StairCase制約を持つ線形計画問題を高速に解くアルゴリズムの開発

東 優太

平成 26 年 11 月 24 日

1 はじめに

制御工学におけるモデル予測制御問題等、現在の状態が直前あるいは一定時間だけ前の状態や入力の影響を受けるような最適化問題は制約式が StairCase と呼ばれる特殊な構造をもつ。StairCase は小さなサイズの制約が互いに影響しながら複数個連結されることによって形成される大規模な制約式の構造であり、一般に解くために時間がかかる。しかし、連続的に一定時間内に最適化問題を解き続けることが求められる場合では、StairCase を高速に解く必要がある。

線形計画問題を解くためのアルゴリズムとして、Dantzig が提案したシンプレックス法や小島, 水野, 吉瀬らが提案した主双対内点法などが存在する。シンプレックス法は多項式時間アルゴリズムではないものの、たいてい変数の数と制約条件の数の大きい方のオーダーの反復回数で解が求まる [1] ため、実用上は高速である。しかし、シンプレックス法で使用されるピボット規則は多項式時間で終了しない問題があり、問題が大規模になるにつれ解くためにかかる時間は増大する。また、主双対内点法は多項式時間で解が求まり、特に大規模な問題に対してシンプレックス法よりも計算効率が高いことが知られている。本研究は大規模な StairCase 制約をもつ問題を、小さなサイズの問題に分割して解いて元の問題の最適解を得るアルゴリズムを提案し、シンプレックス法および主双対内点法との比較を行うことを目的とする。

本稿ではまず第 2 節で線形計画問題やそれを解くためのアルゴリズムであるシンプレックス法など基本的事項について解説する。第 3 節では StairCase 制約をもつ問題を高速に解くための変形について解説する。第 4 節では実際に StairCase 制約をもつ問題を解き、改訂シンプレックス法との性能比較を行う。最後に第 5 節で本研究のまとめと今後の展望について述べる。

2 前提知識

2.1 線形計画問題

線形計画問題とは数理計画問題のひとつである。数理最適化問題は以下のような数式で表される。

$$\begin{aligned} \text{最大化} \quad & f(x) \\ \text{制約} \quad & x \in X \end{aligned}$$

この関数 f を目的関数、集合 X を実行可能領域とよび、X に属するベクトル x を実行可能解とよぶ。線形計画問題は関数 f、集合 X がともに凸である凸計画問題に属し、関数 f が一次関数で実行可能領域が実数空間上であるような問題で、以下のような数式で表される。

$$\begin{aligned} \text{最大化} \quad & cx \\ \text{制約} \quad & Ax \leq b \end{aligned}$$

2.2 シンプレックス法

シンプレックス法は線形計画問題を解くアルゴリズムの一つであり、最も広く使用されている方法である。このアルゴリズムは実行可能領域の頂点のひとつから、目的関数を改善するような他の頂点へ移動を繰り返すことで最適解を得る。

1. 実行可能な頂点を一つ求める
2. 目的関数を改善できる方向にある、実行可能領域の頂点を列挙する
3. ピボット選択規則に基づき、次の頂点へ移動する
4. 目的関数が改善できるなら 2 へ、改善できなければ今の頂点が最適解

ピボット選択規則には最小添字規則や、最大改善規則などが存在する。特に最大改善規則は常に目的関数が最大改善される頂点を選択されるが、目的関数値が同じであるような頂点が隣接する場合には同じ頂点を巡回し続ける場合がある。巡回は最小添字規則で解決できることが知られている。

2.3 Dantzig-Wolfe Decomposition

任意の線形計画問題の解は、その実行可能領域の頂点を示すベクトルである可能基底解を正規化した、可能正規基底解 v の線形結合で表現できる。大規模な問題の制約のうち、一部の制約によって形成される実行可能領域を可能性機基底解で表現することで、元の問題の制約の行方向についてサイズを小さくできる。$m \times n$ 行列 A を任意に m' $\times n$ 行列 A' と $m''\times n$ 行列 A'' に分割し、対応する b についても同様に b' と b'' に分割したとする。このとき A'' と b'' の制約により得られる可能正規基底解 $v^1, ..., v^k$ を用いて、元の問題は以下のように表現される。

$$\begin{aligned} \text{最大化} \quad & \sum_{k=1}^{M} cv^k r_k \\ \text{制約} \quad & \sum_{k=1}^{M} r_k A' v^k \leq b' \\ & \sum_{k=1}^{M} r_k = 1 \end{aligned}$$

以降ではこの問題を親問題と表記する。元の問題の行方向のサイズが減少したことでピボット演算にかかる時間が減少するため、制約の分割を適切に行うことで、高速に解を求めることができる。また、複数の小行列による制約が互いに影響することない形をとる BlockAngularCase について、この手法で高速に解けることがわかっている。(引用)

