大数据分析中的算法 习题解答与实验报告

原生生物

* 对应文再文老师《大数据分析中的算法》课程，教材 1 指刘浩洋、户将、李勇锋、文再文《最优化：建模、算法与理论》，教材 2 指文再文课题组《大数据分析中的算法》。

目录

1 优化引论作业

1. 教材 1 习题 4.2

记 δ_a^b 当 $a=b$ 时为 1，否则为 0。

(a) 设 $A=\{i\mid c_i>0\}$、$B=\{i\mid c_i<0\}$、$Z=\{i\mid c_i=0\}$，则考虑每项知

$$\sum_{i=1}^n c_ix_i=\sum_{i\in A}c_ix_i+\sum_{i\in B}c_ix_i\geq\sum_{i\in B}c_i$$

等号成立当且仅当 $i\in A$ 时 $x_i=0$、$i\in B$ 时 $x_i=1$。

(b) 设 $A=\{i\mid c_i>0\}$、$B=\{i\mid c_i<0\}$、$Z=\{i\mid c_i=0\}$，则考虑每项知

$$\sum_{i=1}^n c_ix_i=\sum_{i\in A}c_ix_i+\sum_{i\in B}c_ix_i\geq\sum_{i\in B}c_i-\sum_{i\in A}c_i=-\|c\|_1$$

等号成立当且仅当 $i\in A$ 时 $x_i=-1$、$i\in B$ 时 $x_i=1$。

(c) 若 c 的两分量 $c_a\neq c_b$，取 $x_i=t(\delta_i^b-\delta_i^a)$，则 $c^Tx=t(c_b-c_a)$，可趋于负无穷，无最小值。

若 c 所有分量均相同，设为 t，则

$$\sum_{i=1}^n c_ix_i=t\sum_{i=1}^n x_i\geq-|t|$$

t 非零时当且仅当 x_i 和为 $-\mathrm{sgn}(t)$ 时取到最小值，$t=0$ 时则恒零。

(d) 设 c 最小的分量为 c_m，则由 $x\geq0$

$$\sum_{i=1}^n c_ix_i\geq c_m\sum_{i=1^n}x_i=c_m$$

等号成立当且仅当 $c_i=c_m$ 的所有 x_i 和为 1。

2. 教材 1 习题 4.4

也即对给定的 a_i 与 b_i 求 (变元为 $X\in\mathcal{S},y\in\mathbb{R}^n,z\in\mathbb{R}$)

$$\min\sum_{i=1}^m(a_i^TXa_i+y^Ta_i+z-b_i)^2$$

使得 X 半负定 (直接考虑二阶导可知其等价于 f 凹) 且

$$\forall l\leq a\leq u,\quad a^TXa+y^Ta+z\geq0$$

$$\forall l\leq a\leq\hat{a}\leq u,\quad a^TXa+y^Ta\leq\hat{a}^TX\hat{a}+y^T\hat{a}$$

下面证明其为凸优化问题。由于目标函数为 X、y、z 的二次函数，且为平方和，根据线性代数知识可知其为正定二次型，于是海瑟矩阵正定，其为凸函数。

由于凸集的交仍为凸集，只需证明三个不等式约束的成立集合分别为凸集即可：

- 若 X_1、X_2 半负定，有

$$\forall s\in\mathbb{R}^n,\quad s^TX_1s\leq0,s^TX_2s\leq0$$

于是

$$\forall s\in\mathbb{R}^n,\forall\lambda\in[0,1],\quad s^T(\lambda X_1+(1-\lambda)X_2)s=\lambda s^TX_1s+(1-\lambda)s^TX_2s\leq0$$

从而第一个不等式约束成立集合为凸集。

- 若 X_1, y_1, z_1、X_2, y_2, z_2 均满足第二个条件，有对任意 $l \le a \le u, \lambda \in [0,1]$

$$\begin{aligned}&a^T(\lambda X_1 + (1-\lambda)X_2)a + (\lambda y_1 + (1-\lambda)y_2)^T a + \lambda z_1 + (1-\lambda)z_2\\ =&\lambda(a^T X_1 a + y_1^T a + z_1) + (1-\lambda)(a^T X_2 a + y_2^T a + z_2)\\ \ge&0\end{aligned}$$

  从而第二个不等式约束成立集合为凸集。

- 若 X_1, y_1, z_1、X_2, y_2, z_2 均满足第三个条件，有对任意 $l \le a \le \hat{a} \le u, \lambda \in [0,1]$

$$\begin{aligned}&a^T(\lambda X_1 + (1-\lambda)X_2)a + (\lambda y_1 + (1-\lambda)y_2)^T a\\ =&\lambda(a^T X_1 a + y_1^T a) + (1-\lambda)(a^T X_2 a + y_2^T a)\\ \le&\lambda(\hat{a}^T X_1 \hat{a} + y_1^T \hat{a}) + (1-\lambda)(\hat{a}^T X_2 \hat{a} + y_2^T \hat{a})\\ =&\hat{a}^T(\lambda X_1 + (1-\lambda)X_2)\hat{a} + (\lambda y_1 + (1-\lambda)y_2)^T \hat{a}\end{aligned}$$

  从而第三个不等式约束成立集合为凸集。

第三个不等式约束可以进行化简：由于此单调性等价于在范围内对每个分量单调 (考虑 $\hat{a}$ 与 a 只有一个分量不同)，而求导可得

$$\nabla_a(a^T X a + y^T a) = 2Xa + y$$

于是由连续性可化为

$$\forall l \le a \le u, \quad 2Xa + y \ge 0$$

再由线性函数最值一定在极点取到可化为

$$\forall a \in \prod_{i=1}^{n}\{l_i, u_i\}, \quad 2Xa + y \ge 0$$

利用第三个不等式约束可进一步化简第二个不等式约束为

$$l^T X l + y^T l + z \ge 0$$

3. 教材 1 习题 4.6

记 δ_a^b 当 $a = b$ 时为 1，否则为 0，直接计算可知

$$\|x\|_0 + \|Dx - a\|_2^2 = \sum_{i=1}^{n}\left((d_i x_i - a_i)^2 + 1 - \delta_{x_i}^0\right)$$

这将原问题拆分为了 n 个独立的优化问题。设 $f_i(x_i) = (d_i x_i - a_i)^2 + 1 - \delta_{x_i}^0$，分类讨论：

- 若 $d_i = 0$，$f_i(x_i) = a_i^2 + 1 - \delta_{x_i}^0 \ge a_i^2$，等号成立当且仅当 $x_i = 0$；
- 若 $d_i \ne 0$、$a_i = 0$，当且仅当 $x_i = 0$ 时 $f_i(x_i)$ 取到最小值 0。
- 若 $d_i \ne 0$、$a_i \ne 0$，$x_i = 0$ 时 $f_i(x_i) = a_i^2$，否则

$$f_i(x_i) = (d_i x_i - a_i)^2 + 1 \ge 1$$

  等号成立当且仅当 $x_i = a_i/d_i$。由此比较两者可得最小值。

综合以上可得最优解为

$$x_i^* = \begin{cases}0 & d_i = 0 \text{ or } a_i^2 < 1\\ a_i/d_i & d_i \ne 0 \text{ and } a_i^2 > 1\\ 0 \text{ or } a_i/d_i & d_i \ne 0 \text{ and } a_i^2 = 1\end{cases}$$

2 对偶理论作业

1. 教材 1 习题 5.1

   设 A 正交相似对角化为 P^TDP，D 的第 i 个对角元为 λ_i，利用乘可逆阵对应可逆变换，令 $y=Px$ 可知原问题等价为

   $$\min_{y\in\mathbb{R}^n} y^TDy+2b^TP^{-1}y$$

   记 $c=P^{-T}b$，进一步写为

   $$\min_{y\in\mathbb{R}^n} y^TDy+2c^Ty=\sum_{i=1}^{n}(\lambda_iy_i^2+c_iy_i)$$

   由此，只要有某个 $\lambda_i<0$，即可改变 y_i 使得值趋于负无穷，最小值不可能存在，于是至少需要 A 半正定。另一方面，A 半正定时根据凸函数二阶定义可知目标函数凸，其稳定点、局部极小点、全局极小点相互等价，因此最小值存在当且仅当驻点存在，求导也即 $Ax+b=0$ 存在解。

   根据线性方程组知识，最终得到原问题最优解存在当且仅当 A 半正定且 $\mathrm{rank}A=\mathrm{rank}(A,b)$。

2. 教材 1 习题 5.7

   (a) Lagrange 对偶函数为

   $$g(\nu)=\inf_x\left(\|x\|_1+\nu^T(Ax-b)\right)$$

   将右侧写为 $\|x\|_1+\nu^TAx-\nu^Tb$，由于对实数 t 的优化问题 $|t|+at$ 当且仅当 $|a|\le 1$ 时最小值存在为 0，考虑 x 每个分量前的系数可发现当且仅当 $\|A^T\nu\|_\infty\le 1$ 时最小值存在，为 $-\nu^Tb$，于是对偶问题为

   $$\max_\nu -\nu^Tb\qquad \text{s.t.}\quad \|A^T\nu\|_\infty\le 1$$

   代换 $y=-\nu$ 最终写为

   $$\max_y y^Tb\qquad \text{s.t.}\quad \|A^Ty\|_\infty\le 1$$

   (b) 记 $r=Ax-b$，等式约束即 $Ax-b-r=0$，Lagrange 对偶函数为

   $$g(\nu)=\inf_{x,r}\left(\|r\|_1+\nu^T(Ax-b-r)\right)$$

   与之前类似，r 的部分最小值存在当且仅当 $\|\nu\|_\infty\le 1$，而 x 的部分为线性函数，最小值存在当且仅当 $\nu^TA=0$，由此最终得到对偶问题为

   $$\max_\nu -\nu^Tb\qquad \text{s.t.}\quad \|\nu\|_\infty\le 1,\nu^TA=0$$

   代换 $y=-\nu$ 最终写为

   $$\max_y y^Tb\qquad \text{s.t.}\quad \|y\|_\infty\le 1,A^Ty=0$$

   (c) 记 $r=Ax-b$，等式约束即 $Ax-b-r=0$，Lagrange 对偶函数为

   $$g(\nu)=\inf_{x,r}\left(\|r\|_\infty+\nu^T(Ax-b-r)\right)$$

   这时 x 部分最小值存在仍然当且仅当 $\nu^TA=0$，而 r 的部分为 $\|r\|_\infty-\nu^Tr$，考虑 r 所有分量绝对值相同的情况，可发现最小值存在当且仅当 $\|\nu\|_1\le 1$，此时这部分最小值为 0，由此最终得到对偶问题为

   $$\max_\nu -\nu^Tb\qquad \text{s.t.}\quad \|\nu\|_1\le 1,\nu^TA=0$$

   代换 $y=-\nu$ 最终写为

   $$\max_y y^Tb\qquad \text{s.t.}\quad \|y\|_1\le 1,A^Ty=0$$

(d) Lagrange 对偶函数为

$$g(\lambda)=\inf_x\big(x^TAx+2b^Tx+\lambda(x^Tx-1)\big),\quad \lambda\geq 0$$

由于此也即

$$g(\lambda)=\inf_x(x^T(A+\lambda I)x+2b^Tx-\lambda),\lambda\geq 0$$

根据条件 $A+\lambda I$ 正定，上述问题对任何 λ 最小值存在唯一，代入稳定点表达式

$$x=-(A+\lambda I)^{-1}b$$

可知最小值为

$$g(\lambda)=-b^T(A+\lambda I)^{-1}b-\lambda$$

由此最终得到对偶问题

$$\max_\lambda\big(-b^T(A+\lambda I)^{-1}b-\lambda\big)\qquad \text{s.t.}\quad \lambda\geq 0$$

3. 教材 1 习题 5.11

为了能应用无约束最优化问题进行分析，考虑等价的最优化问题

$$\min_x\frac{x^TAx}{x^Tx}\qquad \text{s.t.}\quad x\neq 0$$

由于去掉 0 后的空间为开集，其任何一点的一阶最优性条件即可通过无约束优化的情况判别。直接对原函数求梯度得到

$$\nabla\frac{x^TAx}{x^Tx}=\frac{2(x^Tx)Ax-2(x^TAx)x}{(x^Tx)^2}$$

由此其一阶最优性条件为 (考虑到 $x\neq 0$)

$$Ax=\frac{x^TAx}{x^Tx}x$$

由此可看出 x 为 A 的一个特征向量，而计算验证可发现 $Ax=\lambda x$ 时 λ 一定为 $\frac{x^TAx}{x^Tx}$，于是 x 为稳定点当且仅当其为特征向量——对原问题来说，这意味着 x 为 A 的某个模长为 1 的特征向量，此时的目标函数值即为 x 对应的特征值。

