求解一类双线性规划问题的数值算法

摘要

我们考虑一类特殊的双线性规划问题, 其中变量为矩阵, 目标函数为双线性函数, 约束有单纯形约束、非负约束以及对角元为0的约束. 该问题可来源于材料计算中的最优运输问题. 我们分别从双线性规划和非凸二次规划两个方面回顾了现有的求解方法, 并指出了直接应用它们的弊端. 基于对问题特殊结构的剖析, 我们提出了基于交替方向乘子法的算法, 详细地介绍了其中子问题的求解. 之后, 我们给出了一些收敛性分析. 我们证明了在一定条件下我们的算法产生的迭代序列收敛于问题的稳定点. 最后我们在随机生成的小型、大型问题上进行数值实验, 讨论了算法中不同松弛因子、惩罚因子对算法效果的影响. 我们还比较了本文算法与求解非凸二次规划的逐步二次规划算法、积极集法, 得出本文算法在大型问题上更具优势的结论. 我们在文章的末尾给出了总结和展望.

关键词: 双线性规划, 非凸二次规划, 交替方向乘子法, 子问题, 收敛性证明

装 订 线

Numerical Algorithms for a Class of Bilinear Programming

ABSTRACT

We consider a special class of bilinear programming problems, in which the variables are matrices, the objective is bilinear and the constraints include simplex constraints, nonnegative constraints, and zero-diagonal-element constraints. The problem can be derived from the optimal transport problem in material computation. We review the existing methods from the points of both bilinear programming and nonconvex quadratic programming, and point out the drawbacks of directly applying them to the problem. By exploiting the special structure of the problem, we propose a algorithm based on Alternating Directions Methods of Multipliers. We also describe how to solve subproblems in detail. After that, we provide some convergence analysis. We prove that the iterative sequence generated by our algorithm converges to the critical point of the problem under certain conditions. Finally, we carry out the algorithm on some randomly generated small and large problems. Discussions on the effect of different relaxation factors and penalty factors are included. Furthermore, we make a comparison among our algorithm, Sequential Quadratic Programming, and Active-Set Methods. Our algorithm turns out to outperform the latter two methods on large problems. Conclusion and prospects can be viewed at the end of this paper.

Keywords: Bilinear Programming, Nonconvex Quadratic Programming, Alternating Direction Methods of Multipliers, Subproblem, Convergence Analysis

目录

装 订 线

1 引言

1.1 问题背景与陈述

本文所要研究的问题是一类特殊的双线性最优运输问题. 最优运输是多个学科交叉的研究领域, 包括概率、分析以及优化等. 最优运输研究的主要目标是建立有效比较概率分布的几何工具, 可见文[74]. 以法国数学家Gaspard Monge在二百多年前给出的问题为例: 当给定两个沙盘时(每盘沙子可以代表一个概率分布), 可以通过很多方式将一个沙盘运输到另一个沙盘. 基于运输单个沙粒的局部花费, 每一种运输方法均对应一个全局花费. 最优运输的目的就是寻找总体花费最少的运输方案, 从而进一步建立面向概率分布的几何工具集.

最优运输问题有着悠久而丰富的研究历史. 最早可以追溯到上文所提到十八世纪的Gaspard Monge. 而俄国数学家Kantorovich在上世纪四十年代给出了一种更实际的松弛形式, 并在上世纪九十年代因为一系列重要的数学理论成果得到进一步推动, 其中包括法国数学家Brenier的重要工作. 尤其重要的是, 多位菲尔兹奖获得者在最优运输理论研究中做出过重要贡献, 如法国数学家Cédric Villani、意大利数学家Alessio Figalli, 并且已经有多本重要专著, 见文[66, 65, 54, 49]. 在应用方面, 最优运输被广泛用于计算机科学领域, 尤其是在计算机图形学、计算机视觉、医学图像处理以及深度学习等方面取得了显著效果.

现在我们以现代数学语言陈述Gaspard Monge的最优运输问题:

定义 1.1 (Monge问题)**.** 令$P, Q \subset \mathbb{R}^d$. 令$f$为$P$上的概率密度函数, g为Q上的概率密度函数. $c : P \times Q \to [0, \infty)$为连续函数. Monge问题是要求满足$T\#f = g$的运输映射$T : P \to Q$使得$T$最小化泛函

$$M(T) := \int_P c(p, T(p)) f(p)\, \mathrm{d}p.$$

这里$T\#f = g$表示g是f在T下的转移(push-forward), 定义为

$$\int_B g(q)\, \mathrm{d}q = \int_{T^{-1}(B)} f(p)\, \mathrm{d}p, \quad \forall B \subset Q.$$

换句话说, 在概率密度g下集合B的质量等于在概率密度f下集合$T^{-1}(B)$的质量.

注意定义1.1中的c可以看做是成本泛函. 当f, g为离散概率分布函数时, Monge问题就约减为一个线性规划(LP). 假设P, Q是有限集, $P = \{p_1, \ldots, p_m\}, Q = \{q_1, \ldots, q_n\}$. 对应地, 我们有它们的概率(或质量)$\{f_1, \ldots, f_m\}, \{g_1, \ldots, g_n\}$. 于是极小化定义1.1中的泛函等价于求解以下LP:

$$\begin{aligned} \min_{a_{ij}} \quad & \sum_{i,j} c_{ij} a_{ij} \\ \text{s.t.} \quad & \sum_j a_{ij} = f_i, \quad i = 1, \ldots, m, \\ & \sum_i a_{ij} = g_j, \quad j = 1, \ldots, n, \\ & a_{ij} \geq 0, \quad i = 1, \ldots, m,\ j = 1, \ldots, n, \end{aligned} \tag{1.1}$$

其中$c_{ij} := c(x_i, y_j)$. 实际上此LP的输出$A = (a_{ij})_{m \times n}$就构成了一个运输映射. 其行指标代表源$P$中的位置, 列指标代表汇$Q$中的位置. 每个$a_{ij}$表示从$p_i$运输至$q_j$的总质量.

下面我们陈述本文考虑的一类非凸、双线性最优运输问题:

$$\begin{aligned}\min_{X,Y}\quad & \sum_{i\neq j}\frac{x_{ij}}{|r_i-r_j|}+\sum_{i\neq k}\frac{y_{ik}}{|r_i-r_k|}+\sum_{i,j,k:j\neq k}\frac{x_{ij}y_{ik}}{|r_j-r_k|}\\ \text{s.t.}\quad & \sum_j x_{ij}=\rho_i,\quad i=1,2,\dots,n,\\ & \sum_i x_{ij}=\rho_j,\quad j=1,2,\dots,n,\\ & \sum_k y_{ik}=\rho_i,\quad i=1,2,\dots,n,\\ & \sum_i y_{ik}=\rho_k,\quad k=1,2,\dots,n,\\ & x_{ij},y_{ik}\geq 0,\quad i,j,k=1,2,\dots,n,\\ & x_{ii},y_{ii}=0,\quad i=1,2,\dots,n,\end{aligned}\tag{1.2}$$

其中$X=(x_{ij})$, $Y=(y_{ij})\in\mathbb{R}^{n\times n}$为待求的矩阵变量, $r=(r_1,r_2,\dots,r_n)^T$, $\rho=(\rho_1,\rho_2,\dots,\rho_n)^T\in\mathbb{R}^n_+$. 另外, 距离信息$\{|r_i-r_j|\}$储存于矩阵$R=(r_{ij})\in\mathbb{R}^{n\times n}$中, 其中$r_{ij}=1/|r_i-r_j|$. 我们由$R$的定义立可得其对称性. 另外, $\mathrm{tr}(R)=0$. 在此我们说明, 对应于ρ_i的等式约束与非负约束合在一起, 本质上是将第i行(列)限制到单纯形上; $x_{ii},y_{ii}=0$的加入是为了排除平凡解, 即X,Y均是以$\{\rho_i\}$为对角元的对角矩阵.

联系问题(1.2)与问题(1.1), 我们得出:

(i) 在问题(1.2)中, 有两个待求的运输映射$X=(x_{ij})_{n\times n}$和$Y=(y_{ij})_{n\times n}$. 换句话说, 有两个运输工具.

(ii) 在问题(1.2)中, 源集即是汇集.

(iii) 在问题(1.2)中, 成本泛函取成了距离的反比例函数.

(iv) 问题(1.2)中目标函数的前两项类似于问题(1.1)的目标函数, 这表明运输映射X,Y应尽量少消耗成本, 并且每个位置的物质均要向其他位置运送, 不允许停留. 而问题(1.2)目标函数的最后一项则提出了更高的要求. 当r_j和r_k (这里我们将它们看做欧式空间中的点)距离较近时, X倾向于不将r_i中的物质运输到r_j, 而Y则倾向于不将r_i中的物质运输到r_k. 但若$r_j=r_k$, 则这样带来的额外成本是0.

1.2 研究现状

问题(1.2)是一个双线性规划问题, 其一般形式为

$$\begin{aligned}\min_{x,y}\quad & x^TQy+c^Tx+d^Ty\\ \text{s.t.}\quad & Ax+By\leq b,\\ & 0\leq x\leq a_1,\\ & 0\leq y\leq a_2,\end{aligned}\tag{1.3}$$

其中$b\in\mathbb{R}^p, a_1,c,x\in\mathbb{R}^m, a_2,d,y\in\mathbb{R}^n, Q\in\mathbb{R}^{m\times n}, A\in\mathbb{R}^{p\times m}, B\in\mathbb{R}^{p\times n}$. 更一般地, 问题(1.3)从属于更大的一类问题——二次规划. 显然, 问题(1.3)是一个非凸二次规划. 文[48]说明, 即使目标函数的二次项矩阵只有一个负特征值, 这个问题也是NP困难的.

双线性规划分为两大类, 一类为可分离约束的问题, 即问题(1.3)的可行域可表示成

$$x \in \mathcal{X}, y \in \mathcal{Y},$$

这里$\mathcal{X}, \mathcal{Y}$均为多面体. 另一类为联合约束的问题. 显然问题(1.2)是可分离约束的问题. 可分离约束双线性规划最早作为一个等价的二次规划, 出现在双矩阵博弈的求解中, 可见文[45]. 对于此双矩阵博弈, 问题(1.3)的目标函数有下界0, 且在问题全局解处达到此界. 为求解文[45]中的问题, 文[43]利用Rosen的梯度投影法(见文[53]), 而文[44]则使用文[4]的寻找多面体所有顶点的方法. 文[2]考虑了无已知下界的问题, 并设计了求得局部解的算法.

求解可分离约束双线性规划的主要方法包括割平面法、分支定界法、线性化方法及对偶技术. 割平面法均基于文[35]证明的关于可分离问题的解的性质: 如果多面体$\mathcal{X}, \mathcal{Y}$非空且有界, 则可以找到问题的一个最优解(x^*, y^*), 其中x^*是多面体$\mathcal{X}$的顶点, y^*是多面体$\mathcal{Y}$的顶点. 文[51]首先提出求解可分离问题的割平面法. 之后文[35]对其加以改进, 提出两阶段的算法. 尽管他的算法求解了所有的测试问题, 但他却不能保证算法能够收敛到全局解. 文[14]基于文[35]的算法, 加入条件以提高下界上升速度, 提出了加速方法.

利用线性规划的对偶理论, 可分离约束问题可写作等价的极大极小问题. 对此构造, 文[15]提出了一个有限终止的分支定界算法. 而文[19]则对文[61]中针对凹极小化问题提出的两个算法加以改进. 其中第一个是无收敛保证的割平面法, 第二个是扩大多面体法. 当$\mathcal{X}$多面体由文[73]给出的反例中的约束定义时, 后者会陷入循环. 受文[19]启发, 文[64, 63]改进了文[61]的策略, 利用极切得到了具有有限终止性的扩大多面体法和收敛的割平面法. 而后文[57]利用分离面切得到了割平面法的有限终止性. 文[3]分析了一个具有无界定义域的双线性规划最优解是否有界的问题. 在他们的算法中, 变形后的双线性问题被嵌入了分支定界法中. 近期, 文[1]提出了结合凹切和文[3]中的分支定界程序的方法. 这种方法首先用凹切减小可行域, 之后再约减的可行域上使用分支定界法. 不像传统的割平面法, 这种方法给$\mathcal{X}, \mathcal{Y}$都加了凹切, 因此使得计算负荷更加沉重.

以上方法中, 文[15]、文[64]、文[57]的方法保证对所有情形都有限终止. 这其中, 文[15]的方法需要每个父节点有$s + t - 1$个后代, 这里s为y的维数, t为定义$\mathcal{X}$多面体的矩阵的行数. 但是子代问题的规模沿着分支会有效地减小; 文[64]没有报告实施的情况, 但承认他们的程序需要大量的存储和列表处理; 而从报告的计算结果上看, 文[57]的程序似乎是最高效的. 这些方法中, 割平面法的收敛速度较慢. 这是因为当连续割平面接近最优解时, 平面彼此趋于平行, 这导致在每一次添加割平面时只切掉了可行域的一小部分. 而分支定界法在验证一个可行解是全局最优解时需要消耗大量的时间. 结合这两者的方法则可能会面临相同的问题.

问题(1.2)还可看做是一个非凸二次规划问题. 因此用于求解二次规划的算法可以直接用于问题(1.2), 例如逐步二次规划(SQP)算法、积极集法(Active-set Methods)和内点法(Interior-point methods), 可见文[47].

以上方法均以列向量作为求解对象. 直接用于如问题(1.2)以矩阵为变量的双线性问题将会引发计算量的问题.

关于问题(1.3)已有相当多的研究工作, 其中部分原因是其具有广泛的应用, 包括双矩阵博弈、动态马尔科夫分配问题、多商品网络流问题、某些动态生产问题、二次凹极小化问题、直线距离选址问题、三维分配问题以及一些互补规划问题. 可见文[62, 17, 33, 37, 36, 58].

1.3 本文的工作

我们充分考察了问题(1.2)的内部结构, 认识到其变量为矩阵以及其为双凸问题的本质, 采用交替方向乘子法(Alternating Direction Methods of Multipliers, ADMM). 我们详细讨论了算法中子问题的求解, 对子问题设计了高效的求解算法. 最后, 我们给出了相关收敛性定理和数值实验, 比较了我们设计的算法与求解非凸二次规划的SQP和积极集法, 说明了我们所设计算法在求解大型问题时的优越性.

