet \boldsymbol{X} = 0 \ (1 \leq i \leq p), \\ & \begin{pmatrix} \boldsymbol{I} & \boldsymbol{O} \\ \boldsymbol{O} & \boldsymbol{I} \end{pmatrix} \bullet \boldsymbol{X} = 1, \ \boldsymbol{X} \succeq \boldsymbol{O}. \end{array}$$

実験では、$p = 100, 200,\ q = r = 100$ の 2 つのノルム最小化問題を解いた。この問題は、制約行列が密になり、SDPA を含む既存のソフトウェアにおいて、係数行列 $\boldsymbol{B}$ の各要素の計算に計算時間の多くが費やされる問題である。SDPA-CG では、5.1 節で述べたように行列 $\boldsymbol{B}$を計算せず $\boldsymbol{Bp}$ を計算するため係数行列 $\boldsymbol{B}$ の各要素の計算を行わない。当然、制約行列が密であるため、$\boldsymbol{Bp}$ の計算も簡単には行えないが、共役勾配法の総反復回数によっては、SDPA よりも高速に探索方向を計算できる可能性がある。この問題では、係数行列 $\boldsymbol{B}$ の各要素の計算に時間がかかるので、前処理として対角スケーリングは行わず、直接、共役勾配法を適用した。初期点 $(\boldsymbol{X}^0, \boldsymbol{y}^0, \boldsymbol{Z}^0)$ として、以下のような点を使用した。

$$\begin{array}{rcl} y_i^0 & = & 0 \ (1 \leq i \leq p), \ y_{p+1}^0 = 1.1\|\boldsymbol{A}_0\|, \\ \boldsymbol{Z}^0 & = & \begin{pmatrix} \boldsymbol{I} & \boldsymbol{O} \\ \boldsymbol{O} & \boldsymbol{I} \end{pmatrix} y_{p+1}^0 + \begin{pmatrix} \boldsymbol{O} & \boldsymbol{F}_0^T \\ \boldsymbol{F}_0 & \boldsymbol{O} \end{pmatrix} \succ \boldsymbol{O}, \\ \boldsymbol{X}^0 & = & \dfrac{1}{n} \begin{pmatrix} \boldsymbol{I} & \boldsymbol{O} \\ \boldsymbol{O} & \boldsymbol{I} \end{pmatrix}. \end{array}$$

表 6 に $p = 100,\ q = r = 100$ のノルム最小化問題を SDPA および SDPA-CG で解いたときの、相対的双対ギャップが 10^{-1} ~ 10^{-6} 以下になるまでの、主反復回数、累積計算時間を示した。また、表 7 に $p = 200,\ q = r = 100$ のノルム最小化問題を SDPA および SDPA-CG で解いたときの、相対的双対ギャップが 10^{-1} ~ 10^{-6} 以下になるまでの、主反復回数、累積計算時間を示した。ノルム最小化問題は、前の 2 つの組合せ最適化問題と異なり、双対ギャップの精度にはさほど意味がないので、相対的双対ギャップを使用した。相対的双対ギャップとは、$P,\ D$をそれぞれ主問題と双対問題の目的関数値としたとき、

$$\frac{|P - D|}{\max\left\{1.0, \dfrac{|P| + |D|}{2}\right\}}$$

として定義される値である。

表 6: p = 100, q = r = 100 のノルム最小化問題の実験結果

相対的双対ギャップ	SDPA 反復回数	SDPA 累積計算時間 (秒)	SDPA-CG 反復回数	SDPA-CG 累積計算時間 (秒)
10^{-1}	4	369.2	8	255.9
10^{-2}	6	510.1	11	334.5
10^{-3}	7	580.6	13	395.0
10^{-4}	8	651.0	14	435.9
10^{-5}	9	721.5	16	536.2
10^{-6}	10	792.2	17	585.7

表 7: p = 200, q = r= 100 のノルム最小化問題の実験結果

相対的双対ギャップ	SDPA 反復回数	SDPA 累積計算時間 (秒)	SDPA-CG 反復回数	SDPA-CG 累積計算時間 (秒)
10^{-1}	4	794.4	5	368.7
10^{-2}	5	948.7	9	618.9
10^{-3}	7	1257.6	10	713.9
10^{-4}	8	1415.2	12	981.0
10^{-5}	9	1572.1	14	1358.4
10^{-6}	10	1733.6	15	1643.5

8　おわりに

本研究では主として以下の二つのことを行った。

1．SDP を主双対内点法により解くソフトウェア SDPA に共役勾配法を組み入れたソフトウェア SDPA-CG の開発した。SDPA-CG の大きな特徴は、主双対内点法の各反復で探索方向の計算の際に現れる線形方程式系の係数行列を保持していないことにある。これにより、メモリ使用量を少なく押さえることができ、n に対し m が大きな大規模な SDP を扱うこと可能になった。

2．最大クリーク問題の SDP 緩和、グラフ分割問題の SDP 緩和、および、ノルム最小化問題に対する実験的解析を通して、SDPA-CG は問題によって非常に有効に働く場合と、困難を生ずる場合があることを示した。

   (a) 最大クリーク問題の SDP 緩和に対しては、SDPA-CG は SDPA に比べ、計算時間およびメモリ使用量が大幅に改善され、今まで解くことができなかった超大規模な SDP を解くことができた。
   (b) グラフ分割問題の SDP 緩和に対しては、反復の途中で、共役勾配法を使っての近似探索方向が計算できなくなり、解を得ることに失敗した。これはこの問題が内点実行可能解を持たないことによる。
   (c) 制約行列が密であるノルム最小化問題に対しては、SDPA に比べ何割か高速に解を得た。