设 A 的不同特征值从小到大排列为 $\lambda_1,\ldots,\lambda_k$。下面证明，$x$ 为局部极小点、全局极小点均当且仅当其为 λ_1 的模长 1 的特征向量；x 为局部极大点、全局极大点当且仅当其为 λ_k 的模长 1 的特征向量。

由于全局极小点一定为稳定点，全局极小值一定为某个稳定点的目标函数值，从而必然为 A 特征值。由此，λ_1 对应的特征向量一定能使目标函数取到全局极小值，为全局极小点，也自然为局部极小点。

对某个特征值 $\lambda_i>\lambda_1$ 的模长 1 的特征向量 α，取 λ_1 的模长 1 的特征向量 β，则取某 $\mu>0$，利用实对称矩阵不同特征值的特征向量相互垂直直接计算有

$$\frac{(\alpha+\mu\beta)^TA(\alpha+\mu\beta)}{(\alpha+\mu\beta)^T(\alpha+\mu\beta)}=\frac{\lambda_i^2+\mu^2\lambda_1^2}{1+\mu^2}>\lambda_1$$

由于 μ 可以任意小，此时 $\frac{\alpha+\mu\beta}{\|\alpha+\mu\beta\|}$ 可以与 α 任意接近，这就说明了 α 不是原问题的局部极小点。

同理，x 为局部极大点、全局极大点当且仅当其为 λ_k 的特征向量，因此 x 为鞍点当且仅当其为 A 的 λ_1,λ_k 以外的特征值的模长 1 的特征向量。

3 线性规划作业

1. (a) 直接展开

$$(C+uv^T)\left(C^{-1}-\frac{C^{-1}uv^TC^{-1}}{1+v^TC^{-1}u}\right)=I-\frac{uv^TC^{-1}}{1+v^TC^{-1}u}+uv^TC^{-1}-\frac{uv^TC^{-1}uv^TC^{-1}}{1+v^TC^{-1}u}$$

二三两项都是向量 uv^TC^{-1} 的倍数，可合并为

$$I+\frac{(1+v^TC^{-1}u)-1}{1+v^TC^{-1}u}uv^TC^{-1}-\frac{uv^TC^{-1}uv^TC^{-1}}{1+v^TC^{-1}u}=I+\frac{(v^TC^{-1}u)uv^TC^{-1}}{1+v^TC^{-1}u}-\frac{uv^TC^{-1}uv^TC^{-1}}{1+v^TC^{-1}u}$$

而最后一项项分母可将中间的 $v^TC^{-1}u$ 作为数乘提出，从而也剩下

$$u(v^TC^{-1}u)v^TC^{-1}=(v^TC^{-1}u)uv^TC^{-1}$$

由此互相抵消可知结果为 I，从而互逆。

(b) 由于 m 阶方阵与 m 维向量的乘法是 $O(m^2)$ 的，执行如下流程即可：

- 计算行向量 $\alpha=v^TC^{-1}$；
- 计算 $a=\alpha u+1$；
- 计算列向量 $\beta=C^{-1}u$；
- 计算矩阵 $T=\frac{1}{a}\beta\alpha$；
- 计算 $C^{-1}-T$，即为结果。

(c) 由单纯形法定义可知

$$B=(A_1,\ldots,A_{l-1},A_{B(l)},A_{l+1},\ldots,A_m)$$

$$\bar{B}=(A_1,\ldots,A_{l-1},A_{\bar{B}(l)},A_{l+1},\ldots,A_m)$$

由此可知

$$B-\bar{B}=(0,\ldots,0,A_{B(l)}-A_{\bar{B}(l)},0,\ldots,0)$$

直接计算可知右侧即为 $(A_{B(l)}-A_{\bar{B}(l)})e_l^T$。

(d) 由于 B 可逆，两侧同右乘 B 不会影响等式成立性，也即只需证明存在 g_i 使得

$$e_i^T\bar{B}^{-1}B=e_i^T-g_ie_l^T$$

利用上一问直接计算左侧

$$e_i^T\bar{B}^{-1}B=e_i^T\bar{B}^{-1}(\bar{B}+B-\bar{B})=e_i^T+e_i^T\bar{B}^{-1}(A_{B(l)}-A_{\bar{B}(l)})e_l^T$$

由此即得

$$g_i=-e_i^T\bar{B}^{-1}(A_{B(l)}-A_{\bar{B}(l)})=e_i^T\bar{B}^{-1}(A_{\bar{B}(l)}-A_{B(l)})$$

下面给出具体的更新方案：由于

$$\bar{B}=B+(A_{B(l)}-A_{\bar{B}(l)})e_l^T$$

利用 (b)，若 B^{-1} 已知，得到 $\bar{B}^{-1}$ 只需要 $O(m^2)$ 复杂度。由此，在计算过初始的 B 之后，每次迭代用 $O(m^2)$ 复杂度可得到 $\bar{B}^{-1}$，于是 $O(m^2)$ 复杂度也可得到 $\bar{B}^{-1}(A_{\bar{B}(l)}-A_{B(l)})$ (即 g 的每个分量)，这样迭代过程中新的 $c_{\bar{B}}^T\bar{B}^{-1}$ 可以由这些 $e_i^T\bar{B}^{-1}$ 的线性组合快速计算。

2. 由定义，若 d 为可行方向，存在一列 $z_k \in P$ 与实数 $t_k > 0$ 使得 $z_k \to x$ 且

$$d = \lim_{k\to\infty} \frac{z_k - x}{t_k}$$

左乘 A 可得

$$Ad = \lim_{k\to\infty} \frac{Az_k - Ax}{t_k} = \lim_{k\to\infty} 0 = 0$$

此外对比第 i 个分量可知

$$d_i = \lim_{k\to\infty} \frac{(z_k)_i - x_i}{t_k}$$

当 $x_i = 0$ 时，由 $(z_k)_i \geq 0$ 可知 $\frac{(z_k)_i}{t_k} \geq 0$，从而根据极限保序性 $d_i \geq 0$。

另一方面，若 d 满足 $Ad = 0$ 且 x 为 0 的分量 $d_i \geq 0$，记 $x(\lambda) = x + \lambda d$，则由线性性 $Ax(\lambda) = 0$。对所有 $x_i > 0$ 且 $d_i < 0$ 的分量，考虑 $\frac{|x_i|}{|d_i|}$ 的最小值，记作 λ_0，对比每个分量可发现当 $\lambda \in [0, \lambda_0]$ 时 $x(\lambda) \geq 0$，从而 $x(\lambda) \in P$，从而取 $z_k = x(\lambda_0/k)$、$t_k = \lambda_0/k$ 即可得到 d 为可行方向。

4 稀疏优化作业

1. (a) 在对偶理论作业中已经得到结果为

$$\max_y y^T b \qquad \text{s.t.} \quad \|A^T y\|_\infty \le 1$$

(b) 由于只存在等式约束，对应的 KKT 条件为 (仍记 $y = -\nu$，直接计算一范数对每个分量的次梯度)

$$Ax = b$$
$$\begin{cases} (A^T y)_i = 1 & x_i > 0 \\ (A^T y)_i \in [-1, 1] & x_i = 0 \\ (A^T y)_i = -1 & x_i < 0 \end{cases}$$

(c) 验证可知此问题是满足 Slater 约束品性的凸优化问题，因此 x 为最优解当且仅当上述的 y 存在。沿用 (d) 中的记号 $T = \{i \mid z_i \neq 0\}$，并记 A_T 为下标在 T 中的列构成的子矩阵，我们证明，若 A_T 列满秩，且 $i \in T^c$ 时 $(Ay)_i \in (-1, 1)$，则 z 是唯一最优解。

对任何满足 $Ah = 0$ 的非零 h，我们将方程重新写作 $A_T h_T + A_{T^c} h_{T^c} = 0$。由于 A_T 列满秩，存在矩阵 B 使得 $BA_T = I$，从而

$$h_T = -BA_{T^c} h_{T^c}$$

于是若 $h_{T^c} = 0$ 可知 $h = 0$，矛盾，从而 h_{T^c} 不为 0。

记 u 为 z 对应的 $A^T y$，由于所有 $A^T y$ 构成的空间为 A 的行向量生成的线性空间，由行空间解空间正交性可知 $u^T h = 0$，于是分解分量得到

$$\sum_{i \in T} u_i h_i = -\sum_{i \in T^c} u_i h_i$$

由 $h_{T^c} \neq 0$ 进一步得到

$$-\sum_{i \in T} \operatorname{sign}(z_i) h_i = \sum_{i \in T^c} u_i h_i < \sum_{i \in T^c} |h_i|$$

由此，分类讨论可得 $|a + b| - |a| \ge \operatorname{sign}(a) b$，从而

$$\|z + h\|_1 - \|z\|_1 = \sum_{i \in T} (|z_i + h_i| - |z_i|) + \sum_{i \in T^c} |h_i| \ge \sum_{i \in T} \operatorname{sign}(z_i) h_i + \sum_{i \in T^c} |h_i| > 0$$

这就得到了证明。

(d) 由于 $-h$ 也是 $Ah = 0$ 的解，我们将条件等价写为

$$\sum_{i \in T} \operatorname{sign}(z_i) h_i + \sum_{i \in T^c} |h_i| \ge 0$$

若此条件满足，与 (c) 完全相同可得对任何满足 $Ah = 0$ 的 h 有

$$\|z + h\|_1 - \|z\|_1 = \sum_{i \in T} (|z_i + h_i| - |z_i|) + \sum_{i \in T^c} |h_i| \ge \sum_{i \in T} \operatorname{sign}(z_i) h_i + \sum_{i \in T^c} |h_i| \ge 0$$

从而 z 一定为一个最优解。

反之，若此条件不满足，我们说明存在 h 使得 $\|z + h\|_1 < \|z\|_1$ 即可。取出满足

$$Ah = 0, \quad \sum_{i \in T} \operatorname{sign}(z_i) h_i + \sum_{i \in T^c} |h_i| < 0$$

的 h，可发现，若 $a \neq 0$，对任何 b，只要 t 满足 $a+tb$ 与 a 同号，即能使得 $|a+tb|-|a| = \operatorname{sign}(a)tb$，而这只要 $t > 0$ 且足够小即成立，从而考虑每个分量并取公共的最小值即得存在 $t > 0$ 使得

$$\sum_{i\in T}(|z_i + th_i| - |z_i|) = \sum_{i\in T} t\operatorname{sign}(z_i)h_i$$

由此

$$\|z + th\|_1 - \|z\|_1 = \sum_{i\in T}(|z_i + th_i| - |z_i|) + \sum_{i\in T^c} |th_i| = \sum_{i\in T} t\operatorname{sign}(z_i)h_i + \sum_{i\in T^c} t|h_i| < 0$$

而 $A(th) = 0$ 且 t 非零，从而 $z + th$ 为可行解，即得 z 并非最优。

5 核范数作业

1. • 记号严谨化

设 $X \in \mathbb{R}^{m\times n}$ 的奇异值分解为

$$X = \hat{U}\begin{pmatrix}\Sigma & O\\ O & O\end{pmatrix}\hat{V}^T$$

其中 $\hat{U}$ 为 m 阶正交阵，$\hat{V}$ 为 n 阶正交阵，Σ 为对角元均正且逐渐减小的可逆对角阵，其阶数 $r = \operatorname{rank} X$，对角元从大到小 $\sigma_1, \dots, \sigma_r$。

计算可发现，设 U 为 $\hat{U}$ 的前 r 列 (由正交阵定义它们标准正交) 构成的 $m \times r$ 阶矩阵，V 为 $\hat{V}$ 的前 r 列 (同样标准正交) 构成的 $n \times r$ 阶矩阵，则奇异值分解可以写为

$$X = U\Sigma V^T$$

题目中的 U, V 即为上述的 U, V，并记

$$\hat{\Sigma} = \begin{pmatrix}\Sigma & O\\ O & O\end{pmatrix}$$

• 问题转化

根据定义，$m \times n$ 实矩阵 X 的次梯度也即矩阵 $A \in \mathbb{R}^{m\times n}$ 满足对任意 $Y \in \mathbb{R}^{m\times n}$ 有

$$\|Y\|_* \geq \|X\|_* + \operatorname{tr}(A^T(Y - X))$$

由上述记号，因为正交相抵不改变核范数，且正交阵可逆，上式等价于对任意 $Y \in \mathbb{R}^{m\times n}$ 有 (注意 $\|\hat{\Sigma}\|_* = \|\Sigma\|_* = \sigma_1 + \cdots + \sigma_r$)

$$\|\hat{U}^T Y \hat{V}\|_* \geq \|\Sigma\|_* + \operatorname{tr}(A^T\hat{U}(\hat{U}^T Y\hat{V} - \hat{\Sigma})\hat{V}^T)$$