1.4 记号说明

以下, 如不特别说明, $\|\cdot\|$表示欧式范数. I表示单位阵, 有时我们会加下标以表明它的阶数. $\mathrm{tr}(\cdot)$计算矩阵的迹, 或等价地, 矩阵特征值的和. 我们给矩阵加上标T表示转置运算. 我们用"$\circ$"表示两矩阵的阿达玛积, 用"$\otimes$"表示两矩阵的克罗内克积. 我们以符号"$\geq$"表示两矩阵间元素级的比较. 比如$A \geq B \Leftrightarrow a_{ij} \geq b_{ij}, \forall i, j$. 一般地, 我们以$\mathcal{L}_0, \mathcal{L}_A$分别表示拉格朗日函数和增广拉格朗日函数. 有时我们会给它们加上标表明其归属或相关性. 我们以$\mathrm{vec}(\cdot)$表示矩阵的向量化, 以$\mathbf{1}$表示全1向量. 我们也使用$\langle\cdot,\cdot\rangle$表示$\mathbb{R}^n$中两向量的欧式内积. 在提及向量空间时, 我们以$N(\cdot)$表示线性算子的核空间, 以dim表示空间的维数. 在算法描述中, 我们一般使用上标表示迭代数, 用下标表示分量.

1.5 文章结构

本文剩余部分组织如下: 我们在第2节提供本文所需的预备知识. 我们在第3节给出问题(1.2)的最优性条件, 并在第4节利用ADMM算法求解之. 第5节讨论算法的收敛性质. 第6节给出一些数值实验和讨论. 总结和未来工作展望可见第7节.

2 预备知识

2.1 标准矩阵内积

为了更加简洁地叙述问题, 我们引入$\mathbb{R}^{n\times n}$中的一种特殊内积, 它由矩阵Frobenius范数直接诱导: $\forall A, B \in \mathbb{R}^{n\times n}$,

$$\langle A, B\rangle := \mathrm{tr}(A^T B) = \sum_{i,j} a_{ij} b_{ij}.$$

特别地,

$$\langle A, A\rangle = \mathrm{tr}(A^T A) = \|A\|_F^2.$$

此内积对矩阵的求导规则为

$$\frac{\mathrm{d}}{\mathrm{d}A}\langle A, B\rangle = B, \quad \frac{\mathrm{d}}{\mathrm{d}B}\langle A, B\rangle = A.$$

因此, 上述问题(1.2)经过简单运算即可表述成简洁的矩阵形式:

$$\begin{aligned} \min_{X,Y} \quad & \langle R, X\rangle + \langle R, Y\rangle + \langle Y, XR\rangle \\ \text{s.t.} \quad & X\mathbf{1} = \rho, X^T\mathbf{1} = \rho, \mathrm{tr}(X) = 0, X \geq 0, \\ & Y\mathbf{1} = \rho, Y^T\mathbf{1} = \rho, \mathrm{tr}(Y) = 0, Y \geq 0, \end{aligned} \tag{2.1}$$

其中$\mathbf{1}$为全1向量.

注 1. 考虑$X \geq 0$的约束, $\mathrm{tr}(X) = 0$就等价于$x_{ii} = 0$, $i = 1, 2, \ldots, n$. 这对Y也是同样.

使用MATLAB求解二次规划的内置函数`quadprog()`, 我们在一些随机生成的问题上进行了实验, 发现所有输出的解$(\bar{X}, \bar{Y})$均满足$\bar{X} = \bar{Y}$. 因此, 我们做出如下假设:

假设 1. 问题(2.1)的所有稳定点(X, Y)均满足$X = Y$.

这样我们只需考虑下面的简化问题.

$$\begin{aligned} \min_{X} \quad & 2\langle X, R\rangle + \langle X, XR\rangle \\ \text{s.t.} \quad & X\mathbf{1} = \rho, X^T\mathbf{1} = \rho, \mathrm{tr}(X) = 0, X \geq 0. \end{aligned} \tag{2.2}$$

为方便后续的算法设计, 我们引入分裂变量$Z \in \mathbb{R}^{n\times n}$将问题的约束分成两部分, 同时将问题(2.2)目标函数的二次项替换成双线性项. 这样我们得到如下的等价问题:

$$\begin{aligned} \min_{X,Z} \quad & f(X, Z) \triangleq 2\langle X, R\rangle + \langle Z, XR\rangle \\ \text{s.t.} \quad & X\mathbf{1} = \rho, \mathrm{tr}(X) = 0, \\ & Z^T\mathbf{1} = \rho, Z \geq 0, \\ & X = Z. \end{aligned} \tag{2.3}$$

2.2 凸集

令V为实数域上的向量空间. 我们称V中的集合$\mathcal{C}$是凸的, 若对$\forall x, y \in \mathcal{C}$, $\forall t \in (0, 1)$, 我们有

$$(1 - t)x + ty \in \mathcal{C}.$$

换句话说, 连接x, y的线段位于$\mathcal{C}$中. 由定义, 问题(2.1)的可行域就是个$\mathbb{R}^{n\times n}$中的凸集.

3 最优性条件

下面我们推导问题(2.3)的KKT条件. 这对我们最终设置算法的停机准则会有帮助. 问题(2.3)的拉格朗日函数为

$$\begin{aligned}&\mathcal{L}_0(X,Z,\lambda_1,\lambda_2,\mu,\Phi,\Omega)\\&=f(X,Z)-\langle\lambda_1,X\mathbf{1}-\rho\rangle-\langle\lambda_2,Z^T\mathbf{1}-\rho\rangle\\&-\mu\mathrm{tr}(X)-\langle\Omega,Z\rangle-\langle\Phi,X-Z\rangle,\end{aligned}$$

其中$\mu\in\mathbb{R},\lambda_1,\lambda_2\in\mathbb{R}^n,\Phi,\Omega\in\mathbb{R}^{n\times n}$为拉格朗日乘子. 根据文[47], 我们有问题(2.3)的KKT条件: 若(X^*,Z^*)为问题(2.3)的解, 则存在拉格朗日乘子$\mu^*,\lambda_1^*,\lambda_2^*,\Phi^*,0\leq\Omega^*$, 使得

$$\left\{\begin{array}{ll}\left.\begin{array}{l}\nabla_X\mathcal{L}_0=2R+Z^*R-\lambda_1^*\mathbf{1}^T-\Phi^*-\mu I=0,\\\nabla_Z\mathcal{L}_0=X^*R-\mathbf{1}\left(\lambda_2^*\right)^T+\Phi-\Omega=0,\end{array}\right\} & \begin{array}{l}\text{稳定性条件}\\\text{或对偶可行性条件,}\end{array}\\\left.\begin{array}{l}X^*\mathbf{1}=\rho,\mathrm{tr}(X^*)=0,\\\left(Z^*\right)^T\mathbf{1}=\rho,Z^*\geq0,\\\Omega^*\geq0,\end{array}\right\} & \text{原始可行性条件,}\\\Omega^*\circ Z^*=0. & \text{互补松弛条件.}\end{array}\right.\tag{3.1}$$

4 算法设计

4.1 ADMM算法

ADMM最早在20世纪70年代被提出, 可见文[23, 18], 其源头可追溯到求解偏微分方程(可见文[6]). 关于ADMM的全面详述及其应用可见文[6].

ADMM算法求解问题的一般形式为

$$\begin{aligned} \min_{x\in\mathbb{R}^d} \quad & \theta(x) := \sum_{i=1}^{n} \theta_i(x_i) + \ell(x_1,\ldots,x_n) \\ \text{s.t.} \quad & \sum_{i=1}^{n} A_i x_i = b, \end{aligned} \tag{4.1}$$

这里$\theta_i : \mathbb{R}^{d_i} \mapsto (-\infty,+\infty](i=1,2,\ldots,n)$, $\ell : \mathbb{R}^d \mapsto (-\infty,+\infty]$; $x_i \in \mathbb{R}^{d_i}$; $A_i \in \mathbb{R}^{m\times d_i}$, $b\in\mathbb{R}^m$. 我们有问题(4.1)的增广拉格朗日函数

$$\overline{\mathcal{L}}_A(x_1,\ldots,x_n,\bar{\mu}) = \sum_{i=1}^{n}\theta_i(x_i) + \ell(x_1,\ldots,x_n) - \bar{\mu}^T\left(\sum_{i=1}^{n}A_ix_i - b\right) + \frac{\bar{\beta}}{2}\left\|\sum_{i=1}^{n}A_ix_i - b\right\|^2,$$

其中$\bar{\mu}\in\mathbb{R}^m$为拉格朗日乘子, $\bar{\beta}>0$为惩罚因子. 于是便有n块ADMM:

$$\begin{aligned} x_1^{k+1} &:= \arg\min_{x_1\in\mathbb{R}^{d_1}} \overline{\mathcal{L}}_A(x_1,\ldots,x_n^k,\bar{\mu}^k), \\ &\cdots\cdots \\ x_n^{k+1} &:= \arg\min_{x_n\in\mathbb{R}^{d_n}} \overline{\mathcal{L}}_A(x_1^{k+1},\ldots,x_n,\bar{\mu}^k), \\ \bar{\mu}^{k+1} &:= \bar{\mu}^k - \bar{\beta}\left(\sum_{i=1}^{n}A_ix_i^{k+1} - b\right). \end{aligned}$$

关于ADMM算法的收敛性分析大多聚焦于问题(4.1)的特殊形式——两块凸可分问题, 即$n=2,\ell=0$, θ_1,θ_2都是凸函数. 此时, 在较弱的条件下, ADMM算法收敛; 可见文[6]. 在同样的条件下, 一些研究工作(如文[29, 46, 12])表明ADMM算法以$\mathcal{O}(\frac{1}{t})$或$o(\frac{1}{t})$的速度次线性收敛, 且在适当加速后, 速度达到$\mathcal{O}(\frac{1}{t^2})$ (见文[25, 24]). 进一步地, 文[13]表示ADMM在目标函数和约束满足特定条件时是线性收敛的.

对于$n\geq 3$的情形, 大量的研究工作集中在将多块ADMM及其变体应用于带线性约束的可分凸优化问题上, 即问题(4.1)去掉ℓ, $\theta_i, i=1,\ldots,n$都是凸函数. 文[8]表明n块ADMM算法在某些病态问题上可能会发散. 因此, 多数研究的切入点, 或是在问题上添加额外的条件, 或是证明ADMM算法变体的收敛性. 一种典型的方式是合并n块ADMM的所有校正步(见文[28, 29, 27]). 另外, 若问题(4.1)的目标函数中至少有$n-2$个函数是强凸的且惩罚因子被限制在一个特定的范围内, 则n块ADMM算法是全局收敛的(见文[7, 10, 26, 38, 42, 72]). 如若没有强凸的条件, 则在目标函数满足特定的误差界条件时, 若以小步长更新对偶变量, 即乘子更新变为

$$\bar{\mu}^{k+1} = \bar{\mu}^k - \tau\bar{\beta}\left(\sum_{i=1}^{n}A_ix_i^{k+1} - b\right),$$

则n块ADMM算法是线性收敛的(见文[31]). 文[40, 41]则说明了多块ADMM算法在其他一些条件下的收敛性. 文[60]提出了多块ADMM算法的一种随机变体RPADMM. 在每一步, RPADMM构

造$\{1,2\ldots,n\}$的一个随机排列, 然后按排列的顺序更新原始变量$x_i(i=1,2,\ldots,n)$. 令人惊喜的是, RPADMM对任意非奇异的方形线性系统都是按期望收敛的.

相较于可分的情形, 在不可分的问题(即使$n=2$)上的研究非常有限. 文[32]提到当目标函数对于变量来说是不可分时, 即使$n=2$且$\theta(\cdot)$为凸, 多块ADMM的收敛性仍然是开放的问题. 文[30]说明当问题(4.1)是凸的但不可分且满足一定的误差界条件时, 若乘子的更新步长充分的小, 则ADMM迭代收敛到某个原始-对偶最优解. 尽管如此, 步长通常依赖于与误差界相关的一些未知参数, 因此很难计算, 这样反而不利于算法的高效实施. 因此, 往往直接使用经典的ADMM迭代($\tau=1$)或者其变体($\tau\geq 1$). 文[11]讨论了求解有一耦合光滑目标函数的凸优化问题的一种优化ADMM算法. 文[20]研究了2块邻近ADMM及其变体在不可分凸优化问题上的收敛性和遍历复杂度, 其中假设了在问题数据上的一些附加条件. 文[9]则分析了求解不可分凸优化问题时, 邻近ADMM和RPADMM产生的序列的收敛性.

对于目标是非凸的情形, ADMM算法的收敛性也依然不是很清晰. 尽管如此, ADMM算法在多种涉及非凸目标函数的应用上都具有极佳的表现, 例如非负矩阵分解(可见文[71, 59])、分布式矩阵分解(可见文[70])、分布式聚类(可见文[16])、张量分解(可见文[39])、资产配置(可见文[68])等. 然而现有的关于ADMM算法在非凸问题上的分析十分有限——所有已知的全局收敛性分析都需要在迭代序列上强加无法检验的条件. 例如, 文[34, 56, 69, 68]表明在假设聚点存在以及相邻迭代点(原始与对偶)的差趋近于0的条件下, ADMM在某些非凸问题上全局收敛到稳定点的集合. 但这样的假设条件限制过强. 同时, 去除在迭代点上的假设, ADMM是否还有相同的收敛结果还未可知. 文[72]分析了在特定非凸二次规划问题上的一族分裂算法(其中ADMM为特殊情形), 并且说明它们在对偶步长满足一定条件时收敛到稳定解. 当然也有许多研究工作提出了求解非凸非光滑问题的新算法, 例如文[50, 21, 22, 55, 5]. 但它们都不能处理带线性耦合约束的非凸问题. 文[32]建立了ADMM对某些特定类型的非凸问题的收敛性, 其中没对序列作任何假设. 其结果表明, 只要目标函数中的θ_i's和ℓ满足特定的正则条件, 且惩罚因子$\bar{\beta}$选取得足够大, 则由ADMM算法产生的迭代点列收敛到问题的稳定点.