また、計算実験を通して、

   (d) 精度の良い解を求めようとすると共役勾配法の反復回数は指数的に増え、それに伴い計算時間が増大する

ことが分かった。よって、高精度の近似最適解を求めることは現段階では困難である。それゆえ、高精度の近似最適解を求めなくとも、上界、下界などの整数値が求められれば十分な組合せ最適化問題への応用などに、SDPA-CG は有効であると思われる。

謝辞

本研究の一部は、統計数理研究所共同研究 (9-共研 B-2)、同 (9-共会-2)、同 (9-共会-3) に基づいて行われた。ここに、共同研究と研究会のメンバーに加えて下さった田辺國士教授、水野真治助教授、土谷隆助教授に感謝の意を表します。

参考文献

[1] F. Alizadeh, J.-P.A. Haeberly, M.V. Nayakkankuppam, M.L. Overton and S. Schmieta, "SDPpack – User's Guide –," Comp. Sci. Dept., New York University, New York, June 1997. Available at http://cs.nyu.edu/cs/faculty/overton/sdppack/sdppack.html.

[2] B. Borchers, "CSDP, a C library for semidefinite programming," Department of Mathematics, New Mexico Institute of Mining and Technology, 801 Leroy Place Socorro, New Mexico 87801, March 1997. Available at http://www.nmt.edu/~borchers/csdp.html.

[3] K. Fujisawa, M. Fukuda, M. Kojima and K. Nakata, "Numerical Evaluation of SDPA(SemiDefinite Programming Algorithm)," Research Report B-330, Dept. of

Mathematical and Computing Sciences, Tokyo Institute of Technology, Oh-Okayama, Meguro, Tokyo 152, Japan, March 1997.

[4] K. Fujisawa, M. Kojima and K. Nakata, "SDPA(Semidefinite Programming Algorithm) – User's Manual –," Technical Report B-308, Department of Mathematical and Computing Sciences, Tokyo Institute of Technology, Oh-Okayama, Meguro, Tokyo 152, Japan, December 1995, Revised August 1996.
Available at ftp://ftp.is.titech.ac.jp/pub/OpRes/software/SDPA.

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

[6] G. H. Goulb and C. F. Van Loan, *Matrix Computations,* (The John Hopkins University Press, Baltimore, Maryland, 1983).

[7] C. Helmberg, F. Rendl, R.J. Vanderbei and H. Wolkowicz, "An interior-point method for semidefinite programming," *SIAM Journal on Optimization* **6** (1996) 342–361.

[8] M. Hestenes and E. Stiefel, "Methoeds of Conjugate Gradients for Solving Linear Systems," *J. Res. Nat. Bur. Standards*, **49** (1952) 409-436.

[9] M. Kojima, M. Shida and S. Shindoh, "Search directions in the SDP and the monotone SDLCP: generalization and inexact computation," Research Report B-327, Dept. of Mathematical and Computing Sciences, Tokyo Institute of Technology, Oh-Okayama, Meguro, Tokyo 152, Japan, March 1997.

[10] M. Kojima, S. Shindoh and S. Hara, "Interior-point methods for the monotone semidefinite linear complementarity problems," *SIAM Journal on Optimization* **7** (1997) 86–125.

[11] C.-J. Lin and R. Saigal, "Semidefinite Programming and the Quadratic Assignment Problem," 16th International Symposium on Mathematical Programming, Lausanne, Switzerland, August 1997.

[12] C.-J. Lin and R. Saigal, "On solving large-scale semidefinite programming problems – a case study of quadratic assiginment problem," Dept. of Industrial and Operations Engineering, University of Michigan, Ann Arbor, MI 48109, 1998.

[13] R.D.C. Monteiro, "Primal-Dual Path Following Algorithms for Semidefinite Programming," *SIAM Journal on Optimization* **7** (1997) 663–678

[14] R. Monteiro and P. Zhang, "A unified analysis for a class of path-following primal-dual interior-point algorithms for semidefinite programming," Technical report, School of Industrial and Systems Engineering Georgia Tech. USA, 1996.

[15] Yu.E. Nesterov and M.J. Todd, "Self-Scaled Cones and Interior-Point Methods in Nonlinear Programming," Working Paper, CORE, Catholic University of Louvain, Louvain-la-Neuve, Belgium, April 1994, to appear in *Mathematics of Operations Research.*

[16] Yu.E. Nesterov and M.J. Todd, "Primal-dual interior-point methods for self-scaled cones," Technical Report 1125, School of Operations Research and Industrial Engineering, Cornell University, Ithaca, New York 14853-3801, USA, 1995.

[17] F.A. Potra, R. Sheng and N. Brixius, "SDPHA – a MATLAB implementation of homogeneous interior-point algorithms for semidefinite programming," Department of Mathematics, University of Iowa, Iowa City, IA 52242, April 1997.
Available at http://www.math.uiowa.edu/r̃sheng/SDPHA/sdpha.html.

[18] M. Todd, "On search directions in interior-point methods for semidefinite programming," Technical Report No.1205, School of Operations Research and Industrial Engineering, Cornell University, Ithaca, NY 14853-3801, October 1997.

[19] M.J. Todd, K.C. Toh and R.H. Tütüncü, "On the Nesterov-Todd direction in semidefinite programming," Technical Report, School of Operations Research and Industrial Engineering, Cornell University, Ithaca, New York 14853-3801, USA, March 1996, Revised May 1996.

[20] Q. Zhao, S.E. Karisch, F. Rendl and H. Wolkowicz, "Semidefinite programming relaxations for the quadratic assignment problem," CORR Report 95-27, Department of Combinatorics and Optimization, University of Waterloo, Waterloo, Ontario, Canada, September 1996.