由于左乘 $\hat{U}^T$、右乘 $\hat{V}$ 是 $\mathbb{R}^{m\times n}$ 上可逆线性变换，再利用 tr 的可交换性，上式等价于对任意 $Y \in \mathbb{R}^{m\times n}$ 有

$$\|Y\|_* \geq \|\Sigma\|_* + \operatorname{tr}((\hat{U}^T A\hat{V})^T(Y - \hat{\Sigma}))$$

记 $B = \hat{U}^T A\hat{V}$，我们证明

$$A = UV^T + W, \quad U^T W = O, WV = O, \|W\|_2 \leq 1$$

当且仅当

$$B = \begin{pmatrix}I_r & O\\ O & W_0\end{pmatrix}, \quad \|W_0\|_2 \leq 1$$

– 上推下

若上方条件满足，利用正交性直接计算可发现

$$\hat{U}^T U V^T \hat{V} = \begin{pmatrix}I_r\\ O\end{pmatrix}\begin{pmatrix}I_r & O\end{pmatrix} = \begin{pmatrix}I_r & O\\ O & O\end{pmatrix}$$

而由 $U^T W = O$、$WV = O$ 可知 (这里 $U_\perp$、$V_\perp$ 为 $\hat{U}$、$\hat{V}$ 去掉 U、V 后剩下的列)

$$\hat{U}^T W\hat{V} = \begin{pmatrix}U^T\\ U_\perp^T\end{pmatrix} W \begin{pmatrix}V & V_\perp\end{pmatrix} = \begin{pmatrix}O & O\\ O & U_\perp^T W V_\perp\end{pmatrix}$$

记 $W_0 = U_\perp^T W V_\perp$，可发现已经成为 B 的形式，且利用二范数定义与乘正交阵不改变二范数可发现

$$\|W\|_2 = \min_{x\neq 0}\frac{\|Wx\|}{\|x\|} = \min_{x\neq 0}\frac{\|\hat{U}^T Wx\|}{\|x\|} = \min_{x\neq 0}\frac{\|\hat{U}^T W\hat{V}x\|}{\|\hat{V}x\|} = \min_{x\neq 0}\frac{\|\hat{U}^T W\hat{V}x\|}{\|x\|}$$

从而 $\|W\|_2 = \|\hat{U}^T W\hat{V}\|_2$，进一步由定义可发现其与 $\|W_0\|_2$ 相等，从而得证 $\|W_0\|_2 \leq 1$。

– 下推上

若 B 符合要求，设 $A=\hat{U}B\hat{V}^T$，则

$$A=\begin{pmatrix}U & U_\perp\end{pmatrix}\begin{pmatrix}I_r & O\\ O & W_0\end{pmatrix}\begin{pmatrix}V^T\\ V_\perp^T\end{pmatrix}=UV^T+U_\perp W_0V_\perp^T$$

记 $W=U_\perp W_0V_\perp^T$，计算验证与上方类似，又由

$$W=\hat{U}\begin{pmatrix}O & O\\ O & W_0\end{pmatrix}\hat{V}^T$$

可与上方类似验得到 $\|W\|_2=\|W_0\|_2\le 1$，从而得证。

由此，问题最终化为，证明 $B\in\mathbb{R}^{m\times n}$ 满足对任何 $Y\in\mathbb{R}^{m\times n}$ 有

$$\|Y\|_*\ge\|\Sigma\|_*+\operatorname{tr}(B^T(Y-\hat{\Sigma}))$$

当且仅当 (将 W_0 重新记为 W，diag 表示排列为对角阵)

$$B=\operatorname{diag}(I_r,W),\quad \|W\|_2\le 1$$

- 最终证明

利用教材引理 2 与下一道习题可知 B 为次梯度当且仅当

$$\operatorname{tr}(B^T\hat{\Sigma})=\sum_{i=1}^r\sigma_i,\quad \|B\|_2\le 1$$

也即要证上述方程解为

$$B=\operatorname{diag}(I_r,W),\quad \|W\|_2\le 1$$

– 先说明符合上述形式的为解。由于 $\operatorname{tr}(B^T\hat{\Sigma})$ 为对应位置元素乘积求和，直接计算可知其为 $\sum_{i=1}^r\sigma_i$，而将 x 分块为前 r 个分量 y、后 $n-r$ 个分量 z 可发现

$$\|B\|_2=\max_{x\ne 0}\frac{\sqrt{\|y\|^2+\|Wz\|^2}}{\sqrt{\|y\|^2+\|z\|^2}}$$

由于 $\|Wz\|^2\le\|z\|^2$，z 固定时 $\|y\|$ 越大会使值越大，因此最小在 $\|y\|\to\infty$ 时取到，即得 $r>0$ 时 $\|B\|_2=1$，而 $r=0$ 时 $\|B\|_2=\|W\|_2\le 1$，成立。

– 再说明所有解都能写为这样的形式。由 $\hat{\Sigma}$ 的形式可知第一个条件等价于

$$\sum_{i=1}^r b_{ii}\sigma_i=\sum_{i=1}^r\sigma_i$$

若某个 $b_{ii}>1$，考虑 $x=e_i$ 可发现 $\|Bx\|\ge|b_{ii}|>\|x\|$，与第二个条件矛盾，因此每个 $b_{ii}\le 1$，再由上式可知只能取等，从而得到第一个条件化为 $b_{11}=\cdots=b_{rr}=1$。

若前 r 列中还有元素 $b_{ij}\ne 0, i\ne j$，考虑 $x=e_j$ 可发现 $\|Bx\|\ge\sqrt{1+b_{ij}^2}>1=\|x\|$，与第二个条件矛盾；利用之前已证乘正交阵不改变二范数，设 B 奇异值向量为 σ，可知 $\|B\|_2=\|\operatorname{diag}(\sigma)\|_2=\|B^T\|_2$，从而前 r 行中有非零元素可类似得到矛盾。由此将 B 写为分块矩阵

$$B=\begin{pmatrix}I_r & O\\ O & W\end{pmatrix}$$

将 x 分块为前 r 个分量 y、后 $n-r$ 个分量 z 可发现

$$\|Bx\|^2=\|y\|^2+\|Wz\|^2,\quad \|x\|^2=\|y\|^2+\|z\|^2$$

若 $\|W\|_2>1$，取 $y=0$，z 使得 $\|Wz\|>\|z\|$ 即矛盾，从而 $\|W\|_2\le 1$，这就得到了证明。

2. 也即要证明

$$\max_{\|Z\|_2\le 1} \operatorname{tr}(Z^T X) = \|X\|_*$$

在上题中我们已经证明了左/右乘正交阵不改变谱范数与二范数，由此设 X 的奇异值分解为

$$X = U \operatorname{diag}(\Sigma, O) V^T$$

其中 U, V 为正交阵，$\Sigma = \operatorname{diag}(\sigma_1, \dots, \sigma_r)$，可得

$$\max_{\|Z\|_2\le 1} \operatorname{tr}(Z^T X) = \max_{\|Z\|_2\le 1} \operatorname{tr}(V^T Z^T U \operatorname{diag}(\Sigma, O)) = \max_{\|U^T Z V\|_2\le 1} \operatorname{tr}((U^T Z V)^T \operatorname{diag}(\Sigma, O))$$

最终化为

$$\max_{\|Z\|_2\le 1} \operatorname{tr}(Z^T \operatorname{diag}(\Sigma, O)) = \max_{\|Z\|_2\le 1} \sum_{i=1}^{r} z_{ii}\sigma_i$$

若某个 $z_{ii} > 1$，取 $x = e_i$ 即得到 $\|Zx\| \ge |z_{ii}| > \|x\|$，从而所有 z_{ii} 至多为 1，另一方面，取 $Z = \operatorname{diag}(I_r, O)$ 可直接计算得 $\|Z\|_2 = 1$，综合即得

$$\max_{\|Z\|_2\le 1} \sum_{i=1}^{r} z_{ii}\sigma_i = \sum_{i=1}^{r} \sigma_i = \|X\|_*$$

从而得证。

6 最优传输实验报告

6.1 算法实现

6.1.1 问题与算法

我们考虑的问题为最优传输问题

$$\min_{\pi\in\mathbb{R}^{m\times n}} \sum_{i=1}^{m}\sum_{j=1}^{n}\pi_{ij}c_{ij} \qquad \text{s.t.} \quad \forall i, \sum_{j=1}^{n}\pi_{ij}=\alpha_i, \quad \forall j, \sum_{i=1}^{m}\pi_{ij}=\beta_j, \quad \forall i,j,\pi_{ij}\geq 0$$

这里所有 α_i 与所有 β_j 均总和为 1，且每项都非负。

1. Gurobi 求解与最优性检验

   经过了艰难的发邮件配环境以后，就可以使用 Gurobi 求解器了。由于要求直接调用 Gurobi，采用的是方法

   ```
   result = gurobi(model, params)
   ```

   根据 Gurobi 的要求，这里的 model 需要设置为一个线性规划问题。由此需要将读取的 source、dest 与 C 重新写成线性规划问题的标准形式 $\min_{Ax=b,x\geq 0} c^Tx$。利用内置的 repelem 与 repmat 函数可生成 A 中为 1 分量的下标，而 b 由 source、dest 拼接，c 直接由 C 拉平得到 (参考代码中已给出 A、b、c 的构造)，也即

   ```
   model.A = sparse(i_pos, j_pos, 1, m + n, m * n);
   model.rhs = [source; dest];
   model.obj = reshape(C', [], 1);
   ```

   利用 params 参数可以指定方法，查阅手册可发现取 0 和 1 表示原问题与对偶问题的单纯形法，取 2 表示障碍函数内点法，根据需要，我们只需测试这三种情况。

   对于最优性检验，由于 Gurobi 返回的结果中包含每个位置的检验数，只需确认所有检验数都非负即可确认最优性。

2. Mosek 求解与最优性检验

   完成 Mosek 配置后，Mosek 是代替 MATLAB 自带的线性规划求解器 linprog 而使用的。因此，使用 Mosek 仍然需要构建上述的线性规划问题。

   Mosek 的参数设置中，若将 Simplex 设置成 primal 或 dual，使用的就是原始或对偶单纯形法，而根据输出可发现，设置为空则意味着使用原始-对偶内点法计算。我们同样只需要确认这三种方法。

   对于最优性检验，由于 Mosek 返回的结果中包含对应的对偶问题最优解 (乘子的相反数)，检验对偶间隙 c^Tx-b^Ty 即可确认最优性。

3. Sinkhorn 方法

   与之前对线性规划的精确求解不同，Sinkhorn 方法是一种仅针对最优传输使用的迭代近似算法。它考虑添加正则化的问题

   $$\min_{P\in U(a,b)} \big(\langle H,P\rangle - \varepsilon H(P)\big)$$

   理论来说，只要 ε 充分小，此问题解即可以逼近原问题解。

记 $K = \exp(-C/\varepsilon)$ (这里指数代表对每个分量作) 利用 KKT 条件可发现，此问题的最优解 P_ε 满足

$$P_\varepsilon = \mathrm{diag}(u) K \,\mathrm{diag}(v), \quad a = \mathrm{diag}(u) K v, \quad b = \mathrm{diag}(v) K^T u$$

由此可初始化

$$\hat{K} = \mathrm{diag}(a)^{-1} K$$

与 u 后 (为了避免出现奇异，初始化为全 1) 直接进行迭代

$$v = b./(K^T u), \quad u = 1./(\hat{K} v)$$

若 u 的更新较小则停机，最终合成 $P_\varepsilon = \mathrm{diag}(u) K \,\mathrm{diag}(v)$ 作为真实解的近似。

值得一提的是，检验可以发现，用较大 ε 的迭代结果 u 作为较小 ε 时 u 的初值并不能改善迭代次数：参考例子中直接全 1 初值进行 $\varepsilon = 0.1$ 的迭代次数为 18700，用 $\varepsilon = 1$ 的迭代结果作为初值迭代次数为 18653，无明显改善。因此，直接使用 Sinkhorn 方法只能孤立地取较小的 ε 作为估计，无法达成序贯最小化的效果。