以上工作中, 与问题(2.2)最契合的是文[32]中考虑的非凸一致性问题:

$$\begin{aligned}\min_{x}\quad & \sum_{i=1}^{n}\theta_i(x)+\ell(x),\\ \text{s.t.}\quad & x\in\mathcal{X},\end{aligned}$$

其中θ_i's为光滑(可能)非凸的函数, ℓ为凸的非光滑正则化项. 在我们的问题(2.2)中, 显然$\theta(X)=2\langle X,R\rangle$, $\ell(X)=\langle X,XR\rangle$. $\ell(X)$非凸. 因此文[32]中的结论不可直接使用.

基于ADMM算法在许多非凸问题上的良好效果, 我们使用ADMM算法求解问题(2.3). 注意第2节中引入分裂变量方便了ADMM算法的设计. 此外, 将原本问题(2.2)目标函数中的二次项变为双线性项有利于ADMM子问题的求解. 这点我们会在后文说明. 问题(2.3)的增广拉格朗日函数为

$$\mathcal{L}_A(X,Z,\Phi)=f(X,Z)-\langle\Phi,X-Z\rangle+\frac{\beta}{2}\|X-Z\|_F^2. \tag{4.2}$$

这里$\beta\geq 0$为惩罚因子. 下面我们有对于问题(2.3)ADMM算法的一般框架. 见框架1. 我们在本节剩下的部分将详细介绍如何求解框架1中X,Z子问题的求解以及乘子(或对偶变量)的更新方式.

框架 1 求解问题(2.3)的ADMM算法框架

输入: $X^0, Z^0, \Phi^0, \beta^0$

输出: X^k, Z^k, Φ^k

1: **while** 收敛性测试未满足 **do**

2: $X^{k+1} = \arg\min_{X:X\mathbf{1}=\rho,\mathrm{tr}(X)=0} \mathcal{L}_A(X, Z^k, \Phi^k)$;

3: $Z^{k+1} = \arg\min_{Z:Z^T\mathbf{1}=\rho,Z\geq 0} \mathcal{L}_A(X^{k+1}, Z, \Phi^k)$;

4: 更新Φ^k得到Φ^{k+1};

5: 如有需要, 更新β^k得到β^{k+1};

6: $k := k + 1$;

7: **end while**

4.2 X子问题

设当前我们在第k步迭代. 下面我们重述X子问题. 我们省略上标k, 并用"+"表示新迭代点.

$$\begin{aligned} \min_X \quad & 2\langle X, R\rangle + \langle Z, XR\rangle - \langle \Phi, X - Z\rangle + \tfrac{\beta}{2}\|X - Z\|_F^2 \\ \text{s.t.} \quad & X\mathbf{1} = \rho, \quad \mathrm{tr}(X) = 0. \end{aligned} \tag{4.3}$$

上述问题实质上是一个带等式约束的凸二次规划. 因此我们可以直接使用拉格朗日乘数法得到解析解. 问题(4.3)的拉格朗日函数为

$$\begin{aligned} \mathcal{L}_0^X =& 2\langle X, R\rangle + \langle Z, XR\rangle - \langle \lambda_1, X\mathbf{1} - \rho\rangle - \mu\mathrm{tr}(X) \\ & - \langle \Phi, X - Z\rangle + \frac{\beta}{2}\|X - Z\|_F^2, \end{aligned} \tag{4.4}$$

其中$\mathcal{L}$的上标X表示其对X子问题的依赖(下对Z子问题同), $\mu \in \mathbb{R}, \lambda_1 \in \mathbb{R}^n$为乘子. 由表达式(4.4), 直接计算可得一阶充要最优性条件:

$$\begin{cases} \nabla_X \mathcal{L}_0^X = \beta X + (2R + ZR - \lambda_1\mathbf{1}^T - \mu I - \Phi - \beta Z) = 0, & \text{(4.5a)} \\ \nabla_\mu \mathcal{L}_0^X = -\mathrm{tr}(X) = 0, & \text{(4.5b)} \\ \nabla_{\lambda_1} \mathcal{L}_0^X = -(X\mathbf{1} - \rho) = 0. & \text{(4.5c)} \end{cases}$$

由等式(4.5a), 我们有

$$X^+ = -\frac{1}{\beta}(2R + ZR - \lambda_1\mathbf{1}^T - \mu I - \Phi - \beta Z). \tag{4.6}$$

结合公式(4.6)和等式(4.5b),(4.5c), 我们有

$$\begin{aligned}
0 &= \mathrm{tr}(X) \\
&= -\frac{1}{\beta}(2\mathrm{tr}(R) + \mathrm{tr}(ZR) - \mathrm{tr}(\Phi) - \beta\mathrm{tr}(Z) - \mathbf{1}^T\lambda_1 - n\mu) \\
&= -\frac{1}{\beta}(2\mathrm{tr}(R) + \mathrm{tr}(ZR) - \mathrm{tr}(\Phi) - \beta\mathrm{tr}(Z)) + \frac{1}{\beta}(\mathbf{1}^T\lambda_1 + n\mu), \\
\rho &= X\mathbf{1} \\
&= -\frac{1}{\beta}(2R + ZR - \Phi - \beta Z - \lambda_1\mathbf{1}^T - \mu I)\mathbf{1} \\
&= -\frac{1}{\beta}(2R\mathbf{1} + ZR\mathbf{1} - \Phi\mathbf{1} - \beta Z\mathbf{1}) + \frac{1}{\beta}(n\lambda_1 + \mathbf{1}\mu),
\end{aligned}$$

由此得到线性方程组

$$\begin{pmatrix} nI & \mathbf{1} \\ \mathbf{1}^T n & \end{pmatrix} \begin{pmatrix} \lambda_1 \\ \mu \end{pmatrix} = \begin{pmatrix} M_1 \\ m_1 \end{pmatrix}, \tag{4.7}$$

其中

$$\begin{aligned}
M_1 &= 2R\mathbf{1} + ZR\mathbf{1} - \Phi\mathbf{1} - \beta Z\mathbf{1} + \beta\rho, \\
m_1 &= 2\mathrm{tr}(R) + \mathrm{tr}(ZR) - \mathrm{tr}(\Phi) - \beta\mathrm{tr}(Z).
\end{aligned}$$

线性方程组(4.7)有解析解:

$$\mu = \frac{1}{n-1}\left(-\frac{1}{n}\mathbf{1}^T M_1 + m_1\right), \quad \lambda_1 = \frac{1}{n}(M_1 - \mathbf{1}\mu). \tag{4.8}$$

最终由公式(4.6)和公式(4.8)给出问题(4.3)解的具体形式.

4.3 Z子问题

我们重述Z子问题如下.

$$\begin{aligned}
\min_{Z} \quad & \langle Z, X^+R\rangle - \langle \Phi, X^+ - Z\rangle + \tfrac{\beta}{2}\|X^+ - Z\|_F^2 \\
\text{s.t.} \quad & Z^T\mathbf{1} = \rho, \quad Z \geq 0.
\end{aligned} \tag{4.9}$$

类似地, 问题(4.9)也是个凸二次规划. 求解之的主要困难在于不等式约束. 忽略非负约束, 问题(4.9)实际上就是要求我们在超平面$Z^T\mathbf{1} = \rho$上沿着目标函数的下降方向寻找可行解. 且在即将打破非负约束的地方停止搜索, 即得问题(4.9)的解. 求解凸二次规划可直接调用MATLAB的内置函数`quadprog()`. 考虑到问题(4.9)形式略为复杂, 之后我们将考虑问题(4.9)的等价形式问题(4.14). 其间的讨论和推导可作为它们等价性的说明. 我们将最后说明求解问题(4.14)等价于求解有限个互不相关的简单二次规划.

问题(4.9)$\Leftrightarrow$问题(4.14). 在超平面$Z^T\mathbf{1} = \rho$上寻求最优解, 我们可先在超平面上找到一点. 我们暂且忽略非负约束, 而仅考虑问题(4.9)中的等式约束. 于是我们有问题(4.9)的拉格朗日函数

$$\mathcal{L}_0^Z = \langle Z, X^+R\rangle - \langle \Phi, X^+ - Z\rangle + \frac{\beta}{2}\|X^+ - Z\|_F^2 - \langle \lambda_2, Z^T\mathbf{1} - \rho\rangle, \tag{4.10}$$

其中$\lambda_2 \in \mathbb{R}^n$为乘子. 由表达式(4.10), 直接计算可得一阶充要最优性条件:

$$\begin{cases}
\nabla_Z \mathcal{L}_0^Z = \beta Z + (X^+R + \Phi - \beta X^+ - \mathbf{1}\lambda_2^T) = 0, & \text{(4.11a)} \\
\nabla_{\lambda_2} \mathcal{L}_0^Z = -(Z^T\mathbf{1} - \rho) = 0. & \text{(4.11b)}
\end{cases}$$

由等式(4.11a)我们有

$$Z = -\frac{1}{\beta}(X^+R + \Phi - \beta X^+ - \mathbf{1}\lambda_2^T). \tag{4.12}$$

结合公式(4.12)和等式(4.11b), 我们有

$$\rho = Z^T\mathbf{1} = -\frac{1}{\beta}\left(R\left(X^+\right)^T\mathbf{1} + \Phi^T\mathbf{1} - \beta\left(X^+\right)^T\mathbf{1} - n\lambda_2\right),$$

这就给出

$$\lambda_2 = \frac{1}{n}\left[R\left(X^+\right)^T\mathbf{1} + \Phi^T\mathbf{1} - \beta\left(X^+\right)^T\mathbf{1} + \beta\rho\right]. \tag{4.13}$$

由公式(4.12)和公式(4.13)我们得到了等式约束下的解. 我们记之为$\widetilde{Z}$. 于是我们得到问题(4.9)的等价形式:

$$\begin{aligned} \min_{Z} \quad & \|Z - \widetilde{Z}\|_F^2 \\ \text{s.t.} \quad & Z^T\mathbf{1} = \rho, \quad Z \geq 0. \end{aligned} \tag{4.14}$$

以上问题(4.14)是方便用现有算法求解的. 事实上, 对$Z, \widetilde{Z}$做列分块,

$$Z = [z_1, \ldots, z_n], \quad \widetilde{Z} = [\tilde{z}_1, \ldots, \tilde{z}_n].$$

那么问题(4.14)就可以写成

$$\begin{aligned} \min_{z_1,\ldots,z_n} \quad & \sum_{j=1}^{n} \|z_j - \tilde{z}_j\|^2 \\ \text{s.t.} \quad & \mathbf{1}^T z_j = \rho_j, \quad z_j \geq 0, \quad j = 1, \ldots, n. \end{aligned} \tag{4.15}$$

显然, 问题(4.15)可以进一步分拆成n个互补相关的列子问题. 对每个列子问题, 由于已经是向量形式, 我们就可以直接用MATLAB内置函数`quadprog()`求解. 记更新后的Z为Z^+.

注 2. 以上两子问题的求解在没有将问题(2.2)目标函数的二次项替换成双线性项时是无法想象的.

4.4 拉格朗日乘子更新

对于一般的等式约束非线性优化问题

$$\begin{aligned} \min_{x} \quad & \bar{f}(x) \\ \text{s.t.} \quad & c_i(x) = 0, \quad i \in \mathcal{E}, \end{aligned} \tag{4.16}$$

其中$x \in \mathbb{R}^n$, $\mathcal{E}$表示等式约束的指标集. 于是有增广拉格朗日函数

$$\widetilde{\mathcal{L}}_A(x, \lambda) = \bar{f}(x) - \sum_{i\in\mathcal{E}} \lambda_i c_i(x) + \frac{\gamma}{2}\sum_{i\in\mathcal{E}} c_i^2(x).$$

这里$\lambda_i \in \mathbb{R}^n$为拉格朗日乘子, γ为惩罚因子. 令

$$x^k \in \arg\min_x \widetilde{\mathcal{L}}_A(x, \lambda^k),$$

我们有此无约束极小问题的最优性条件:

$$0 = \nabla_x \widetilde{\mathcal{L}}_A(x^k, \lambda^k) = \nabla \bar{f}(x^k) - \sum_{i\in\mathcal{E}} \left[\lambda_i^k - \gamma^k c_i(x^k)\right] \nabla c_i(x^k).$$

比较上式与问题(4.16)的KKT条件, 我们推出

$$\lambda_i^* \approx \lambda_i^k - \gamma^k c_i(x^k), \quad \forall i \in \mathcal{E}, \tag{4.17}$$

这里λ^*表示最优拉格朗日乘子. 文[47]表示, 公式(4.17)可作为增广拉格朗日函数法的乘子更新策略. 于是, 我们的ADMM算法中的乘子更新步可选为

$$\Phi^+ = \Phi - \beta(X^+ - Z^+). \tag{4.18}$$

进一步, 我们可以加入松弛因子$\alpha > 0$:

$$\Phi^+ = \Phi - \alpha\beta(X^+ - Z^+). \tag{4.19}$$

4.5 停机准则与KKT违反度

我们回顾问题(2.3)的KKT条件(4.20): 若(X^*, Z^*)为问题(2.3)的解, 则存在拉格朗日乘子$\mu^*, \lambda_1^*, \lambda_2^*, \Phi^*, 0 \le \Omega^*$, 使得

$$\begin{cases} 2R + Z^*R - \lambda_1^*\mathbf{1}^T - \mu^* I - \Phi^* = 0, & (4.20a)\\ X^*R - \mathbf{1}\left(\lambda_2^*\right)^T + \Phi^* - \Omega^* = 0, & (4.20b)\\ X^*\mathbf{1} = \rho, \mathrm{tr}(X^*) = 0, & (4.20c)\\ \left(Z^*\right)^T\mathbf{1} = \rho, Z^* \ge 0, & (4.20d)\\ \Omega^* \ge 0, & (4.20e)\\ X^* = Z^*, & (4.20f)\\ \Omega^* \circ Z^* = 0. & (4.20g) \end{cases}$$