2.4 Column Generation

親問題を解けば元の問題の最適解が得られるが、可能正規基底解は実行可能領域の頂点の数だけ存在するために、問題の列方向のサイズは非常に大きなものとなる。また、すべての可能正規基底解を求めるためには非常に時間がかかる。しかし、実際には親問題に対応する部分問題を解くことで、目的関数を改善しうる可能正規基底解を得ることができ、親問題の対応する制約の列を生成できる。部分問題は以下のように表現される。

$$\begin{aligned} \text{最大化} \quad & \hat{c}x \\ \text{制約} \quad & A''x \leq b'' \end{aligned}$$

$\hat{c}$ は親問題のピボット演算により更新される、元の問題の c に対応する部分である。

2.5 BlockAngularCase 制約

BlockAngularCase 制約は小行列が独立して階段状に連なった制約の形を指す。

$$A = \begin{bmatrix} a_{1\,1} & a_{1\,2} & 0 & 0 & \cdots \\ a_{2\,1} & a_{2\,2} & 0 & 0 & \cdots \\ 0 & 0 & a_{3\,3} & a_{3\,4} & \cdots \\ 0 & 0 & a_{4\,3} & a_{4\,4} & \cdots \\ & & \vdots & & \ddots \end{bmatrix}, b = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ \vdots \end{bmatrix}$$

BlockAngularCase 制約は小さなサイズの複数の問題とみなすことができるので、それぞれの問題でピボット演算にかかる時間は少なくて済む。そのため、元の問題のサイズが膨大であっても比較的高速に解くことができる。

3 StairCase 制約

3.1 StairCase 制約とは

StairCase 制約は以下のような小行列が互いに影響しながら階段状に連なった制約の形を指す。

$$A = \begin{bmatrix} a_{1\,1} & a_{1\,2} & 0 & 0 & \cdots & 0 & 0 \\ a_{2\,1} & a_{2\,2} & 0 & 0 & \cdots & 0 & 0 \\ 0 & a_{2\,3} & a_{2\,4} & 0 & \cdots & 0 & 0 \\ 0 & a_{3\,3} & a_{3\,4} & 0 & \cdots & 0 & 0 \\ \vdots & & & & \ddots & & \\ 0 & 0 & 0 & 0 & \cdots & a_{m-1\,n-1} & a_{m-1\,n} \\ 0 & 0 & 0 & 0 & \cdots & a_{m\,n-1} & a_{m\,n} \end{bmatrix}$$

BlockAngularCase とは異なり、複数の小さなサイズの問題として解くことはできない。そのため、Dantzig-Wolfe Decomposition で効率的に解くためには適切に変形を行う必要がある。

3.2 変形

StairCase に Dantzig-Wolfe Decomposition を適用する場合、適切な制約の分割を行って 2 つの BlockAngularCase とみなすことができる。

$$A' = \begin{bmatrix} a_{1\,1} & a_{1\,2} & 0 & 0 & \cdots \\ a_{2\,1} & a_{2\,2} & 0 & 0 & \cdots \\ 0 & 0 & a_{5\,3} & a_{5\,4} & \cdots \\ 0 & 0 & a_{6\,3} & a_{6\,4} & \cdots \\ \vdots & & & & \ddots \end{bmatrix}$$

$$A'' = \begin{bmatrix} 0 & a_{3\,2} & a_{3\,3} & 0 & 0 & \cdots \\ 0 & a_{4\,2} & a_{4\,3} & 0 & 0 & \cdots \\ 0 & 0 & 0 & a_{7\,4} & a_{7\,5} & \cdots \\ 0 & 0 & 0 & a_{8\,4} & a_{8\,5} & \cdots \\ \vdots & & & & & \ddots \end{bmatrix}$$

4 実験

4.1 実験内容

ランダムに生成した StairCase 制約をもつ線形計画問題を Dantzig-Wolfe Decomposition を用いて解き、シンプレックス法で解いた場合と処理時間について比較する。また、問題のサイズに対する処理時間の改善の効果を測定するため、StairCase を構成する小行列のサイズ m, n を固定し、その数 k を変化させたときの処理時間について比較する。生成する問題は以下の通りである。

$$m = 4, n = 10$$

$$k = 5, 7, ..., 101$$

$$c = \begin{bmatrix} c_1 & c_2 & \cdots & c_{k*n} \end{bmatrix}$$

$$i = 1, .., k$$

$$A^i = \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \\ 1 & \cdots & 1 \end{bmatrix}, b^i = \begin{bmatrix} 1 \\ \vdots \\ 1 \\ 1 \end{bmatrix}$$

$c_1, ..., c_{kn}$ は-1 以上 1 以下のランダムな実数

$a_{11}, ..., a_{mn}$ は-1 以上 1 以下のランダムな実数

4.2　実験環境

4.3　結果

4.4　考察

5　総括

参考文献

[1] Steve Smale(1983),On the average number of steps of the simplex method of linear programming