由于此为迭代近似算法，我们将以之前单纯形法得到的最优值比较误差。

4. ADMM 方法

除此以外，我们还可以利用线性规划的对偶问题利用 ADMM 的思路实现算法。

将原问题的对偶写为

$$\max_{A^T y + s = c} \left(b^T y - I_{\geq 0}(s) \right)$$

这里示性函数 I 在条件成立时为 0，否则为正无穷。

此已经符合 ADMM 算法可以应用的形式，取定步长参数 t、l，可写出增广 Lagrange 函数

$$L_t(x, y, s) = -b^T y + I_{\geq 0}(s) + x^T (A^T y + s - c) + \frac{t}{2} \| A^T y + s - c \|_2^2$$

与对应的迭代公式

$$y^{k+1} = \arg\min_y L_t(y, x^k, s^k)$$

$$s^{k+1} = \arg\min_s L_t(y^{k+1}, x^k, s)$$

$$x^{k+1} = x^k + lt(A^T y^{k+1} + s^{k+1} - c)$$

由于此形式对 s、y 均为二次函数，假设 A 行满秩可得到显示迭代公式

$$y^{k+1} = -(AA^T)^{-1} \left(\frac{1}{t} (Ax^k - b) + A(s^k - c) \right)$$

$$s^{k+1} = \max \left(c - A^T y^{k+1} - \frac{1}{t} x^k, 0 \right)$$

代码实现时，可先行算出 $s - c$ 与 $A^T y$，避免重复计算。由于例子中遇到的 A 行数远少于列数，我们也将先算出 $(AA^T)^{-1}$ 以方便后续迭代。

不过，这里由于写出了 $(AA^T)^{-1}$，必须要求 A 行满秩，因此，我们需要去掉之前构造的 A、b 的最后一行，这是一个不独立的条件。其余条件对应的矩阵即符合满秩特性。

6.1.2 代码文件结构

`ADMM.m`	线性规划ADMM算法实现
`gen_data.m`	根据参考代码自行调整而成的生成source、dest与C的脚本
`sinkhorn.m`	Sinkhorn算法实现
`text_my_solver.m`	Sinkhorn与ADMM算法测试
`text_solver.m`	调用求解器求解测试

运行时应先运行生成数据的脚本，再执行两个测试脚本。

6.2 结果展示

首先，我自己取了更小规模的优化例子，也即两张随机选取的 16×16 灰度图片 (即文件夹中的 my_source.png 与 my_dest.png)，对应 source、dest 维数均为 256，参数个数 65536 对比各个算法的效果。

Gurobi 与 Mosek 求解器得到精确解的时间如下表，也附上了 Mosek 算法得到的对偶间隙 (Gurobi 由于直接验证检验数，无需考虑对偶间隙)：

表 1: Gurobi 与 Mosek 小规模例子求解时间表

算法	Gurobi		Mosek		
	时间 (s)	迭代次数	时间 (s)	迭代次数	对偶间隙
原始单纯形	0.16	8716	0.11	4113	1.45e-6
对偶单纯形	0.21	772	0.22	1116	6.66e-16
内点法	0.35	13	0.21	11	2.63e-11

可以发现，总体来说，单纯形方法的迭代次数远多于内点法，但虽然理论复杂度更高，其在充分优化后实际上对不太奇异的问题有很好的效果，让总时间效率更高。对偶问题的单纯形法比起原问题单纯形法降低了迭代次数 (这可能与变量个数大幅减少有关)，不过时间表现也未必更优。考虑对偶间隙可以发现，对偶单纯形由于变量更少，且迭代为精确迭代，误差累积相对更少。对更大规模的参考图片情况如下表：效果基本与之前的观察相同，不过在这样更大规模的情况下，内点法降低的迭代次数优势就更加显著了。

表 2: Gurobi 与 Mosek 更大规模例子求解时间表

算法	Gurobi		Mosek		
	时间 (s)	迭代次数	时间 (s)	迭代次数	对偶间隙
原始单纯形	3.13	17282	3.77	22999	1.03e-4
对偶单纯形	3.14	1655	12.51	6640	3.55e-15
内点法	2.65	13	5.9	12	1.72e-9

刚才两个例子中，我们利用精确算法得到的最优值分别为 0.5393647 与 5.257395，接下来将以它们作为标准衡量迭代算法的收敛性。

简单例子与复杂例子的 Sinkhorn 算法效果对比如下表，当 u 更新不超过 1e-6 时停机：

表 3: Sinkhorn 算法求解效果

ε	简单例子			复杂例子		
	时间 (s)	迭代次数	结果	时间 (s)	迭代次数	结果
1	0.17	1125	1.1885	1.17	1572	5.9277
0.1	2.94	18078	0.5398	13.5	18700	5.2577
0.02	38.6	281288	0.5394	-	-	NaN
0.01	-	-	NaN	-	-	NaN

为了加快迭代，复杂例子的迭代并未同时计算误差，而简单例子的迭代每十次计算一次误差 (从而会消耗额外时间)。从表中可以看到，Sinkhorn 算法在 ε 较小时确实表现出了收敛性，但由于存在 $K = \exp(-C/\varepsilon)$，当 ε 过小时将很快变得奇异，影响后续计算，且规模越大时奇异越容易发生。此外，它得迭代开销相对来说较大，每次需要计算稠密矩阵的乘法，对更大规模的情况时间消耗更加明显。

以 $\varepsilon = 0.02$ 时的简单例子为例，作出误差对数随迭代时间变化的图如图 1 。从图中可以看出，取 $\varepsilon = 0.02$ 已经足以让误差很小，最终稳定在，迭代效果已经较好。迭代过程在起初的不稳定后，很快进入

了相对稳定收敛的状态。

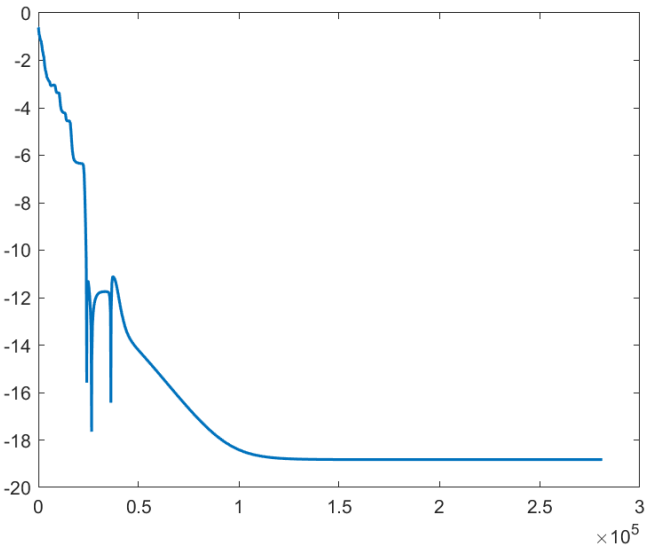

图 1: Sinkhorn 算法误差曲线

与之相对，此例子应用 ADMM 算法进行线性规划求解虽然能够收敛 (已用简单例子进行正确性验证)，但速度极慢，在简单情况，取定合适步长 $t = 0.1$、$l = 0.1$ 后，以对偶间隙 1e-6 作为停机标准，效果如图 2 。

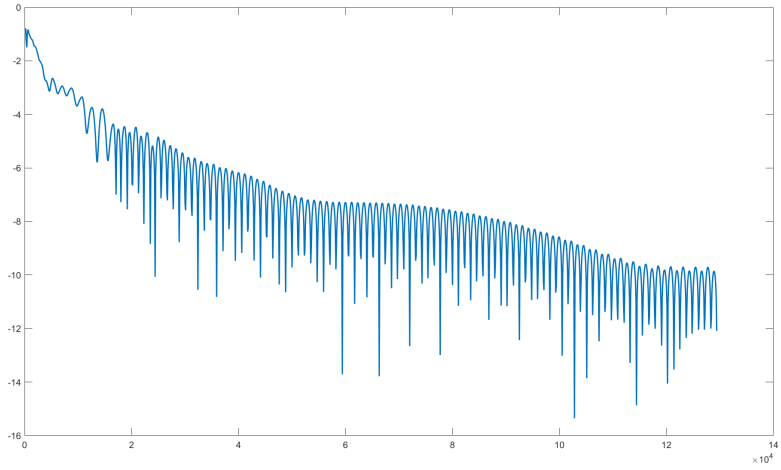

图 2: ADMM 算法误差曲线

即使选取其他参数也几乎无法消除迭代过程中的振荡，且时间达到了 282 秒。对复杂情况，无论如何调整参数都很难在可行时间内收敛。这说明，我们实现的 ADMM 算法并不适用这类变量极多的大规模情况求解。

7 期中实验报告

7.1 一范数最优化

本部分考虑的优化问题为

$$\min_x \mu\|x\|_1 + \frac{1}{2}\|Ax-b\|_2^2$$

并固定 A 为随机生成的 $m=512$ 行 $n=1024$ 列实矩阵，且存在稀疏向量 u 使得 $Au=b$，$\mu=0.01$。

* **实验问题 5 的解答在第二、三、四部分中分别叙述，实验问题 6 的解答在第二部分中叙述。**此外，为了结果稳定性，如无特殊说明，采用**固定种子**进行随机数生成，这样测试时所有算法对应的最优结果是相同的。

7.1.1 调包实现

算法实现

首先，我们需要用 Gurobi 与 Mosek 实现对应的优化。为了将其转化为能够求解的二次规划问题，我们将原问题化为如下的形式 (**1** 表示各分量全为 1 的向量)

$$\min_{z,x} \frac{1}{2}x^TA^TAx - b^TAx + \mu\mathbf{1}^Tz + \frac{1}{2}b^Tb \qquad \text{s.t.} \quad z \geq x, z \geq (-x)$$

由于每个 z_i 都大于等于 x_i 与 $-x_i$，且 μ 为正，它必然在取 $|x_i|$ 时最小，从而得到等价性。

由此，令 y 为 z、x 拼接成的列向量，其进一步写为

$$\min_y y^T\begin{pmatrix} O & O \\ O & \frac{1}{2}A^TA \end{pmatrix}y + y^T\begin{pmatrix} \mu\mathbf{1} \\ -A^Tb \end{pmatrix} + \frac{1}{2}b^Tb \qquad \text{s.t.} \quad \begin{pmatrix} I & I \\ I & -I \end{pmatrix}y \geq 0$$

当给定初值 x_0 时，迭代初值应为 (绝对值对每个分量取)

$$y_0 = \begin{pmatrix} |x_0| \\ x_0 \end{pmatrix}$$

结果展示

对于 Gurobi，设置 model 的 Q、obj、ojbcon 为上述二次函数的二次项、一次项、常数项，并设置 lb、sense、A 实现大于等于 0 的线性约束，设置 start 实现初始化，然后即可以采用默认算法进行求解。经过 13 次障碍函数法迭代后，输出的目标函数值约为 0.8530727，对应的 x 一范数约为 85.30566。可以发现，事实上 $\frac{1}{2}\|Ax-b\|_2^2$ 占比非常小，主要误差产生就是一范数项。此外，Gurobi 模型显示的目标函数值为 0.8530970，这比用输出的 x 计算的目标函数值稍大，是因为计算可以发现优化结果中 z 要比 $|x|$ 大 10^{-6} 量级，这是算法的迭代终止条件导致的。

对于 Mosek，由于改变的是内置的 quadprog 函数，需要按照问题要求设置参数。两者的参数设置存在一些差别，如最小化模型不允许常数项，且二次项为 $\frac{1}{2}x^THx$ 而非 x^TQx 等。采用默认算法进行求解，输出的目标函数值约为 0.85317120，对应的 x 一范数约为 85.31494。可以发现，Mosek 迭代的最终效果并不如 Gurobi，z 与 $|x|$ 的误差也在 10^{-5}，比 Gurobi 高一个量级，用

$$\frac{\|x_G - x_M\|_1}{1+\|x_G\|_1}$$

刻画两个解的误差，结果在 10^{-4} 量级。

不过，多次测量时间可发现，Gurobi 求解的平均时间在 0.7 秒左右，而 Mosek 时间在 0.6 秒左右，相对更快。之后，我们将以 Gurobi 的结果作为**最优结果参考值**，每当算出新的结果后与此最优结果对比，若更好则更新，之后自己实现的三种算法的迭代收敛性等以此作为参照。