注意在第k次迭代, X^{k+1}是问题(4.3)的解, 因此存在拉格朗日乘子$\lambda_1^{k+1}, \mu^{k+1}$使得

$$\begin{cases} 2R + Z^kR - \Phi^k + \beta(X^{k+1} - Z^k) - \lambda_1^{k+1}\mathbf{1}^T - \mu^{k+1}I = 0, & (4.21a)\\ X^{k+1}\mathbf{1} = \rho, \mathrm{tr}(X^{k+1}) = 0, & (4.21b) \end{cases}$$

其中等式(4.21a),(4.21b)分别对应于等式(4.20a),(4.20c). 特别地, 若使用更新策略(4.18),

$$\begin{aligned} &2R + Z^kR - \Phi^k + \beta(X^{k+1} - Z^k) - \lambda_1^{k+1}\mathbf{1}^T - \mu^{k+1}I\\ &\quad = 2R + Z^{k+1}R - \Phi^{k+1} - \lambda_1^{k+1}\mathbf{1}^T - \mu^{k+1}I + (Z^{k+1} - Z^k)(\beta I - R), \end{aligned}$$

这说明我们必须多加关注$(Z^{k+1} - Z^k)(\beta I - R)$. 若使用更新策略(4.19), 则

$$\begin{aligned} &2R + Z^kR - \Phi^k + \beta(X^{k+1} - Z^k) - \lambda_1^{k+1}\mathbf{1}^T - \mu^{k+1}I\\ &\quad = R + Z^{k+1}R - \Phi^{k+1} - \lambda_1^{k+1}\mathbf{1}^T - \mu^{k+1}I + (Z^{k+1} - Z^k)(\beta I - R) + (1-\alpha)\beta(X^{k+1} - Z^{k+1}), \end{aligned}$$

这说明我们还需要多考虑$(1-\alpha)\beta(X^{k+1} - Z^{k+1})$. 而对于$Z$子问题, Z^{k+1}是问题(4.9)的解, 于是存在拉格朗日乘子$\lambda_2^{k+1}, \Omega^{k+1} \ge 0$使得

$$\begin{cases} X^{k+1}R - \Phi^k - \beta(X^{k+1} - Z^{k+1}) - \mathbf{1}\left(\lambda_2^{k+1}\right)^T - \Omega^{k+1} = 0, & (4.22a)\\ \left(Z^{k+1}\right)^T\mathbf{1} = \rho, Z^{k+1} \ge 0, & (4.22b)\\ \Omega^{k+1} \ge 0, & (4.22c)\\ \Omega^{k+1} \circ Z^{k+1} = 0, & (4.22d) \end{cases}$$

其中等式(4.22a)-(4.22d)分别对应于等式(4.20b),(4.20d),(4.20e),(4.20g). 若使用更新策略(4.18), 则易知$\left(Z^{k+1},\lambda_2^{k+1},\Omega^{k+1},\Phi^{k+1}\right)$自动满足KKT条件. 事实上,

$$\begin{aligned}X^{k+1}R-\Phi^k-\beta(X^{k+1}-Z^{k+1})-\mathbf{1}\left(\lambda_2^{k+1}\right)^T-\Omega^{k+1}\\ =X^{k+1}R-\Phi^{k+1}-\mathbf{1}\left(\lambda_2^{k+1}\right)^T-\Omega^{k+1}.\end{aligned}$$

若使用更新策略(4.19), 则有

$$\begin{aligned}X^{k+1}R-\Phi^k-\beta(X^{k+1}-Z^{k+1})-\mathbf{1}\left(\lambda_2^{k+1}\right)^T-\Omega^{k+1}\\ =X^{k+1}R-\Phi^{k+1}-\mathbf{1}\left(\lambda_2^{k+1}\right)^T-\Omega^{k+1}-(1-\alpha)\beta(X^{k+1}-Z^{k+1}).\end{aligned}$$

于是再次地, 我们需考虑$(1-\alpha)\beta(X^{k+1}-Z^{k+1})$.

总之, KKT违反度可用

$$t^{k+1}\triangleq\|X^{k+1}-Z^{k+1}\|_\infty,\quad s^{k+1}\triangleq\|(Z^{k+1}-Z^k)(\beta I-R)\|_\infty. \tag{4.23}$$

度量. 我们分别称t^{k+1},s^{k+1}为原始残差和对偶残差. 因此, 我们可在以下情形终止内循环:

情形 1 t^{k+1},s^{k+1}都足够小; 或

情形 2 对某个$p^{k+1}\in(0,1)$, $p^{k+1}s^{k+1}+\left(1-p^{k+1}\right)t^{k+1}$足够小. 其中$p^{k+1}$反映了我们主观上对两类残差所赋的权值. 默认情形下, $p^{k+1}=0.5$.

4.6 完整算法

我们使用

$$E^{k+1}=(1-p^{k+1})t^{k+1}+p^{k+1}s^{k+1}, \tag{4.24}$$

作为KKT违反度.

算法 2 求解问题(2.3)的ADMM算法

输入: 初始值X^0,Z^0,Φ^0, 初始惩罚因子β^0, 容忍限ϵ, 初始化迭代数$k=0$, 初始化对偶残差和原始残差$s^0:=1,t^0:=1$, 松弛因子$\alpha>0$(默认值为1), 初始化权重$p^0\in(0,1)$.

输出: X^k,Z^k,Φ^k

1: $E^k=(1-p^k)t^k+p^ks^k$;
2: **while** $E^k>\epsilon$ **do**
3: *求解X子问题
4: 由公式(4.8)计算$\mu^{k+1},\lambda_1^{k+1}$;
5: 由公式(4.6)计算X^{k+1};
6: *求解Z子问题
7: 使用MATLAB内置函数`quadprog()`求解列子问题(4.15)得到Z^{k+1};
8: *更新乘子
9: $\Phi^{k+1}=\Phi^k-\alpha\beta^k(X^{k+1}-Z^{k+1})$;
10: *更新残差与各因子
11: 由公式(4.23)计算原始和对偶残差t^{k+1},s^{k+1};

12: 更新β^k得到β^{k+1};
13: 更新p^k得到p^{k+1};
14: 由公式(4.24)更新KKT违反度得到E^{k+1};
15: $k := k + 1$;
16: **end while**

5 收敛性分析

下面的定理5.1表明, 在迭代序列收敛且满足一定条件时, 算法2收敛到问题(2.3)的稳定点.

定理 5.1. 假设算法2每一步X, Z问题均精确求解, 且产生的迭代序列$\{X^k\}, \{Z^k\}, \{\Phi^k\}$分别收敛到$X^*, Z^*, \Phi^*$, 满足$X^* = Z^*$. 则$(X^*, Z^*, \Phi^*)$为问题(2.3)的稳定点.

证明. 回顾KKT条件(4.20), 我们只需要证明如下2条, 便能由它们的KKT条件和$X^* = Z^*$证明定理.

1. $X^* \in \arg\min_{X\mathbf{1}=\rho, \mathrm{tr}(X)=0} f(X, Z^*) - \langle \Phi^*, X - Z^* \rangle$.
2. $Z^* \in \arg\min_{Z^T\mathbf{1}=\rho, Z\geq 0} f(X^*, Z) - \langle \Phi^*, X^* - Z \rangle$.

由于X子问题精确求解, 所以

$$X^{k+1} = \arg\min_{X\mathbf{1}=\rho, \mathrm{tr}(X)=0} f(X, Z^k) - \langle \Phi^k, X - Z^k \rangle + \frac{\beta}{2}\|X - Z^k\|_F^2,$$

且

$$\begin{aligned} & f(X, Z^k) - \langle \Phi^k, X - Z^k \rangle + \frac{\beta}{2}\|X - Z^k\|_F^2 \\ &= \frac{\beta}{2}\|X\|_F^2 - \langle X, \beta Z^k - Z^k R - 2R + \Phi^k \rangle + \frac{\beta}{2}\|Z^k\|_F + \langle \Phi^k, Z^k \rangle \\ &= \frac{\beta}{2}\left\| X - \frac{1}{\beta}(\beta Z^k - Z^k R - 2R + \Phi^k) \right\|_F^2 + C, \end{aligned}$$

其中C相对于X是常数. 考虑到集合$\{X : X\mathbf{1} = \rho, \mathrm{tr}(X) = 0\}$的凸性, 根据文[52], 我们得出

$$\left\langle \frac{1}{\beta}(\beta Z^k - Z^k R - 2R + \Phi^k) - X^{k+1}, X - X^{k+1} \right\rangle \leq 0 \tag{5.1}$$

对任何$X : X\mathbf{1} = \rho, \mathrm{tr}(X) = 0$都是成立的. 由于

$$\begin{aligned} f(X, Z^k) - f(X^{k+1}, Z^k) &= \big(2\langle X, R \rangle + \langle Z^k, XR \rangle\big) - \big(2\langle X^{k+1}, R \rangle + \langle Z^k, X^{k+1}R \rangle\big) \\ &= \langle 2R + Z^k R, X - X^{k+1} \rangle, \end{aligned}$$

所以不等式(5.1)等价于

$$f(X, Z^k) - f(X^{k+1}, Z^k) - \left\langle \Phi^k - \beta(X^{k+1} - Z^k), X - X^{k+1} \right\rangle \geq 0.$$

两边取极限$k \to \infty$, 我们有

$$f(X, Z^*) - f(X^*, Z^*) - \langle \Phi^*, X - X^* \rangle \geq 0.$$

利用$X^* = Z^*$, 我们就完成了第1条的证明:

$$f(X, Z^*) - \langle \Phi^*, X - Z^* \rangle \geq f(X^*, Z^*) - \langle \Phi^*, X^* - Z^* \rangle.$$

第2条的证明过程类似.

6 数值实验

我们将算法2用于一系列随机生成的问题上. 我们随机选取初值$X^0, Z^0, \Phi^0, \Omega^0$并将它们储存起来, 以方便比较不同条件下的结果. 以下所有的数值实验都在MATLAB R2019a中完成, 运行环境为Windows 10 操作系统(64位), Intel Core i7-8550@CPU 1.80 GHz 1.99GHz, RAM: 8 GB. 以上随机生成均基于MATLAB内置函数`randn()`和`abs()`.

6.1 随机生成问题

我们分别在随机生成的小型问题和大型问题上测试了算法2. 小型问题以$n = 3, 4, 5$各一问题为代表, 大型问题则以$n = 20, 30, 40$各一问题为代表. 如不加说明, 小型问题中算法2中的常量取法为

$$\alpha = 1, \quad \beta^k \equiv 10^3, \quad \epsilon = 10^{-8}, \quad p^k \equiv 0.5,$$

而大型问题中常量取法为

$$\alpha = 1, \quad \beta^k \equiv 10^4, \quad \epsilon = 10^{-6}, \quad p^k \equiv 0.5.$$

计算结果可见表6.1-6.6.

表 6.1: $n = 3$, 不同的松弛因子

α	$n = 3$			
	迭代数	所耗时间(s)	KKT违反度	目标值
0.1	**452**	**0.0072**	3.40×10^{-9}	**1.1722**
0.2	491	0.0079	8.45×10^{-9}	**1.1722**
0.3	510	0.0088	7.07×10^{-9}	**1.1722**
0.4	517	0.0098	$\mathbf{2.86\times10^{-9}}$	**1.1722**
0.5	503	0.0080	7.45×10^{-9}	**1.1722**
0.6	534	0.0103	7.79×10^{-9}	**1.1722**
0.7	538	0.0093	5.69×10^{-9}	**1.1722**
0.8	525	0.0082	3.24×10^{-9}	**1.1722**
0.9	548	0.0105	8.97×10^{-9}	**1.1722**
1.0	560	0.0097	8.02×10^{-9}	**1.1722**

表格包含如下内容: 总迭代次数(迭代数)、CPU所耗时间(所耗时间(s))、KKT违反度、目标值. 表格中每列的最佳值用粗体显式. 为更好地展现计算得到的矩阵变量, 我们使用MATLAB内置函数`imagesc()`画出矩阵图像, 结果可见图6.1-6.6.

图中不同颜色代表了不同的取值, 颜色相近代表取值相近.

典型的收敛曲线如图6.7所示. 其中第一行为目标函数值, 第二行为KKT违反度, 第三行为增广拉格朗日函数值. 其中第二行的y轴数据以10为底. 从图中我们发现, KKT违反度并不是一直下降,

表 6.2: $n = 4$, 不同的松弛因子

α	$n = 4$			
	迭代数	所耗时间(s)	KKT违反度	目标值
0.1	1065	0.0343	4.51×10^{-9}	**4.4934**
0.2	**1034**	**0.0324**	8.04×10^{-9}	**4.4934**
0.3	1046	**0.0324**	3.77×10^{-9}	**4.4934**
0.4	1051	0.0329	9.32×10^{-9}	**4.4934**
0.5	1064	0.0327	4.71×10^{-9}	**4.4934**
0.6	1094	0.0328	4.67×10^{-9}	**4.4934**
0.7	1077	0.0356	3.27×10^{-9}	**4.4934**
0.8	1109	0.0351	7.44×10^{-9}	**4.4934**
0.9	1112	0.0367	5.31×10^{-9}	**4.4934**
1.0	1120	0.0375	$\mathbf{1.56\times10^{-9}}$	**4.4934**

表 6.3: $n = 5$, 不同的松弛因子

α	$n = 5$			
	迭代数	所耗时间(s)	KKT违反度	目标值
0.1	1003	0.0477	3.30×10^{-9}	**2.7741**
0.2	996	0.0503	8.45×10^{-9}	**2.7741**
0.3	981	0.0533	4.57×10^{-9}	**2.7741**
0.4	936	0.0462	8.22×10^{-9}	**2.7741**
0.5	902	0.0419	9.47×10^{-9}	**2.7741**
0.6	958	0.0462	1.00×10^{-9}	**2.7741**
0.7	923	0.0498	$\mathbf{8.35\times10^{-10}}$	**2.7741**
0.8	**815**	**0.0406**	8.89×10^{-9}	**2.7741**
0.9	891	0.0462	2.57×10^{-9}	**2.7741**
1.0	860	0.0438	9.54×10^{-9}	**2.7741**

表 6.4: $n = 20$, 不同的松弛因子

α	$n = 20$			
	迭代数	所耗时间(s)	KKT违反度	目标值
0.1	131061	45.8773	6.23×10^{-7}	**9.8592**
0.2	88223	30.3214	9.33×10^{-7}	9.8692
0.3	80565	27.7640	$\mathbf{4.97\times10^{-7}}$	9.8692
0.4	80562	28.4144	7.41×10^{-7}	9.8692
0.5	79925	28.3376	6.39×10^{-7}	9.8692
0.6	80059	28.0364	6.58×10^{-7}	9.8692
0.7	79251	**27.5997**	9.18×10^{-7}	9.8692
0.8	79245	27.6568	7.52×10^{-7}	9.8692
0.9	**78652**	27.6214	5.91×10^{-7}	9.8692
1.0	79052	28.2271	5.02×10^{-7}	9.8692

表 6.5: $n = 30$, 不同的松弛因子

α	$n = 30$			
	迭代数	所耗时间(s)	KKT违反度	目标值
0.1	117886	74.7150	9.74×10^{-7}	26.2298
0.2	69365	42.6729	8.60×10^{-7}	25.8871
0.3	70989	43.2812	8.88×10^{-7}	25.8828
0.4	**65533**	**40.0219**	6.46×10^{-7}	25.8757
0.5	122029	76.4424	9.28×10^{-7}	25.6165
0.6	112336	82.8559	9.63×10^{-7}	25.6165
0.7	112002	70.1342	8.81×10^{-7}	25.6165
0.8	109121	67.5131	7.99×10^{-7}	25.6165
0.9	113655	70.1853	8.83×10^{-7}	25.6165
1.0	207788	129.1006	$\mathbf{6.00\times10^{-7}}$	**24.9461**

装 订 线

表 6.6: $n = 40$, 不同的松弛因子

α	$n = 40$			
	迭代数	所耗时间(s)	KKT违反度	目标值
0.1	198867	223.6596	9.01×10^{-7}	**18.6029**
0.2	**109602**	135.3309	8.64×10^{-7}	19.3505
0.3	146723	174.2690	6.63×10^{-7}	19.1569
0.4	118460	143.4190	9.68×10^{-7}	19.0929
0.5	111804	**131.5869**	9.94×10^{-7}	19.0929
0.6	132657	163.7255	6.34×10^{-7}	19.0610
0.7	124324	149.4135	8.06×10^{-7}	19.0523
0.8	129025	148.9755	6.99×10^{-7}	19.0610
0.9	128645	152.9072	$\mathbf{5.65\times10^{-7}}$	19.0610
1.0	151285	177.2822	5.82×10^{-7}	19.0920

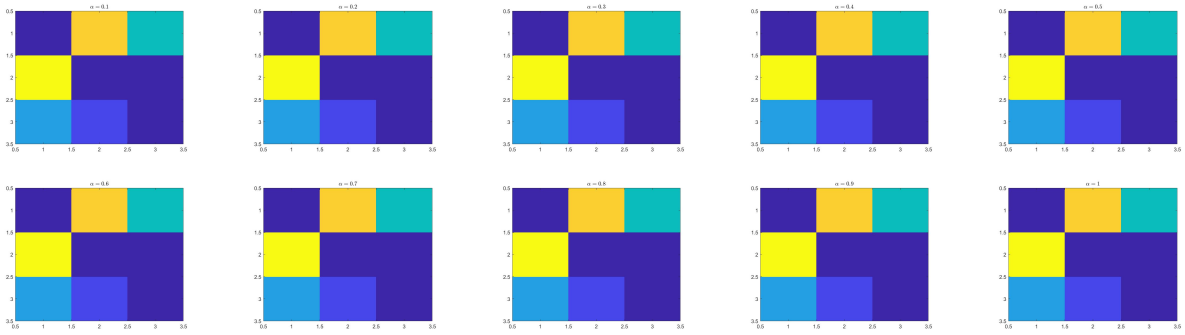


图 6.1: $n = 3$, 不同松弛因子

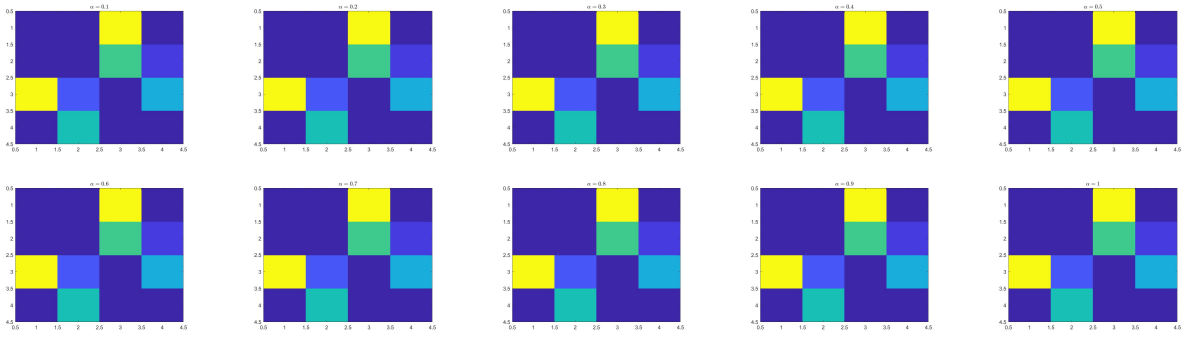


图 6.2: $n = 4$, 不同松弛因子

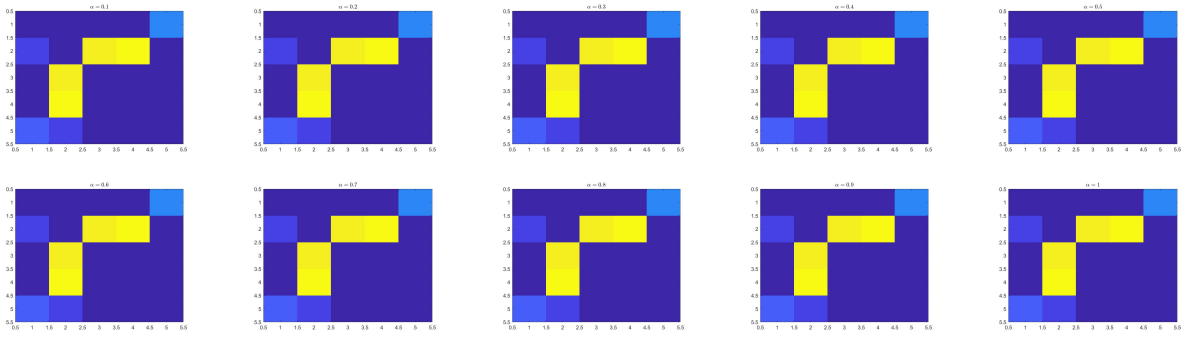


图 6.3: $n = 5$, 不同松弛因子

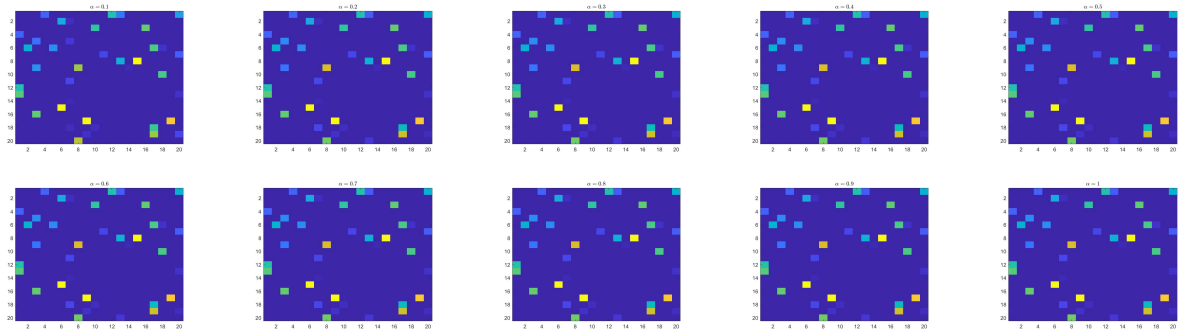


图 6.4: $n = 20$, 不同松弛因子

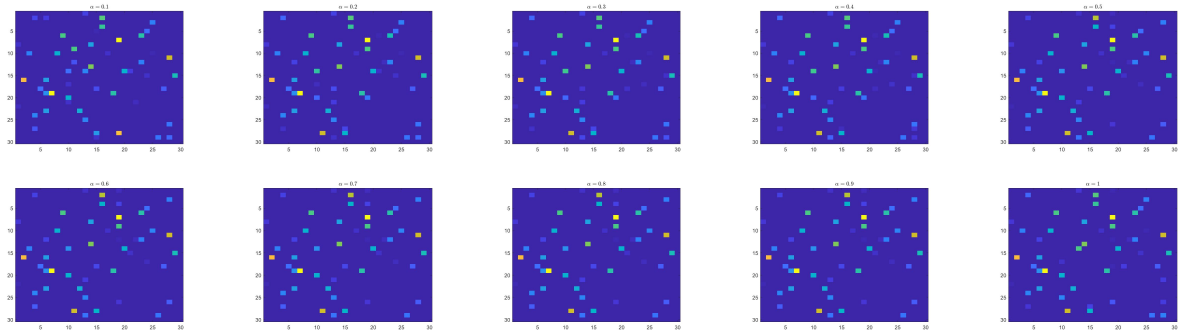


图 6.5: $n = 30$, 不同松弛因子

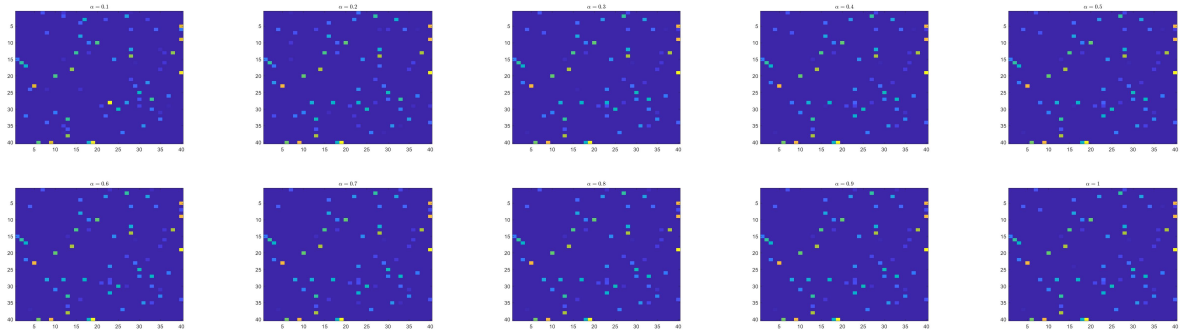
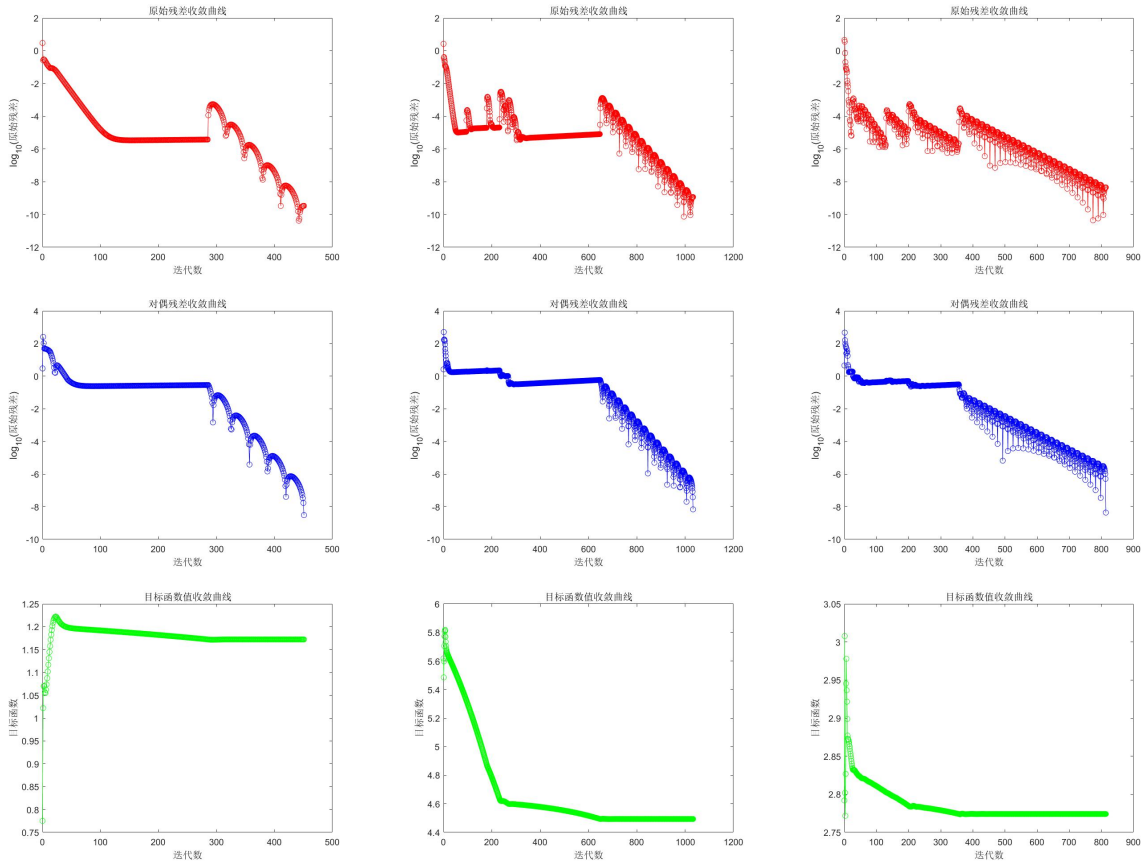


图 6.6: $n = 40$, 不同松弛因子



(a) $\alpha = 0.1, n = 3$ (b) $\alpha = 0.2, n = 4$ (c) $\alpha = 0.8, n = 5$

图 6.7: 收敛曲线

而是在不断抖动. 但尽管如此, 抖动形成的"山峰"一次比一次低, 最终满足停机准则. 这表明残差可能是以子列趋于0而不是整体收敛到0.

6.2 松弛因子α的选取

我们令$\alpha = 0.1, 0.2, \ldots, 1.0$, 而保持其他参数为默认值. 我们不考虑大于1的取值, 因为这可能导致算法不收敛.