7.1.2 近端梯度下降

算法实现

对原问题的近端梯度下降指的是，在上一次迭代值 x^k 的近处将一范数外的项 $f(x)=\frac{1}{2}\|Ax-b\|_2^2$ 近似为线性函数 $\nabla f(x^k)^T(x-x^k)$，再加上保证近端的项 $\frac{1}{2\tau}\|x-x^k\|_2^2$，最终求解

$$x^{k+1}=\arg\min_x\left\{\mu\|x\|_1+\nabla f(x^k)^T(x-x^k)+\frac{1}{2\tau}\|x-x^k\|_2^2\right\}$$

配方并忽略常数项可写为

$$x^{k+1}=\arg\min_x\left\{\mu\|x\|_1+\frac{1}{2\tau}\|x-(x^k-\tau\nabla f(x^k))\|_2^2\right\}$$

直接计算次梯度可发现若 $\frac{|c|}{\tau}\le\mu$，则最小值点在 0 取到，否则 $c>0$ 时在 $c-\tau\mu$ 取到，$c<0$ 时在 $c+\tau\mu$ 取到。记函数

$$\mathrm{shrink}(c,\alpha)=\mathrm{sign}(c)\max(|c|-\alpha,0)$$

则上式最小值点最终可写为

$$x^{k+1}=\mathrm{shrink}(x^k-\tau\nabla f(x^k),\mu\tau)$$

再计算得 $\nabla f(x)=A^T(Ax-b)$ 即得到 (这称为 ISTA 迭代)

$$x^{k+1}=\mathrm{shrink}(x^k-\tau A^T(Ax^k-b),\mu\tau)$$

结果展示

不过，我们的例子中直接用此方法的效果并不好。一旦 τ 取到 10^{-3} 量级，算法就不再收敛，而取更小时会出现如图 3 的误差曲线 (纵轴为每步算出来的目标函数值与 Gurobi 最优结果的误差取 10 为底对数)。

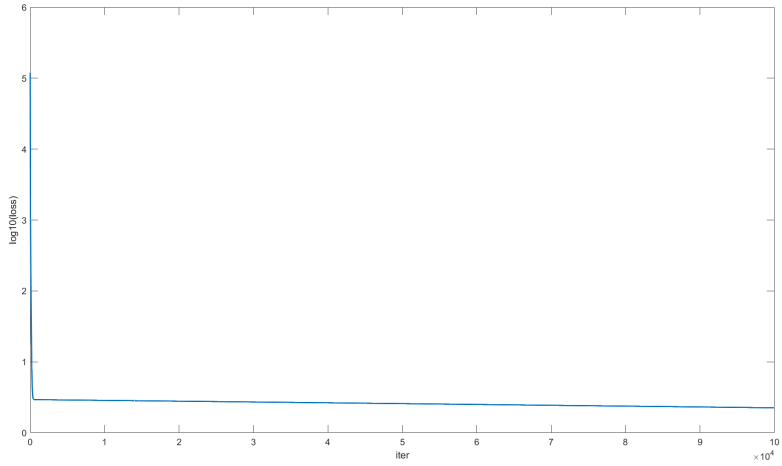

图 3: 普通近端梯度法误差曲线

可以发现，开始算法不收敛的主要问题在于初始误差过大，而初始误差进行很快的下降后，再采用 10^{-4} 量级的 τ 会导致迭代收敛速度过慢，这是因为基本的近端梯度下降本身只能保证误差正比于迭代次数的倒数。

我们设置迭代次数为 10^6 次，花费时间约为 24 秒，取定 $\tau=10^{-4}$ 得到的迭代结果为 3.101，与 Gurobi 最优 x_G 的误差为 1 左右 (这意味着几乎相差一倍)；取定 $\tau=3\times10^{-4}$ 得到的迭代结果为 2.100，与 Gurobi 最优 x_G 的误差也有 0.8 左右。我们最终取 $\tau=5\times10^{-4}$，设置最大迭代次数 10^7 次，若一范数更新不超过 10^{-6} 则停止迭代，得到的误差曲线如图 4 。

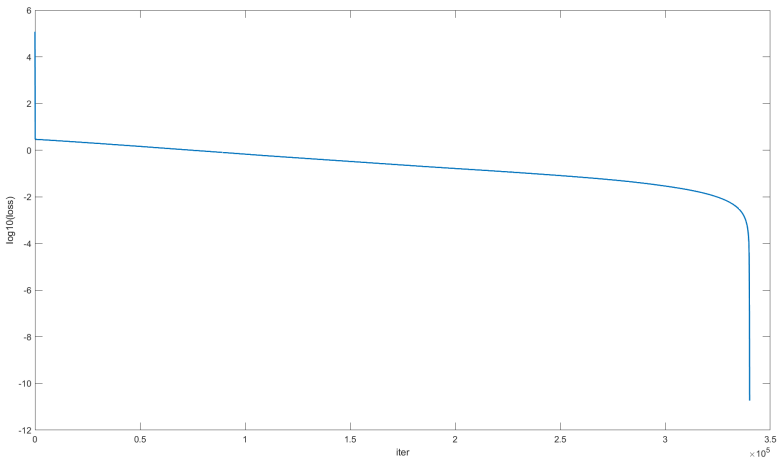

图 4: 近端梯度法最优参数下误差曲线

最终迭代次数为 340192 次，可以发现，最后一段与最优解的误差在急剧缩小。这种请款的部分原因是此算法得到的最优解为 0.8530633，好过 Gurobi 默认参数下的最优解，因此最优解进行了更新，在当前估计最优解附近可能获得并不真实的很好逼近效果。不过，事实上近端梯度下降法在解的近处 (以步长参数为标准) 本就会表现出非常好的收敛性，代价是，容差为 10^{-6} 已经需要花费 75 秒进行迭代了，想得到更高精度的解还需要远远更长的时间。

算法改进

利用讲义介绍的 FISTA 算法，我们将迭代过程改进为

```
r = A * y - b;
x = shrink(y - tau * (A' * r), tau * mu);
gamma_new = (1 + sqrt(1 + 4 * gamma^2)) / 2;
y = x + (gamma - 1) / gamma_new * (x - xold);
xold = x;
gamma = gamma_new;
```

通过给 x^k 的迭代补充一些额外的步长以加速收敛。

这时，取定 $\tau = 3 \times 10^{-4}$，仅需要 6554 次迭代，1.44 秒就可以得到 0.8530633 的最优解估算，误差曲线如图 5 ，远好于不加速的版本。真实最优解与 Gurobi 默认求解的最优解误差在 2.4×10^{-5} 左右。

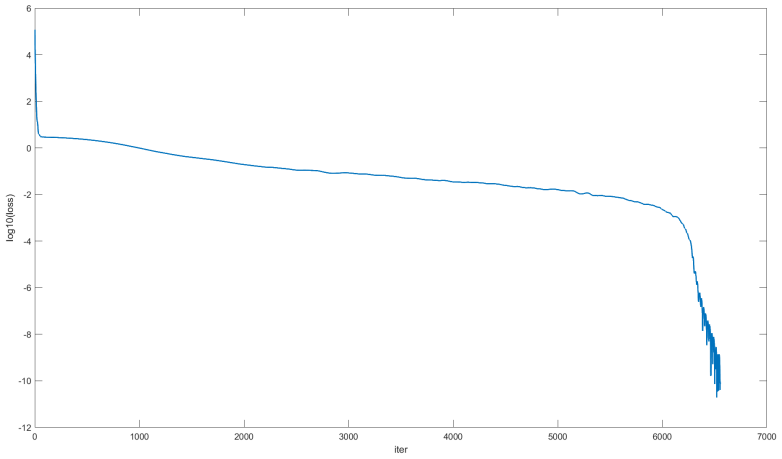

图 5: FISTA 误差曲线

并行设计

现在，我们针对优化后的 FISTA 算法，讨论如何利用并行加速计算，迭代过程即为上述六行代码。

在此算法中，最耗时间的部分是两次矩阵乘法 Ay 与 A^Tr，乘法次数均为 mn，其他部分都为 $O(n)$，在 m、n 都相对较大时可以忽略不计。但是，A^Tr 与 Ay 的计算又相互依赖，这两部分无法同时进行。

因此，并行只能对矩阵乘向量进行加速：设 A 的行为 $\alpha_1^T,\dots,\alpha_m^T$，则 Ay 的第 i 行为 α_i^Ty，而理论来说所有行可以同时计算，只涉及对 b 的读取不会冲突，因此有 K 个线程时可以将 Ay 的行等分为 K 部分计算，再把 A^Tr 的行等分为 K 部分计算，这样，最终单次迭代的时间可以近似为原本的 $\frac{1}{K}$。

神经网络方法

为了利用神经网络训练加速迭代，记 $\eta_\theta(x)=\mathrm{shrink}(x,\theta)$，我们重写 ISTA 迭代为

$$x^{k+1}=\eta_\theta(W^xx^k+W^bb),\quad \theta=\mu\tau,\quad W^x=I-\tau A^TA,\quad W^b=\tau A^T$$

但是，我们不直接计算过渡矩阵进行迭代，而是指定层数为 K，对应的参数为 K 个 θ、W^x 与 W^b，以下标 $0,1,\dots,K-1$ 区分，可以构造如下的网络：网络的输入为 x^0 与 b，第 i 层将输入 x^{i-1} 左乘 W^x_{i-1}，加上 b 左乘 W^b_{i-1}，然后以参数 θ_{i-1} 进行收缩，得到第 i 层输出 x^i，一共经历 K 层得到最终输出的 x^0 进行 K 次迭代后的结果 x^K。

为了让 x^K 尽量逼近真实的收敛结果，我们生成一系列稀疏向量 x_i^*，并计算 b_i^* 为 Ax_i^* 加一定的随机噪声 (以增强网络稳定性)。我们希望任何 x^0 出发，迭代后能以最快速度逼近真解 x_i^*，由此指定某 x^0，定义误差函数为

$$L(W^x_{0,\dots,K-1},W^b_{0,\dots,K-1},\theta_{0,\dots,K-1})=\sum_i\|\hat{x}^K(b_i^*)-x_i^*\|_2^2$$

这里 $\hat{x}^K(b_i^*)$ 代表以指定的 x^0 与 $b=b_i^*$ 作为输入后输出的结果。利用自动梯度法可对此误差函数进行梯度下降，从而对 $W^x_{0,\dots,K-1},W^b_{0,\dots,K-1},\theta_{0,\dots,K-1}$ 进行迭代更新。迭代完成后，每 K 次迭代不再直接计算，而是通过一次网络，由于网络是以 K 次迭代到真解为目标进行训练的，当生成数据集适当时，此网格可以远比真实 ISTA 的 K 次迭代迅速。

7.1.3 ADMM

原问题 ADMM

先考虑对原问题的 ADMM，将原问题写为适用 ADMM 的形式

$$\min_{x,z}\left\{\frac{1}{2}\|Ax-b\|_2^2+\mu\|z\|_1\right\}\qquad\text{s.t.}\quad x=z$$

其增广 Lagrange 函数为

$$L_\rho(x,z,y)=\frac{1}{2}\|Ax-b\|_2^2+\mu\|z\|_1+y^T(x-z)+\frac{\rho}{2}\|x-z\|_2^2$$

给定步长 τ，迭代格式为

$$\begin{aligned}x^{k+1}&=\arg\min_x L_\rho(x,z^k,y^k)\\z^{k+1}&=\arg\min_z L_\rho(x^{k+1},z,y^k)\\y^{k+1}&=y^k+\tau\rho(x^{k+1}-z^{k+1})\end{aligned}$$

由于 L_ρ 是对 x 的二次函数，可发现二次项系数 $\frac{1}{2}(A^TA+\rho I)$，一次项系数 $-A^Tb-\rho z+y$，从而由凸性计算驻点可发现第一个更新可写为

$$x^{k+1}=(A^TA+\rho I)^{-1}(A^Tb+\rho z^k-y^k)$$

而 z 的更新每个分量独立，为 $\mu|z_i| - y_i^T z_i + \frac{\rho}{2}(x_i - z_i)^2$ 的优化问题，配方类似之前推导可得

$$z^{k+1} = \text{shrink}(x^{k+1} + y^k/\rho, \mu/\rho)$$

这就得到了完整更新。更进一步地，由于 $n > m$，可验证 SMW 公式

$$(A^TA + \rho I)^{-1} = \rho^{-1}I - \rho^{-1}A^T(\rho I + AA^T)^{-1}A$$

若能提前将 $\rho I + AA^T$ 的 Cholesky 分解 LL^T 算出来，每次迭代中即需要计算三次矩阵乘向量、进行两次 m 阶三角方程组求解，复杂度 $3mn + m^2$。