从表6.1-6.6中我们可归纳出以下结论:

- 不同的松弛因子可能会影响目标值.
- 松弛因子对小型问题不会有多大影响, 但却十分影响大型问题.

我们应当指出, 对于算法的评价不能仅限制在迭代数和消耗的时间上. 尤其是我们的问题非凸, 可能有众多局部极小点. 因此, 我们有必要比较解的形式. 同样地, 我们使用MATLAB内置函数`imagesc()`画出不同α得到的解的图像. 见图6.1-6.6.

显然, 图6.1-6.4中的子图大多是相同的, 而图6.5,6.6中的子图则有所差异. 进一步, 我们将$n = 30, 40$的停机准则改为$\epsilon = 10^{-9}, 10^{-10}$, 重新运行程序. 最终得到的结果是一样的. 这种现象说明算法很可能陷入了局部极小点, 而非凸性随着维数的升高会变得愈发明显.

6.3 惩罚因子β的选取

ADMM算法中, 我们需要构造增广拉格朗日函数. 这就引入了额外的惩罚因子β. 对于两块可分离的凸问题, 文[6]中表明算法的收敛并不依赖于β的取值. 但我们的问题是非凸双线性的, 因此β的大小是否影响算法结果是值得讨论的. 我们选取不同的惩罚因子β进行数值实验. 在实际实验中, 小型问题上, 我们令

$$\beta = 10^2, 10^3, 10^4, 10^5,$$

大型问题上, 我们令

$$\begin{cases} \beta = 10^3, 10^4, 10^5, & n = 20, \\ \beta = 10^4, 10^5, & n = 30. \end{cases}$$

实验时保持其他参数为默认值. 实验结果可见表6.7-6.11.

表 6.7: $n = 3$, 不同惩罚因子

β	$n = 3$			
	迭代数	所耗时间(s)	KKT违反度	目标值
100	**224**	**0.0043**	5.46×10^{-9}	**1.1722**
1000	560	0.0095	8.02×10^{-9}	**1.1722**
10000	3998	0.0647	$\mathbf{4.92\times10^{-9}}$	**1.1722**
100000	38377	0.5818	7.68×10^{-9}	**1.1722**

表 6.8: $n = 4$, 不同惩罚因子

β	$n = 4$			
	迭代数	所耗时间(s)	KKT违反度	目标值
100	**601**	**0.0195**	8.28×10^{-9}	**4.4934**
1000	1120	0.0369	$\mathbf{1.56\times10^{-9}}$	**4.4934**
10000	7458	0.2249	8.59×10^{-9}	**4.4934**
100000	70836	2.1851	7.55×10^{-9}	**4.4934**

表 6.9: $n = 5$, 不同惩罚因子

β	$n = 5$			
	迭代数	所耗时间(s)	KKT违反度	目标值
100	969	0.0465	5.00×10^{-9}	**2.7741**
1000	**860**	**0.0384**	9.54×10^{-9}	**2.7741**
10000	3857	0.1698	$\mathbf{4.64\times10^{-9}}$	**2.7741**
100000	33470	1.4344	9.47×10^{-9}	**2.7741**

表 6.10: $n = 20$, 不同惩罚因子

β	$n = 20$			
	迭代数	所耗时间(s)	KKT违反度	目标值
1000	**12122**	**4.1522**	9.95×10^{-7}	**9.8692**
10000	79052	26.7695	$\mathbf{5.02\times10^{-7}}$	**9.8692**
100000	753429	255.0805	9.98×10^{-7}	**9.8692**

表 6.11: $n = 30$, 不同惩罚因子

β	$n = 30$			
	迭代数	所耗时间(s)	KKT违反度	目标值
10000	**207788**	**133.6154**	$\mathbf{6.00\times10^{-7}}$	**24.9461**
100000	2053406	1327.5987	7.08×10^{-7}	24.9542

每列的最佳值均以粗体显式.
从表中我们可归纳出:

- 惩罚因子需适当选取, 过大的惩罚因子均会减缓算法的收敛. 而过小的惩罚因子可能会使算法失效. 例如在$n = 30, \beta = 10^3$时, 迭代的过程中原始残差和对偶残差陷入了循环. 见图6.8.

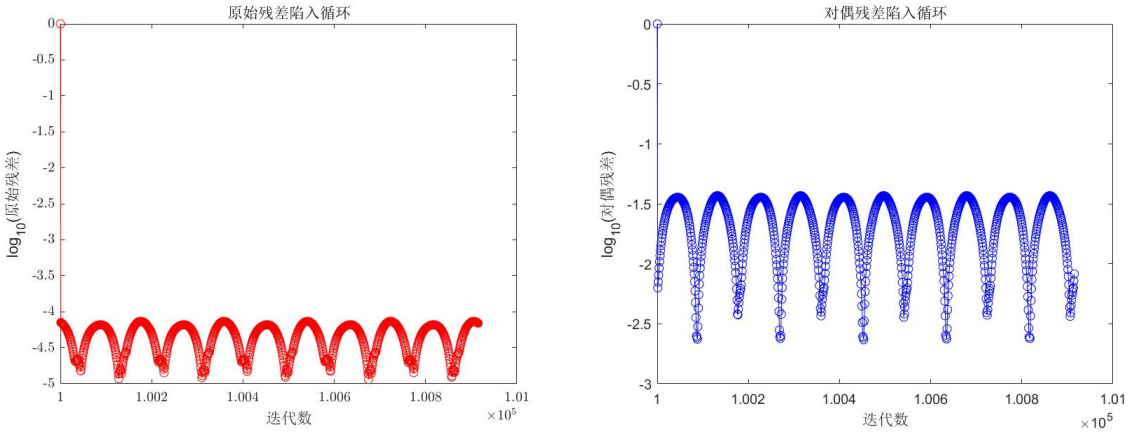


图 6.8: 残差陷入循环. 左: 原始残差; 右: 对偶残差

- 适宜惩罚因子的值与问题的维数有着正相关的关系, 即问题维数越大, 我们越应当选取更大的惩罚因子. 例如表6.7,6.8中的小型问题以$\beta = 100$为最佳, 而表6.9-6.11的数据则说明$\beta = 1000, 10000$更适合于较大型问题.

一种自适应的调整惩罚因子的方式为

$$\beta^{k+1} := \begin{cases} \tau^{\text{incr}}\beta^k, & \text{若} t^{k+1} > ms^{k+1}, \\ \beta^k/\tau^{\text{decr}}, & \text{若} s^{k+1} > mt^{k+1}, \\ \beta^k, & \text{其它}, \end{cases} \tag{6.1}$$

这里$m > 1, \tau^{\text{incr}} > 1, \tau^{\text{decr}} > 1$为参数. 其中$\tau^{\text{incr}}, \tau^{\text{decr}}$的选取时为了保持原始残差和对偶残差处在同一量级, 共同趋于0.

这种自适应的策略来自s^{k+1}的定义(4.23)和惩罚项的定义(4.2). 当原始残差相对于对偶残差较大时, 我们就增大惩罚因子在下一步迭代中使算法向可行性移动. 反之, 就应当减小惩罚因子, 因为对偶残差的定义中包括了β. 但在实验中我们发现, 两种残差总有一方率先迅速变小. 以上述策略更新惩罚因子很可能会使β的值上溢或下溢.

6.4 小型问题的特殊性

在实验过程中我们发现算法总是会在$n = 3, 4$的小型问题上失效. 其中有一共同的特征是, 两个残差其一保持非零正数而不继续下降, 而另一方则迅速下降趋于0. 这就是说, 我们的算法收敛到了不可行点. 为解决这样的问题, 我们曾引入松弛因子α, 但作用不大.

进一步的讨论表明导致算法失效的是问题本身的不可行性. 以$n = 3$为例. 考虑到约束, 可行的X必定有形式

$$X = \begin{pmatrix} 0 & \rho_1 + \rho_2 - \rho_3 - a & \rho_3 - \rho_2 + a \\ a & 0 & \rho_2 - a \\ \rho_1 - a & \rho_3 - \rho_1 + a & 0 \end{pmatrix},$$

其中参数a待定. 由于$X \geq 0$, 所以a的取值范围需要进一步讨论:

$$\begin{cases} 0 \leq a \leq \rho_1, \rho_2, \\ 0 \leq \rho_1 - a \leq \rho_1, \rho_3, \\ 0 \leq \rho_2 - a \leq \rho_2, \rho_3, \\ 0 \leq \rho_3 - \rho_1 + a \leq \rho_2, \rho_3, \\ 0 \leq \rho_3 - \rho_2 + a \leq \rho_1, \rho_3, \\ 0 \leq \rho_1 + \rho_2 - \rho_3 - a \leq \rho_1, \rho_2, \end{cases} \Leftrightarrow \begin{cases} 0 \leq a \leq \rho_1, \rho_2, \\ \rho_1 - \rho_3, 0 \leq a \leq \rho_1, \\ \rho_2 - \rho_3, 0 \leq a \leq \rho_2, \\ \rho_1 - \rho_3 \leq a \leq \rho_1 + \rho_2 - \rho_3, \rho_1, \\ \rho_2 - \rho_3 \leq a \leq \rho_1 + \rho_2 - \rho_3, \rho_2, \\ \rho_1 - \rho_3, \rho_2 - \rho_3 \leq a \leq \rho_1 + \rho_2 - \rho_3, \end{cases}$$

而这些等价于

$$\max(0, \rho_1 - \rho_3, \rho_2 - \rho_3) \leq a \leq \min(\rho_1, \rho_2, \rho_1 + \rho_2 - \rho_3). \tag{6.2}$$

于是由关系式(6.2), 问题(2.3)不可行当且仅当$\max(0, \rho_1 - \rho_3, \rho_2 - \rho_3) > \min(\rho_1, \rho_2, \rho_1 + \rho_2 - \rho_3)$. 经检验, 实验中算法失效的随机生成问题均是不可行的. 然而当n变得更大时, 所生成的问题基本是可行的. 问题的不可行性也可以用MATLAB的内置函数检测, 如`quadprog()`和`linprog()`.

6.5 随机初始化

实验表明算法2严重依赖于Z, Φ的初始化. $n = 10$的例子可见图6.9. 其中我们固定R, ρ, 选取$\alpha = 1, \epsilon = 10^{-9}$, 随机初始化$Z, \Phi$6次. 图6.9表明不同的初始化往往带来不同的结果.

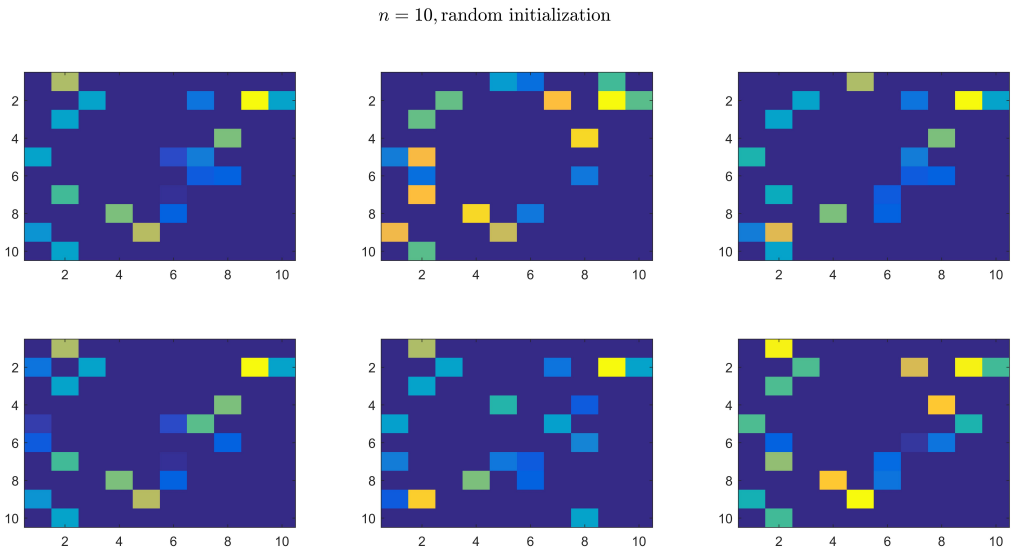


图 6.9: 随机初始化的影响

6.6 测试问题

我们设计了特殊的问题, 用以表明我们的算法的确可以达到全局最优解. 具体地说, 给定一数对$(p,q):p\neq q,\ p,q\in\{1,2,\ldots,n\}$, 定义$R$为

$$r_{ij}=\begin{cases}1, & i=p,j=q\text{或}i=q,j=p,\\ 0, & \text{其它}.\end{cases}$$

设$\rho=\mathbf{1}$. 展开问题(2.3)中的目标函数, 我们有

$$\begin{aligned}\min_{X}\quad & 2x_{pq}+2\sum_{i}x_{ip}x_{iq}\\ \text{s.t.}\quad & X\mathbf{1}=\rho,X^T\mathbf{1}=\rho,X\geq 0,\mathrm{tr}(X)=0.\end{aligned}\tag{6.3}$$

问题(6.3)的目标函数只与X的第p和第q列有关. 事实上问题(6.3)的最优值为0. 首先目标函数有个天然的下界0, 其次我们可以构造出可行点使其函数值就是0. 例如,

$$X=\begin{array}{c}\\ p\\ \\ \\ q\\ \\ \\ \end{array}\begin{array}{c}\begin{array}{cccc}p & & q & \end{array}\\ \begin{pmatrix}1 & & 0 & \\ 0 & & 1 & \\ 0 & & 0 & \\ \vdots & * & \vdots & * \\ 0 & & 0 & \\ 0 & & 0 & \\ 0 & & 0 & \end{pmatrix}\end{array}$$

我们对$n=5,10,15,20$的问题测试了数对$(p,q)=(3,4)$. 实验时令$\alpha=1,\beta=10^3,\epsilon=10^{-8}$. 结果可见表6.12.

表 6.12: 测试问题, $n=5,10,15,20$

n	迭代数	所耗时间(s)	KKT违反度	目标值
5	3187	0.1536	3.00×10^{-9}	-1.44×10^{-11}
10	1447	0.1766	1.65×10^{-9}	-4.94×10^{-15}
15	2243	0.3284	2.30×10^{-9}	6.54×10^{-13}
20	2030	0.3299	4.53×10^{-9}	-8.07×10^{-13}

注意表中目标值一栏的负值可能源于计算时的舍入误差和不完全收敛.

6.7 与求解非凸二次规划的算法比较

我们在$n = 3, 4, 5, 20, 30, 40, 50, 60, 70, 80$的随机问题上比较算法2与现有的求解非凸二次规划的算法. 为此, 我们先将问题(2.1)化成向量形式, 即

$$\begin{aligned} \min_{X,Y} \quad & (\mathrm{vec}(X) + \mathrm{vec}(Y))^T \mathrm{vec}(R) + \mathrm{vec}(X)^T (R \otimes I) \mathrm{vec}(Y) \\ \text{s.t.} \quad & \left(\mathbf{1}^T \otimes I\right) \mathrm{vec}(X) = \left(I \otimes \mathbf{1}^T\right) \mathrm{vec}(X) = \rho, \mathrm{vec}(I)^T \mathrm{vec}(X) = 0, \mathrm{vec}(X) \geq 0, \\ & \left(\mathbf{1}^T \otimes I\right) \mathrm{vec}(Y) = \left(I \otimes \mathbf{1}^T\right) \mathrm{vec}(Y) = \rho, \mathrm{vec}(I)^T \mathrm{vec}(Y) = 0, \mathrm{vec}(Y) \geq 0. \end{aligned} \tag{6.4}$$

类似地, 有问题(2.2)的向量形式:

$$\begin{aligned} \min_{X} \quad & 2\mathrm{vec}(X)^T \mathrm{vec}(R) + \mathrm{vec}(X)^T (R \otimes I) \mathrm{vec}(X) \\ \text{s.t.} \quad & \left(\mathbf{1}^T \otimes I\right) \mathrm{vec}(X) = \left(I \otimes \mathbf{1}^T\right) \mathrm{vec}(X) = \rho, \mathrm{vec}(I)^T \mathrm{vec}(X) = 0, \mathrm{vec}(X) \geq 0. \end{aligned} \tag{6.5}$$

之后使用MATLAB内置函数`fmincon()`求解问题(6.4)和问题(6.5). 这里我们使用的算法为SQP, 并将算法停机准则之一`ConstraintTolerance`设置为上文提到的默认值, 初始点与算法2相同. 求解问题(6.5)的结果可见表6.13. 在使用算法2求解$n = 50, 60, 70, 80$的问题时, 我们在算法2中设置的惩罚

表 6.13: 使用现成算法

$n = 3$	所耗时间(s)	目标值	$n = 4$	所耗时间(s)	目标值
算法2	0.0097	**1.1722**	算法2	0.0375	**4.4934**
SQP	**0.0033**	**1.1722**	SQP	**0.0039**	**4.4934**
$n = 5$	所耗时间(s)	目标值	$n = 20$	所耗时间(s)	目标值
算法2	0.0438	**2.7741**	算法2	28.2271	**9.8692**
SQP	**0.0048**	**2.7741**	SQP	**0.6388**	9.9096
$n = 30$	所耗时间(s)	目标值	$n = 40$	所耗时间(s)	目标值
算法2	129.1006	**24.9461**	算法2	177.2822	**19.0920**
SQP	**12.0369**	26.5668	SQP	**42.4949**	21.6995
$n = 50$	所耗时间(s)	目标值	$n = 60$	所耗时间(s)	目标值
算法2	195.8105	**42.8754**	算法2	507.7331	**30.3166**
SQP	**132.2582**	46.8143	SQP	**472.3746**	33.0913
$n = 70$	所耗时间(s)	目标值	$n = 80$	所耗时间(s)	目标值
算法2	**1440.9720**	30.2839	算法2	**2275.5156**	**40.6460**
SQP	2174.3117	**30.1568**	SQP	3589.5316	43.1043

因子分别为

$$\begin{cases} \beta = 10^4, & n = 50, 60, \\ \beta = 2 \times 10^4, & n = 70, 80. \end{cases}$$

从表6.13可知, 从所耗时间上来看, 对于小型问题, 直接使用SQP要比我们所设计的算法2更好. 但这一优势随着问题规模的增大将逐渐消失, 例如在$n = 70, 80$时, 算法2所用时间显著少于SQP. 这主要是因为随着问题规模增大, SQP所需的计算量增长过快. 而针对问题问题(2.3)特殊结构而设计的算法2相较之在大型问题上具有一定优势. 算法2求解问题(2.3)、SQP求解问题(6.5)和SQP求解问题(6.4)三者运行时间比较的形象阐释可见图6.10. 其中“SQP-X”代表SQP求解问题(6.5), “SQP-XY”

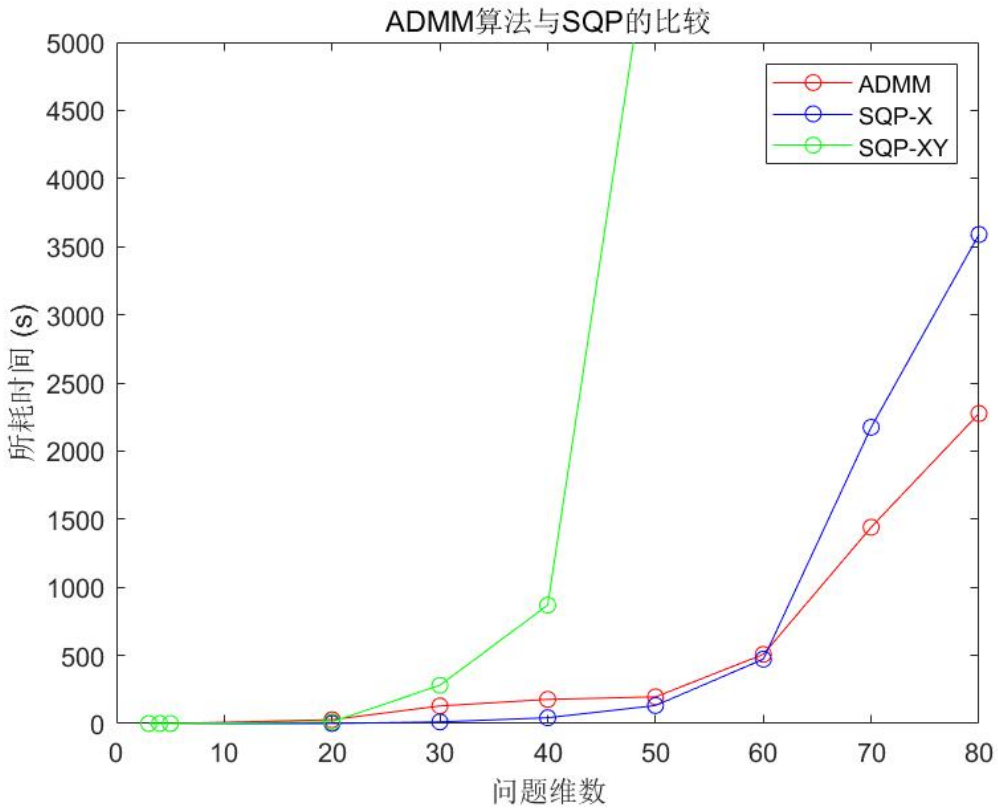


图 6.10: 算法2和SQP运行时间对比

代表SQP求解问题(6.4). 它们在最终的目标函数值上并不存在绝对的优劣关系, 可见6.13的“目标值”列. 值得说明的是, 由于迭代过程不同, 两种算法得到的函数值并不一样; 算法2在“大多数情形下”得到的目标值会更好. 而不经假设1直接将SQP应用于问题(6.4)上需要的运行时间增长得更为迅猛.

我们同样测试了积极集法. 结论是, 积极集法同样在小型问题上更加迅捷(甚至好于SQP), 但在大型问题上表现不佳(差于SQP, 且运行时间增长比SQP更快). 由于积极集法基于对最优积极集的不断估计, 因此它需要不断遍历约束. 所以这样的方法不适用于约束过多的问题.

比较算法2、SQP和积极集法, 我们得出: 对于小型问题, 积极集法和SQP最为便捷; 对于大型问题, 我们所设计的算法2更具优势. 算法2的缺点在于, 我们需要根据问题的维度不断重新设置惩罚因子β.

7 总结与未来工作展望

本文针对双线性规划问题(1.2), 充分考虑其特殊结构, 设计了相应的ADMM算法. 之后我们证明了在迭代序列收敛的前提下, 算法收敛到问题(2.3)的稳定点. 最后, 我们在多个随机生成的问题上进行实验, 验证了算法的有效性, 并讨论了多个人工给定因子(惩罚因子β、松弛因子α)对算法效果的影响. 我们还将设计的算法与直接求解非凸二次规划的SQP算法和积极集法进行了比较, 得出了我们的算法在大型问题上更具优势的结论.

未来我们的工作将集中在以下几点:

1. 证明假设1, 即问题(2.1)与问题(2.3)的等价性.
2. 设计求解Z子问题(4.9)更加高效、精确的算法.
3. 证明算法2在较弱条件下(比如无需迭代序列收敛的前提)的收敛性.
4. 对松弛因子、惩罚因子的进一步讨论. 我们在第6节中讨论了提前固定不同的松弛因子、惩罚因子对算法的影响. 但设计一些自适应调节松弛因子、惩罚因子的策略也是极有意义的.
5. 初始点对算法结果的影响. 显然, 问题(1.2)是非凸的二次规划, 因此必定有众多局部极小点. 尽管对于一般问题设计全局优化的算法并不可行, 但对于特殊的问题, 我们可以去证明算法的确能收敛到全局解. 影响求解非凸问题算法的一个重要因素就是初始点的选取.
6. 与现有双线性规划算法进行比较. 我们在第1节中对现有的双线性规划算法做了简单介绍, 但我们在实验(见第6节)中仅拿我们的算法与求解非凸二次规划的SQP算法和积极集法进行了比较. 相信更加丰富的实验比较能进一步说明我们所设计的算法的优越性. 同时, 我们也可以对比多个评价算法的指标, 除了运行时间, 还有计算复杂度等.
7. 子问题的并行求解. 在第4.3节中我们设计了针对Z子问题的算法, 高效地求解了子问题. 在更新Z的过程中, 列与列之间的计算是互不影响的. 利用并行计算, 我们能期望在求解时间上获得较大的改善.

参考文献

[1] Alarie S, Audet C, Jaumard B, et al. Concavity cuts for disjoint bilinear programming[J]. Mathematical Programming, 2001, 90(2): 373-398.

[2] Altman M. Bilinear programming[J]. BULLETIN DE L ACADEMIE POLONAISE DES SCIENCES-SERIE DES SCIENCES MATHEMATIQUES ASTRONOMIQUES ET PHYSIQUES, 1968, 16(9): 741-746.

[3] Audet C, Hansen P, Jaumard B, et al. A symmetrical linear maxmin approach to disjoint bilinear programming[J]. Mathematical Programming, 1999, 85(3): 573-592.

[4] Balinski M L. An algorithm for finding all vertices of convex polyhedral sets[J]. Journal of the Society for Industrial and Applied Mathematics, 1961, 9(1): 72-88.

[5] Bolte J, Sabach S, Teboulle M. Proximal alternating linearized minimization for nonconvex and nonsmooth problems[J]. Mathematical Programming, 2014, 146(1-2): 459-494.

[6] Boyd S, Parikh N, Chu E, et al. Distributed optimization and statistical learning via the alternating direction method of multipliers[J]. Foundations and Trends®in Machine learning, 2011, 3(1): 1-122.

[7] Cai X, Han D, Yuan X. On the convergence of the direct extension of ADMM for three-block separable convex minimization models with one strongly convex function[J]. Computational Optimization and Applications, 2017, 66(1): 39-73.

[8] Chen C, He B, Ye Y, et al. The direct extension of ADMM for multi-block convex minimization problems is not necessarily convergent[J]. Mathematical Programming, 2016, 155(1-2): 57-79.

[9] Chen C, Li M, Liu X, et al. Extended ADMM and BCD for nonseparable convex minimization models with quadratic coupling terms: convergence analysis and insights[J]. Mathematical Programming, 2019, 173(1-2): 37-77.

[10] Chen C, Shen Y, You Y. On the convergence analysis of the alternating direction method of multipliers with three blocks[C]//Abstract and Applied Analysis. Hindawi, 2013, 2013.

[11] Cui Y, Li X, Sun D, et al. On the convergence properties of a majorized alternating direction method of multipliers for linearly constrained convex optimization problems with coupled objective functions[J]. Journal of Optimization Theory and Applications, 2016, 169(3): 1013-1041.

[12] Davis D, Yin W. Convergence rate analysis of several splitting schemes[M]//Splitting methods in communication, imaging, science, and engineering. Springer, Cham, 2016: 115-163.

[13] Deng W, Yin W. On the global linear convergence of alternating direction methods[J]. Preprint, 2012.

[14] Ding X, Al-Khayyal F. Accelerating convergence of cutting plane algorithms for disjoint bilinear programming[J]. Journal of Global Optimization, 2007, 38(3): 421-436.

[15] Falk J E. A linear max—min problem[J]. Mathematical Programming, 1973, 5(1): 169-188.

[16] Forero P A, Cano A, Giannakis G B. Distributed clustering using wireless sensor networks[J]. IEEE Journal of Selected Topics in Signal Processing, 2011, 5(4): 707-724.

[17] Frieze A M. A bilinear programming formulation of the 3-dimensional assignment problem[J]. Mathematical Programming, 1974, 7(1): 376-379.

[18] Gabay D, Mercier B. A dual algorithm for the solution of nonlinear variational problems via finite element approximation[J]. Computers & Mathematics with Applications, 1976, 2(1): 17-40.

[19] Gallo G, ÜlkücüA. Bilinear programming: an exact algorithm[J]. Mathematical Programming, 1977, 12(1): 173-194.

[20] Gao X, Zhang S Z. First-order algorithms for convex optimization with nonseparable objective and coupled constraints[J]. Journal of the Operations Research Society of China, 2017, 5(2): 131-159.

[21] Ghadimi S, Lan G. Accelerated gradient methods for nonconvex nonlinear and stochastic programming[J]. Mathematical Programming, 2016, 156(1-2): 59-99.