对偶问题 ADMM

为了进一步降低复杂度，考虑其**对偶问题**。将原问题重新看成

$$\min_{x,r}\left\{\mu\|x\|_1 + \frac{1}{2}\|r\|_2^2\right\} \qquad \text{s.t.} \quad r = Ax - b$$

由于 Lagrange 函数为

$$L(x, r, \lambda) = \mu\|x\|_1 + \frac{1}{2}\|r\|_2^2 - \lambda^T(Ax - b - r)$$

考虑其在 x、r 任取时的最大值。x 对应的部分为 $\mu\|x\|_1 - (A^T\lambda)^T x$，当 $\|A^T\lambda\|_\infty > \mu$ 时考虑某个分量不断缩小可知最小值不存在，否则最小值为 0；r 对应的部分 $\frac{1}{2}\|r\|_2^2 + \lambda^T r$ 利用二次函数知识得最小值为 $-\frac{1}{2}\|\lambda\|_2^2$，由此可知其对偶问题为

$$\max_{\lambda}\left\{b^T\lambda - \frac{1}{2}\|\lambda\|_2^2\right\} \qquad \text{s.t.} \quad \|A^T\lambda\|_\infty \le \mu$$

由于我们只需要最优点 λ，将此问题进一步改写成

$$\min_{\lambda,s}\left\{-b^T\lambda + \frac{1}{2}\|\lambda\|_2^2 + I_{\|s\|_\infty \le \mu}(s)\right\} \qquad \text{s.t.} \quad s = A^T\lambda$$

这里 I 在符合要求的区域为 0，否则为 ∞。此时增广 Lagrange 函数为

$$L_\rho(\lambda, s, x) = -b^T\lambda + \frac{1}{2}\|\lambda\|_2^2 + I_{\|s\|_\infty \le \mu}(s) + x^T(A^T\lambda - s) + \frac{\rho}{2}\|A^T\lambda - s\|_2^2$$

给定步长 τ，迭代格式为

$$\lambda^{k+1} = \arg\min_{\lambda} L_\rho(\lambda, s^k, x^k)$$

$$s^{k+1} = \arg\min_{s} L_\rho(\lambda^{k+1}, s, x^k)$$

$$x^{k+1} = x^k + \tau\rho(A^T\lambda^{k+1} - s^{k+1})$$

由于 L_ρ 为 λ 的二次函数，再次计算二次项、一次项系数可得

$$\lambda^{k+1} = (\rho AA^T + I)^{-1}(\rho As^k - Ax^k + b) = (\rho AA^T + I)^{-1}(A(\rho s^k - x^k) + b)$$

对于 s 的更新，实际要求解的问题是

$$s^{k+1} = \arg\min_{\|s\|_\infty \le \mu}\left\{-(x^k)^T s + \frac{\rho}{2}\|A^T\lambda^{k+1} - s\|_2^2\right\}$$

由于右侧对 s 的每个分量独立，且为开口向上的二次函数，最小值在 $[-\mu, \mu]$ 中时直接取最小值，在 μ 右侧则区间中 μ 点最小，在 $-\mu$ 左侧则区间中 $-\mu$ 点最小，从而结果为

$$s^{k+1} = \text{proj}_{[-\mu,\mu]}(x^k/\rho + A^T\lambda^{k+1})$$

这里 proj 即代表将每个分量投影为 $[-\mu, \mu]$ 中最近的点。

由于 $A^T\lambda^{k+1}$ 的计算结果可以保存，若能提前将 $I+\rho AA^T$ 的 Cholesky 分解 LL^T 算出来 (这个预处理的复杂度为 m^3)，每次迭代中即需要计算两次矩阵乘向量、进行两次 m 阶三角方程组求解，复杂度 $2mn+m^2$，确实比原始问题更低。

结果展示

我们采用对偶 ADMM 方法进行实现。取定 $\rho=0.8$、$\tau=1.2$，并以**对偶间隙**

$$\mu\|x\|_1+\frac{1}{2}\|Ax-b\|_2^2-b^T\lambda+\frac{1}{2}\|\lambda\|_2^2$$

作为迭代结束判定，对偶间隙不超过 10^{-7} 时结束迭代，得到的误差曲线与对偶间隙收敛曲线如图 6 (纵轴仍为两种误差的 10 为底对数)。

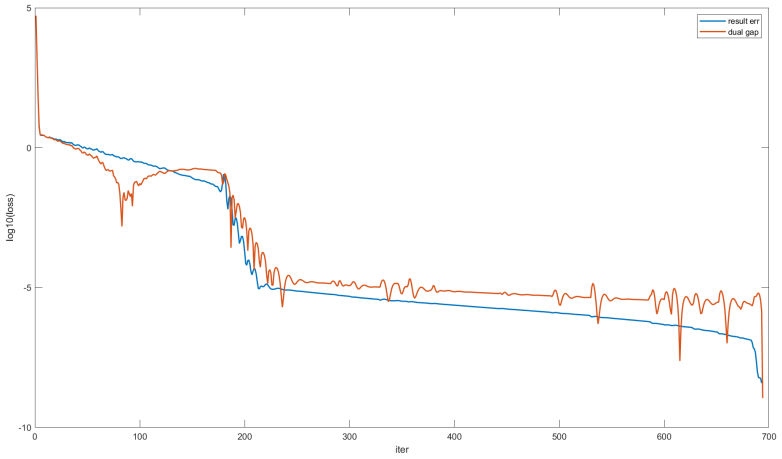

图 6: ADMM 误差与对偶间隙下降曲线

算法耗时 0.4 秒左右，进行了 694 次迭代，即得到了 0.8530642 的五位有效数字结果 (之所以没有更好的收敛，是由于此参数下振荡相对严重，适当调小 τ 可提高精度)，比 FISTA 的收敛速度更快，从这个例子上来说，同样精度解的计算速度事实上要快于用 Gurobi 进行二次规划求解。

并行设计

首先，对 s 和 x 的更新，可发现只要 $A^T\lambda$ 已经算出，s 与 x 就可各分量独立更新，由此可类似近端梯度中分线程进行 $A^T\lambda$ 的不同分量计算，并用于独立更新 s 与 x 的分量，就达成了 K 个线程时单次迭代时间近似为 $\frac{1}{K}$ 的 s、x 更新。

下面考虑 λ 的更新。$A(\rho s^k-x^k)+b$ 仍然可以分线程计算分量，于是主要的难点在于对 $(\rho AA^T+I)^{-1}\alpha$ 的计算。当 n 远大于 m 时，进行预处理后这部分的时间复杂度 m^2 比起计算 $A^T\lambda$ 等结果的 mn 数量级可以忽略不计。否则，m、n 都很大时，为了易于并行计算，我们可以考虑不精确求解，而是以迭代方法 (如 Jacobi 迭代) 进行近似。迭代方法中每次只需要计算矩阵乘法，由此可以与之前类似按分量进行并行计算，假设单步迭代中进行 l 次迭代求解，复杂度为 lm^2，K 线程并行后为 $\frac{lm^2}{K}$，从而得到了一个单步迭代复杂度 $\frac{1}{K}(lm^2+2mn)$、无需预处理的并行算法。

7.1.4 PDHG

算法实现

记 $f(x)=\mu\|x\|_1$、$h(z)=\frac{1}{2}\|z-b\|_2^2$ 原问题即

$$\min_x\{f(x)+h(Ax)\}$$

从而其化为鞍点问题形式为 (利用 h 是闭凸函数，$h^{**}=h$)

$$\min_x \max_z \{f(x)+z^TAx-h^*(z)\}$$

这里 $h^*(z)$ 为 $h(z)$ 的共轭函数，利用二次函数知识可得

$$h^*(z)=\sup_y\{z^Ty-h(y)\}=\max_y\left\{-\frac{1}{2}\|y-b\|_2^2+z^Ty\right\}=\frac{1}{2}\|z+b\|_2^2-\frac{1}{2}\|b\|_2^2=\frac{1}{2}\|z\|_2^2+b^Tz$$

利用 PDHG 的思路，我们需要交替最大化 z/最小化 x，并添加合适的近端项。具体来说，设最大化步长为 δ，最小化步长为 α，有

$$z^{k+1}=\arg\max_z\left\{f(x^k)+z^TAx^k-h^*(z)-\frac{1}{2\delta}\|z-z^k\|_2^2\right\}$$

$$x^{k+1}=\arg\min_x\left\{f(x)+(z^{k+1})^TAx-h^*(z^{k+1})+\frac{1}{2\alpha}\|x-x^k\|_2^2\right\}$$

消去无关的常数项并代入可得

$$z^{k+1}=\arg\max_z\left\{(Ax^k-b)^Tz-\frac{1}{2\delta}\|z-z^k\|_2^2-\frac{1}{2}\|z\|_2^2\right\}$$

$$x^{k+1}=\arg\min_x\left\{\mu\|x\|_1+(z^{k+1})^TAx+\frac{1}{2\alpha}\|x-x^k\|_2^2\right\}$$

第一个函数为各个分量独立的二次函数，第二个函数为各个分量独立的 $\mu|x|+a(x-h)^2$ 类型函数，与之前完全类似可求解得更新步骤

$$z^{k+1}=\frac{1}{\delta+1}(\delta(Ax^k-b)+z^k)=\frac{\delta}{\delta+1}(Ax^k-b)+\frac{1}{\delta+1}z^k$$

$$x^{k+1}=\text{shrink}(x^k-\alpha A^Tz^{k+1},\alpha\mu)$$

结果展示

取定 $\delta=0.03$、$\alpha=0.02$，以更新部分的一范数不超过 10^{-6} 为收敛标准，误差曲线如图 7 。

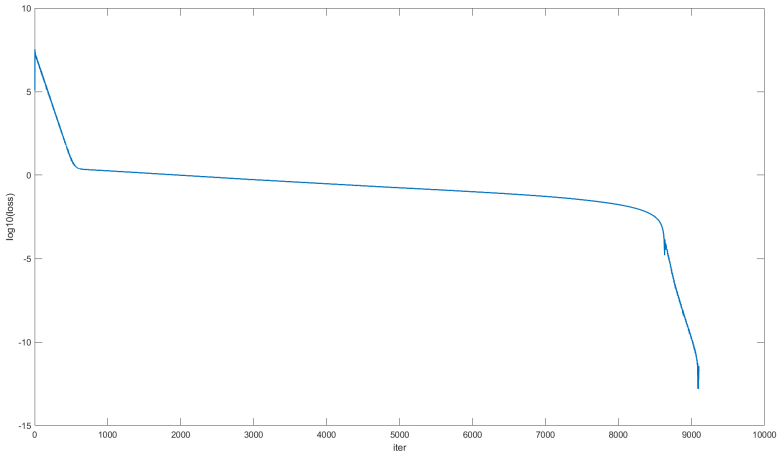

图 7: PDHG 误差下降曲线

进行了 9101 次迭代，历时 2.0 秒后，PDHG 算法收敛到了七位有效数字的最优结果 0.8530633。可以发现，它的误差表现出了与近端梯度下降非常类似的性质：在接近最优解时误差快速下降。此外，由于从原始问题对偶问题两个角度同时更新，它的允许步长比近端梯度下降高了两个量级，收敛速度远快于 ISTA 迭代。

并行设计

此算法中可以并行的部分与近端梯度下降法几乎完全相同：由于 x 与 z 相互依赖，计算 Ax 与 A^Tz 两件事无法同时进行，但对每件事利用矩阵乘法的分量特性可以由 K 个线程同时计算，而这就是全部 mn 量级的部分，因此总时间可以变为原本的 $\frac{1}{K}$。

7.2 核范数最优化

本部分考虑的优化问题为

$$\min_X \mu\|X\|_* + \sum_{(i,j)\in\Omega}(X_{ij}-M_{ij})^2$$

它是作为

$$\min_X \operatorname{rank} X \qquad \text{s.t.} \quad \forall(i,j)\in\Omega, X_{ij}=M_{ij}$$

的松弛而存在的。

其中 X 为 $m\times n$ 实矩阵，M 为随机生成的秩为 r 的矩阵，且 r 远小于 $\min(m,n)$。Ω 中包含 p 个分量，保证从这 p 个分量可以恢复出秩为 r 的矩阵。

为了方便之后的计算，我们约定一些记号：$\partial_X f$ 表示 f 对每个 X_{ij} 求导拼成的矩阵；内积 $\langle A,B\rangle$ 表示 A,B 对应位置元素乘积并求和，等同于 $\operatorname{tr}(A^TB)$，Frobenius 范数 $\|X\|_F^2=\langle X,X\rangle$；$\Omega$ 也可视为一个所有在其中的 (i,j) 分量为 1，其他为 0 的矩阵；Hadamard 积 $A*B$ 的每个分量为 A、B 对应分量乘积的结果。

7.2.1 近端梯度下降

算法实现

我们仍然采取二范数近似为附近一次函数与近端项结合的方案进行近端梯度下降算法的构造。仍记步长为 τ，设 $f(X)=\sum_{(i,j)\in\Omega}(X_{ij}-M_{ij})^2$，将其在 X^k 的附近近似为线性函数 $\left\langle\partial_X f(X^k), X-X^k\right\rangle$，则优化过程为

$$X^{k+1}=\arg\min_X\left\{\mu\|X\|_*+\left\langle\partial_X f(X^k),X-X^k\right\rangle+\frac{1}{2\tau}\|X-X^k\|_F^2\right\}$$

由于二次函数的常数项并不影响，利用 $\partial_X f(X)=2\Omega*(X-M)$，我们可以重新配方得到

$$X^{k+1}=\arg\min_X\left\{\mu\|X\|_*+\frac{1}{2\tau}\|X-\hat{X}^k\|_F^2\right\},\quad \hat{X}^k=X^k-2\tau(X^k-M)*\Omega$$

我们定义 $S_\tau(Y)$ 为如下的算子：构造 $Y=U\Sigma V^T$ 为其某个奇异值分解，将 Σ 的每个对角元 σ_i 替换为 $\operatorname{shrink}(\sigma_i,\tau)$ 后成为矩阵 Σ'，则 $S_\tau(Y)=U\Sigma'V^T$。利用教材中已证明的，取

$$X^{k+1}=S_{\mu\tau}(\hat{X}^k)$$

即为要求的最优点，且利用最优解唯一性 $S_{\mu\tau}$ 事实上是良好定义的 (每个 Y 像唯一)，由此可以实现迭代。与之前类似，我们也将**实现 FISTA 迭代以加速**。

由于并不存在其他的停止标准，我们每次迭代后将计算目标函数更新的量，若其小于容差则停止迭代。

结果展示

对 $m=n=40$、$p=480$、$r=3$ 的例子，考虑 $\mu=0.1$、0.01、0.001，设置容差为 10^{-8}，迭代次数上限 10000，步长 $\tau=0.01$，进行 ISTA 迭代与 FISTA 迭代的效果如下表：

需要补充几个注释：

表 4: 核范数近端梯度下降优化效果表

参考值		ISTA 迭代			FISTA 迭代		
μ	$\mu\|A\|_*$	迭代次数	时间 (s)	结果	迭代次数	时间 (s)	结果
0.1	13.535	10000	4.0	15.959	3691	1.5	13.196
0.01	1.3535	10000	4.1	1.8753	3620	1.6	1.3295
0.001	0.13535	10000	4.2	0.19519	9936	3.9	0.13332

- 之所以在左侧列出了 $\mu\|A\|_*$，是由于根据我们的生成方式，A 必然满足

$$\sum_{(i,j)\in\Omega}(A_{ij}-M_{ij})^2=0$$

  于是 $\mu\|A\|_*$ 就是 A 对应的目标函数值。由于 A 是低秩矩阵，真实的最优值应比 A 好且不会好太多。

- ISTA 的三次迭代都是到达次数上限中止了，得到的目标函数值也离最优较有距离。

- FISTA 的三次迭代都到达容差而停止了，由于容差设置，这说明至少在我们写出的五位有效数字范围内其为最优值。此最优值也可以印证第一条所说的性质。

- FISTA 与 ISTA 单次迭代的时间基本相同，均为 0.0004 秒左右。

- 可以发现，随着 μ 不断减小，得到的最优解与 $\mu\|A\|_*$ 将越来越接近，当 $\mu\to 0$ 时相当于强制 $(i,j)\in\Omega$ 时 $X_{ij}=M_{ij}$，$X=A$ 几乎为最优。

- 随着 μ 减小，迭代计算的时间稍有提升，可能是收缩过程非零元个数更少导致需要计算的次数更多导致的。

作出两种迭代的误差曲线如图 8 。这里相对误差是指最优值除以 $\mu\|A\|_*$ 再减 1 的结果，以保证不同 μ 时的相对误差量级一致，小于 0 代表结果较好。从图中也可以明显看出，FISTA 迭代有着远比 ISTA 优秀的收敛性。

当 m、n 增大时，所耗的时间会迅速增加。当 $m=n=100$、$p=3000$、$r=2$ 时，仍以相同参数进行迭代，可发现单次迭代的耗时是之前的 5 倍左右。此外，虽然 FISTA 方法仍然能收敛且收敛次数一致，此时收敛的结果可能会大于 $\mu\|A\|_*$，如 $\mu=0.001$ 时 $\mu\|A\|_*=0.182$，但收敛结果为 0.199。也即，将 μ 取得过小也会影响迭代结果。此时作出相对误差曲线如图 9 ，可发现由于 A 的秩更低、参数更多，可能会更难以收敛。

7.2.2 ADMM

算法实现

将原问题写为适用 ADMM 的形式

$$\min_{X,Z}\left\{\|(X-M)*\Omega\|_F^2+\mu\|Z\|_*\right\}\qquad \text{s.t.}\quad X=Z$$

其增广 Lagrange 函数为

$$L_\rho(X,Z,Y)=\|(X-M)*\Omega\|_F^2+\mu\|Z\|_*+\langle Y,X-Z\rangle+\frac{\rho}{2}\|X-Z\|_F^2$$

给定步长 τ，迭代格式为

$$X^{k+1}=\arg\min_X L_\rho(X,Z^k,Y^k)$$

$$Z^{k+1}=\arg\min_Z L_\rho(X^{k+1},Z,Y^k)$$

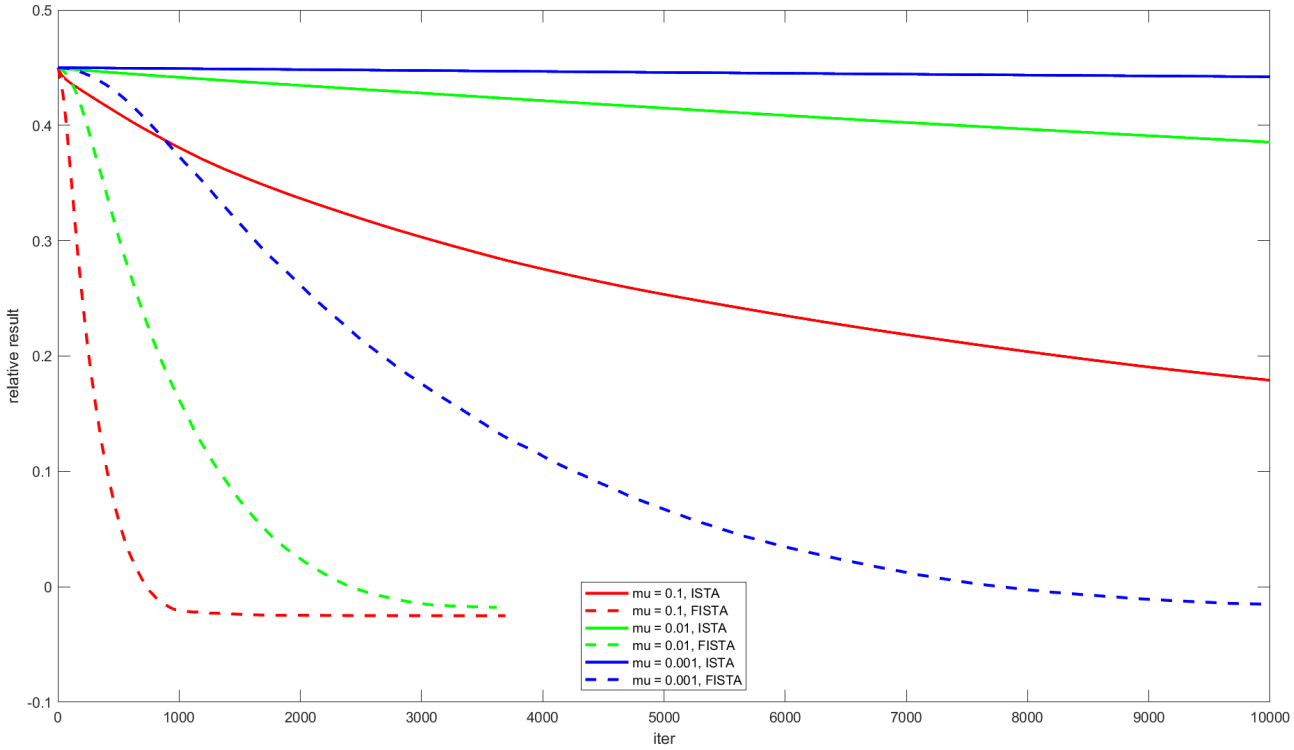

图 8: $m = n = 40$ 时 ISTA 与 FISTA 相对误差下降曲线

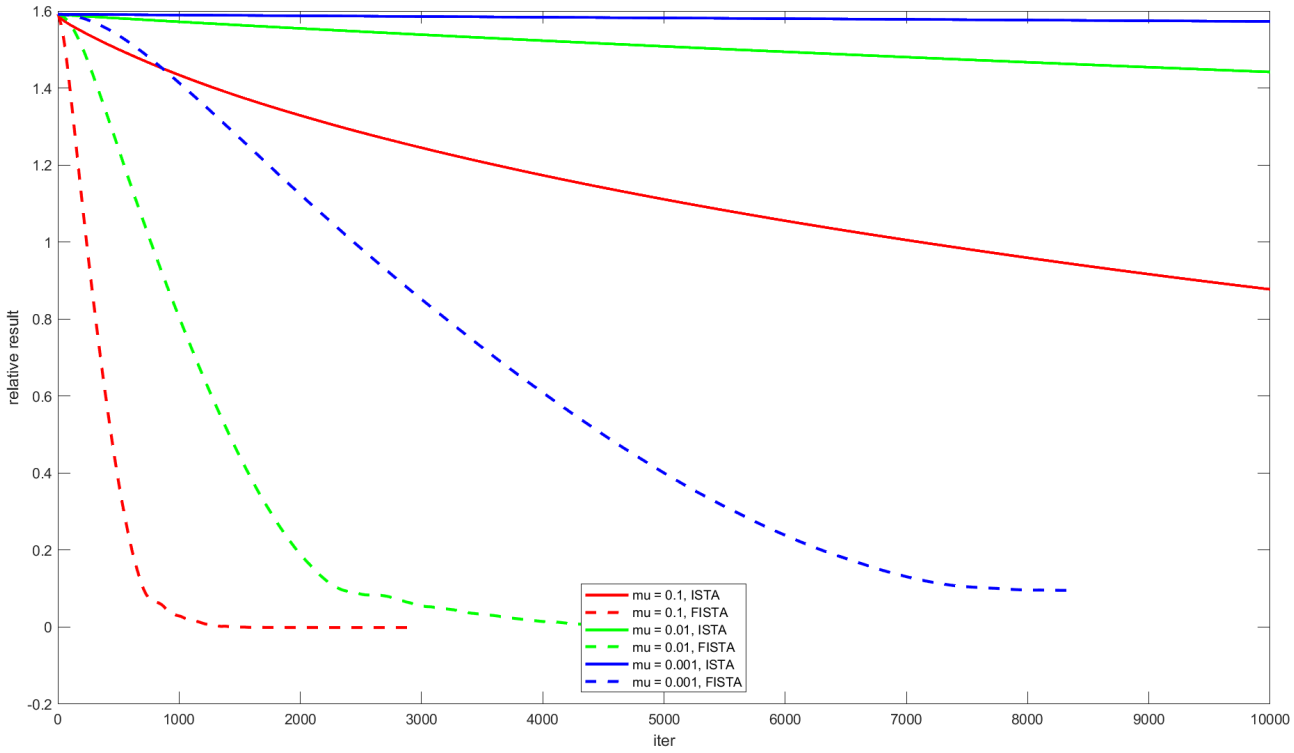

图 9: $m = n = 100$ 时 ISTA 与 FISTA 相对误差下降曲线

$$Y^{k+1} = Y^k + \tau\rho(X^{k+1} - Z^{k+1})$$

由于对 X 每个分量实质上是独立的二次函数，可写出更新后的

$$X_{ij}^{k+1} = \begin{cases} \frac{1}{\rho+2}(2M_{ij} + \rho Z_{ij}^k - Y_{ij}^k) & (i,j) \in \Omega \\ \frac{1}{\rho}(\rho Z_{ij}^k - Y_{ij}^k) & (i,j) \notin \Omega \end{cases}$$