[22] Ghadimi S, Lan G, Zhang H. Mini-batch stochastic approximation methods for nonconvex stochastic composite optimization[J]. Mathematical Programming, 2016, 155(1-2): 267-305.

[23] Glowinski R, Marroco A. Sur l'approximation, par éléments finis d'ordre un, et la résolution, par pénalisation-dualitéd'une classe de problèmes de Dirichlet non linéaires[J]. ESAIM: Mathematical Modelling and Numerical Analysis-Modélisation Mathématique et Analyse Numérique, 1975, 9(R2): 41-76.

[24] Goldfarb D, Ma S, Scheinberg K. Fast alternating linearization methods for minimizing the sum of two convex functions[J]. Mathematical Programming, 2013, 141(1-2): 349-382.

[25] Goldstein T, O'Donoghue B, Setzer S, et al. Fast alternating direction optimization methods[J]. SIAM Journal on Imaging Sciences, 2014, 7(3): 1588-1623.

[26] Han D, Yuan X, Zhang W, et al. An ADM-based splitting method for separable convex programming[J]. Computational Optimization and Applications, 2013, 54(2): 343-369.

[27] He B, Tao M, Yuan X. A splitting method for separable convex programming[J]. IMA Journal of Numerical Analysis, 2014, 35(1): 394-426.

[28] He B, Tao M, Yuan X. Alternating direction method with Gaussian back substitution for separable convex programming[J]. SIAM Journal on Optimization, 2012, 22(2): 313-340.

[29] He B , Yuan X . On the $O(1/n)$ Convergence Rate of the Douglas–Rachford Alternating Direction Method[J]. SIAM Journal on Numerical Analysis, 2012, 50(2):700-709.

[30] Hong M, Chang T H, Wang X, et al. A block successive upper bound minimization method of multipliers for linearly constrained convex optimization[J]. arXiv preprint arXiv:1401.7079, 2014.

[31] Hong M, Luo Z Q. On the linear convergence of the alternating direction method of multipliers[J]. Mathematical Programming, 2017, 162(1-2): 165-199.

[32] Hong M, Luo Z Q, Razaviyayn M. Convergence analysis of alternating direction method of multipliers for a family of nonconvex problems[J]. SIAM Journal on Optimization, 2016, 26(1): 337-364.

[33] Ibaraki T. Complementary programming[J]. Operations Research, 1971, 19(6): 1523-1529.

[34] Jiang B, Ma S, Zhang S. Alternating direction method of multipliers for real and complex polynomial optimization models[J]. Optimization, 2014, 63(6): 883-898.

[35] Konno H. A cutting plane algorithm for solving bilinear programs[J]. Mathematical Programming, 1976, 11(1): 14-27.

装 订 线

[36] Konno H. Bilinear Programming: Part II. Application of Bilinear Programming[R]. STANFORD UNIV CALIF DEPT OF OPERATIONS RESEARCH, 1971.

[37] Konno H. Maximization of a convex quadratic function under linear constraints[J]. Mathematical programming, 1976, 11(1): 117-127.

[38] Li M, Sun D, Toh K C. A convergent 3-block semi-proximal ADMM for convex minimization problems with one strongly convex block[J]. Asia-Pacific Journal of Operational Research, 2015, 32(04): 1550024.

[39] Liavas A P, Sidiropoulos N D. Parallel algorithms for constrained tensor factorization via alternating direction method of multipliers[J]. IEEE Transactions on Signal Processing, 2015, 63(20): 5450-5463.

[40] Lin T, Ma S, Zhang S. Iteration complexity analysis of multi-block ADMM for a family of convex minimization without strong convexity[J]. Journal of Scientific Computing, 2016, 69(1): 52-81.

[41] Lin T, Ma S, Zhang S. On the global linear convergence of the admm with multiblock variables[J]. SIAM Journal on Optimization, 2015, 25(3): 1478-1497.

[42] Lin T Y, Ma S Q, Zhang S Z. On the sublinear convergence rate of multi-block ADMM[J]. Journal of the Operations Research Society of China, 2015, 3(3): 251-274.

[43] Mangasarian O L. Equilibrium points of bimatrix games[J]. Journal of the Society for Industrial and Applied Mathematics, 1964, 12(4): 778-780.

[44] Mangasarian O L, Stone H. Two-person nonzero-sum games and quadratic programming[J]. Journal of Mathematical Analysis and applications, 1964, 9(3): 348-355.

[45] Mills H. Equilibrium points in finite games[J]. Journal of the Society for Industrial and Applied Mathematics, 1960, 8(2): 397-402.

[46] Monteiro R D C, Svaiter B F. Iteration-complexity of block-decomposition algorithms and the alternating direction method of multipliers[J]. SIAM Journal on Optimization, 2013, 23(1): 475-507.

[47] Nocedal J, Wright S. Numerical optimization[M]. Springer Science & Business Media, 2006.

[48] Pardalos P M, Vavasis S A. Quadratic programming with one negative eigenvalue is NP-hard[J]. Journal of Global Optimization, 1991, 1(1): 15-22.

[49] PeyréG, Cuturi M. Computational optimal transport[J]. Foundations and Trends®in Machine Learning, 2019, 11(5-6): 355-607.

[50] Razaviyayn M, Hong M, Luo Z Q. A unified convergence analysis of block successive minimization methods for nonsmooth optimization[J]. SIAM Journal on Optimization, 2013, 23(2): 1126-1153.

[51] Ritter K. A method for solving maximum-problems with a nonconcave quadratic objective function[J]. Probability Theory and Related Fields, 1966, 4(4): 340-351.

[52] Rockafellar R T. Convex analysis[M]. Princeton university press, 2015.

[53] Rosen J B. The gradient projection method for nonlinear programming. Part I. Linear constraints[J]. Journal of the society for industrial and applied mathematics, 1960, 8(1): 181-217.

[54] Santambrogio F. Optimal transport for applied mathematicians[J]. Birkäuser, NY, 2015, 55: 58-63.

[55] Scutari G, Facchinei F, Song P, et al. Decomposition by partial linearization: Parallel optimization of multi-agent systems[J]. IEEE Transactions on Signal Processing, 2014, 62(3): 641-656.

[56] Shen Y, Wen Z, Zhang Y. Augmented Lagrangian alternating direction method for matrix separation based on low-rank factorization[J]. Optimization Methods and Software, 2014, 29(2): 239-263.

[57] Sherali H D, Shetty C M. A finitely convergent algorithm for bilinear programming problems using polar cuts and disjunctive face cuts[J]. Mathematical Programming, 1980, 19(1): 14-31.

[58] Sherali A D, Shetty C M. The rectilinear distance location-allocation problem[J]. AIIE Transactions, 1977, 9(2): 136-143.

[59] Sun D L, Fevotte C. Alternating direction method of multipliers for non-negative matrix factorization with the beta-divergence[C]//2014 IEEE international conference on acoustics, speech and signal processing (ICASSP). IEEE, 2014: 6201-6205.

[60] Sun R, Luo Z Q, Ye Y. On the expected convergence of randomly permuted ADMM[J]. arXiv preprint arXiv:1503.06387, 2015.

[61] Tuy H. Concave programming under linear constraints[C]//Soviet Mathematics Doklady. 1964, 5: 1437-1440.

[62] Vaish H. Nonconvex programming with applications to production and location problems[D]. Georgia Institute of Technology, 1974.

[63] Vaish H, Shetty C M. A cutting plane algorithm for the bilinear programming problem[J]. Naval Research Logistics Quarterly, 1977, 24(1): 83-94.

[64] Vaish H, Shetty C M. The bilinear programming problem[J]. Naval Research Logistics Quarterly, 1976, 23(2): 303-309.

[65] Villani C. Optimal transport: old and new[M]. Springer Science & Business Media, 2008.

[66] Villani C. Topics in optimal transportation[M]. American Mathematical Soc., 2003.

[67] Wen Z, Goldfarb D, Scheinberg K. Block coordinate descent methods for semidefinite programming[M]//Handbook on semidefinite, conic and polynomial optimization. Springer, Boston, MA, 2012: 533-564.

[68] Wen Z, Peng X, Liu X, et al. Asset allocation under the basel accord risk measures[J]. arXiv preprint arXiv:1308.1321, 2013.

[69] Xu Y, Yin W, Wen Z, et al. An alternating direction algorithm for matrix completion with nonnegative factors[J]. Frontiers of Mathematics in China, 2012, 7(2): 365-384.

[70] Zhang R, Kwok J. Asynchronous distributed ADMM for consensus optimization[C]//International Conference on Machine Learning. 2014: 1701-1709.

[71] Zhang Y. An Alternating Direction Algorithm for Nonnegative Matrix Factorization[J]. 2010.

[72] Zhang Y. Convergence of a Class of Stationary Iterative Methods for Saddle Point Problems[J]. 2010.

[73] Zwart P B. Nonlinear programming: counterexamples to two global optimization algorithms[J]. Operations Research, 1973, 21(6): 1260-1266.

[74] 顾险峰. 最优传输理论在机器学习中的应用[J]. 中国人工智能学会通讯, 2017, 7(5): 1-13.

谢辞

谨在论文完成之际, 我要把我最真挚的谢意献给我的两位论文指导老师——我院殷俊锋教授和中国科学院数学与系统科学研究院的刘歆副研究员. 殷老师是我在同济大学数学科学学院的数值分析和计算方法任课老师, 是将我引入计算数学大门的人. 您渊博的学识在课堂上展现得淋漓尽致, 您独特的幽默感总能给课程注入活力. 您从不将学生的视野局限在课本知识, 反倒是推动我们自发地在学术的田野中开疆扩土. 在您的课堂里, 我们能了解到学术的最新动态, 感受到学术的魅力. 与我而言, 您不仅是一位严师, 更像是一位慈父. 您从不吝惜课堂以外的时间, 乐意与学生交流. 我也从您的口中学到了许多学术以外的道理. 您曾经向我解说国内外的经济局势, 提醒我要不时关注国家大事; 您曾经私信我长文, 为我讲述做人做事做学术的道理. 在我的心中, 您的形象早已超越了老师的存在. 刘老师是我未来五年在中科院的指导老师. 我从您的身上看到了学者的谦逊、师者的威严、父亲的慈爱、丈夫的体贴和跑者的坚守. 在访问您的短短两个多月时间我学到了太多. 您教会我如何写作汇报和论文、如何做一位合格的学生、如何遵守学术的严谨、如何妥善利用身边的资源以及如何保持一颗积极向上的心. 与您的每次交谈, 我都能收获受益终生的警句和观点. 您始终是一个给予的角色, 一心一意为学生构建美好的未来. 在您的帮助下, 我有幸作为志愿者参加了丝路数学中心的国际会议, 与国际优化领域的专家面对面接触, 获得书本之外的宝贵经历. 在与外国学者专家的长期交流后, 我的英语口语和写作能力均获得了极大的提升. 同时您也带我加入了您领衔的优化跑马组, 让我享受丰富的经验和咨询, 打造更健康的身体. 您给了我无数次机遇, 我必不负所期.

我还要特别感谢我院副院长潘生亮教授. 您是帮助我打下分析基础的恩师, 平常课上课下教会了我许多的道理. 同时我还要感谢我院院长许学军教授. 若没有您和潘老师, 我也不会有未来的中科院之旅. 我也感谢您的谆谆教诲. 记得去中科院面试时, 您提醒我不要忘记自己是同济数院人, 在外要有担当, 刻苦学习, 对内要懂得回馈母校, 永怀感恩之心.

我要感谢蒋志洪教授、陈滨副教授、周羚君副教授以及其他悉心教导过我的任课老师. 感谢你们长期以来的鼓励和帮助. 感谢(曾经)团委和负责学生工作的李扬帆老师、郁霞老师、赵盈老师和金文心老师, 我的三位班主任——王鹏教授(现在福建师范大学)、彭婧老师、尚培培副教授. 大学四年给你们添了不少麻烦, 而你们总是满怀热情地帮助我解决问题. 其中特别感谢李扬帆老师和彭婧老师. 大一一进数院我们就称呼李老师“帆姐”. 在我的印象里“帆姐”永远焦头烂额地忙于各种学生工作. 这四年来您辛苦了! 您永远是我们的“帆姐”! 因为岁数差得不大, 我总是称呼彭婧老师为“婧哥哥”. 你是我在“数学外卖”中的引路人, 不论是组员时还是组长时. 也多亏了你, 我总是敢于挑战一些自己没有做过的事. 从你的身上, 我学到了对待他人要打心底里热心, 要善于伸出援助之手. 你说我们这些人毕业了以后你会哭, 其实我们也不例外. 这样的情感是世间少有的.

我要感谢曾经帮助过我的王锦东、姚天洋、郝亦雯、吴悦同、王铭恺等同学, 我的室友、学弟学妹们. 你们为我原本枯燥的大学生活增添了色彩. 我要特别感谢我的两位好友——刘炳言和宋增春. 以前我们三人常年“混迹”于各种讨论班, 课上课下讨论问题, 分享自己的生活和爱好. 尽管今朝各奔东西, 但今后我们也是永远的好朋友.

我要感谢曾经和我一同参加比赛的张越、周柳曼、汤晨宇、张远航、王子鑫等同学. 感谢我们一起奋斗过的时光. 其中尤其要感谢我的死党张越. 尽管你是我的高中(不同班的)同学, 但我大学才认识你. 相见恨晚, 我们有许多共同的爱好, 曾经互相倾诉过自己的不快. 你拥有积极向上的生活态度和一颗温暖待人的心. 期待你在未来闯出自己的天地.

我要感谢在我访问中科院计算数学所时帮助过我的各位老师、师兄师姐和同学们. 感谢你们带我融入新的环境, 让我得以快速地进入状态. 其中尤其要感谢刘颖老师、刘为师兄、陈雅丹师姐、肖纳川师兄、高斌师兄、谢鹏程、裴骞、郭仲琨.

最后, 我要感谢我的家人和女朋友. 是你们无微不至的关怀和无私的爱给予了继续向前的勇气, 使我在自己选择的道路上不畏挫折, 坚定地走下去. 我会满怀热情地面对今后的学习、工作以回报他们的殷切希望.

装 订 线