由此设 $\mathbf{1}$ 为全 1 矩阵，上述过程可写为

$$X^{k+1} = (\rho Z^k - Y^k + 2M * \Omega) \div (\rho \mathbf{1} + 2\Omega)$$

这里除法也表示逐元素相除。

对于 Z^{k+1}，与近端梯度下降相同推导可将问题写为

$$Z^{k+1} = \arg\min_Z \left\{ \mu \|Z\|_* + \frac{\rho}{2} \|Z - (X^{k+1} + Y^k/\rho)\|_F^2 \right\} = S_{\mu/\rho}(X^{k+1} + Y^k/\rho)$$

结果展示

首先，取 $\rho = 0.01$、$\tau = 1$，容差为 10^{-8}，用 ADMM 方法进行迭代的效果如下表，上三行为 $m = n = 40$、$p = 480$、$r = 3$ 的例子，下三行为 $m = n = 100$、$p = 3000$、$r = 2$ 的例子：

表 5: 核范数近端 ADMM 优化效果表

μ	$\mu\|A\|_*$	迭代次数	时间 (s)	结果
0.1	13.535	1194	0.55	13.195
0.01	1.3535	527	0.22	1.3245
0.001	0.13535	2732	1.02	0.13250
0.1	18.200	2149	4.97	18.181
0.01	1.8200	1462	3.19	1.8198
0.001	0.18200	952	1.95	0.18201

可以发现，ADMM 方法不但收敛速度比起近端梯度方法更快，收敛效果也更好。$m = n = 100$ 时，它的相对误差下降曲线如图 10 。

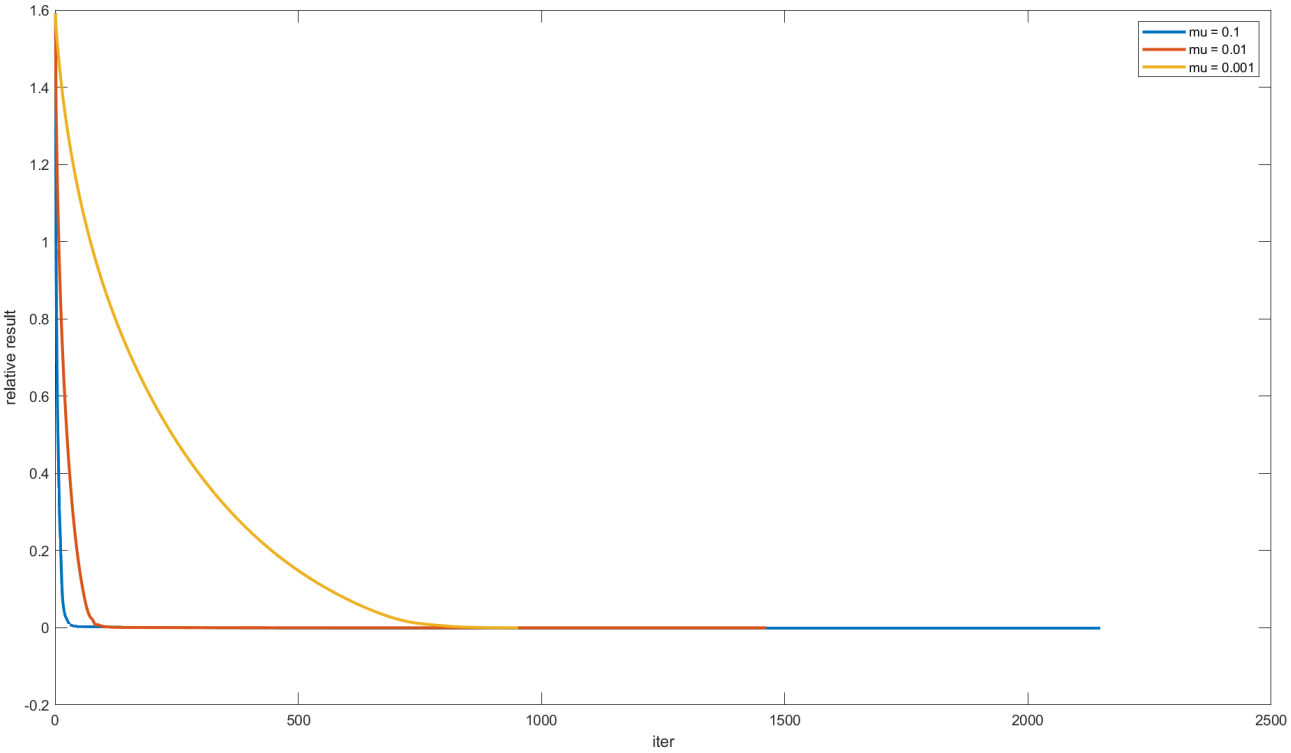

图 10: $m = n = 100$ 时 ADMM 相对误差下降曲线

比起近端梯度方法，ADMM 方法的误差很快下降到了 0 附近，并迅速稳定。即使对 m、n 较大的例子，当 $\mu = 0.001$ 时，它也没有收敛到并非最优的位置。

7.3 讨论

7.3.1 总结

对于一范数最优化的 LASSO 问题，我们用 Gurobi 求解器、Mosek 求解器进行了求解，并实现了 ISTA 迭代、FISTA 迭代、ADMM 算法与 PDHG 算法。就此问题而言，ADMM 算法表现出了最好的收敛性，能以最快速度收敛到指定有效数字的结果，甚至快过直接调用二次规划的求解器。虽然 ISTA 迭代本身很慢，但进行合适的加速处理后它能表现出很好的收敛性，成为了手动实现算法中收敛速度第二快的。PDHG 利用引入对偶问题，在基础版本就能取到较高的步长，有很大的推广、优化空间。此外，无论是 ISTA、FISTA 还是 PDHG，都属于近端方法，它们的特点是在误差很大与接近最优时都能够快速下降误差，但在中等误差时的下降速度较慢。除了这些，我们还讨论如何利用分布式计算加速各种算法与将近端梯度法展开为某种神经网络以进行训练的优化方案，这些方法都是为了解决直接调用算法时在大规模情况收敛速度不足的问题。

对于低秩恢复中出现的核范数最优化问题，我们实现了近端梯度下降与 ADMM 两种方法。可以发现，虽然 FISTA 迭代有着较快的结果，在这类问题还是远不如 ADMM 收敛迅速，也不如 ADMM 精确。当约束项 $\sum_{(i,j)\in\Omega}(X_{ij}-M_{ij})^2$ 的系数比例趋于无穷时，利用 ADMM 算法可以发现，X 最终将趋于真实问题的最优解 A，这就得到了足够好的低秩恢复结果。

7.3.2 文件列表

```
l1_ADMM.m   LASSO问题 ADMM 算法求解
l1_gurobi.m LASSO问题 调用 Gurobi 求解
l1_mosek.m  LASSO问题 调用 Mosek 求解
l1_PDHG.m   LASSO问题 PDHG 算法求解
l1_prox.m   LASSO问题 近端梯度下降方法求解
n_ADMM.m    核范数问题 ADMM 算法求解
n_prox.m    核范数问题 近端梯度下降方法求解
Test_L1.m   LASSO问题 测试程序
Test_N.m    核范数问题 测试程序
```

运行时直接运行两个测试程序即可。

8 整数规划作业

1. (a) 对于其线性松弛，由于有五个独立约束，利用几何观察考虑其中两约束交点可发现最优值当且仅当 $-3x_1+4x_2=4$、$3x_1+2x_2=11$ 时取到，也即 $x_1=2$、$x_2=2.5$ 时取到，为 7。
对原问题，由于 $x_1\geq 0$、$x_2\geq 0$ 且 $3x_1+2x_2\leq 11$，对区域中的整点尝试可发现最优值当且仅当 $x_1=x_2=2$ 时取到，为 6。

   (b) 如图 11 所示，灰色区域交出的五边形为线性规划问题的可行域，绿色五边形为整数规划可行点的凸包。

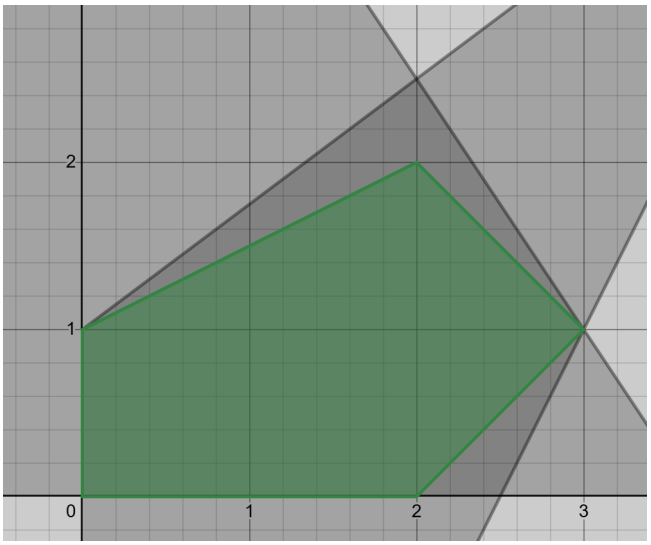

图 11: 线性规划可行域与整数规划可行点凸包

   (c) 先将线性规划问题的约束写为标准形，即求 x_1+2x_2 最大值使得

$$\begin{pmatrix} -3 & 4 & 1 & 0 & 0 \\ 3 & 2 & 0 & 1 & 0 \\ 2 & -1 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ y_1 \\ y_2 \\ y_3 \end{pmatrix} = \begin{pmatrix} 4 \\ 11 \\ 5 \end{pmatrix}$$

且 $x_1,x_2,y_1,y_2,y_3\geq 0$。
由于已知最优解为 $x_1=2$、$x_2=2.5$，对应 $y_1=y_2=0$、$y_3=3.5$，于是由系数矩阵的 1、2、5 列作为最优基，将约束改写为

$$\begin{pmatrix} 1 & 0 & -\frac{1}{9} & \frac{2}{9} & 0 \\ 0 & 1 & \frac{1}{6} & \frac{1}{6} & 0 \\ 0 & 0 & \frac{7}{18} & -\frac{5}{18} & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ y_1 \\ y_2 \\ y_3 \end{pmatrix} = \begin{pmatrix} 2 \\ \frac{5}{2} \\ \frac{7}{2} \end{pmatrix}$$

将右端非整数的两行进行取整可得

$$x_2\leq 2,\quad -y_2+y_3\leq 3$$

将对应约束与其作差得到割平面

$$y_1+y_2\geq 3,\quad 7y_1+13y_2\geq 9$$

可发现第二个约束事实上被第一个约束包含，从而割平面即为 $y_1 + y_2 \geq 3$，或化为原不等式约束的 $x_2 \leq 2$。

类似上方讨论，由于只要每次得到的新的线性规划问题的最优解不是整数规划的最优解，割平面法就可以改善可行域，且考虑到系数矩阵中取可行基对应矩阵的行列式存在最大值，每次改善的幅度有下界，因此有限次割平面后可以将可行域选取为整数规划问题可行解的凸包，此时求解线性规划即得最优解。

(d) 由于线性规划问题的最优解为 $x_1 = 2$、$x_2 = 2.5$，利用 x_2 进行子问题拆分：

- 原问题添加约束 $x_2 \geq 3$，可发现线性松弛版本无可行解。
- 原问题添加约束 $x_2 \leq 2$，可发现线性松弛版本最优解为 $x_1 = \frac{7}{3}$、$x_2 = 2$，继续拆分：
  - 进一步添加约束 $x_1 \leq 2$，可发现线性松弛版本最优解为 $x_1 = x_2 = 2$，最优值 6。
  - 进一步添加约束 $x_1 \geq 3$，可发现线性松弛版本可行域只有一点 $x_1 = 3$、$x_2 = 1$，最优值 5。

由于划分到的每个情况已经线性松弛最优解不存在或为整数，综合即得原问题最优解 $x_1 = x_2 = 2$，最优值 6。

(e) 对应的 Lagrange 函数

$$Z(\lambda) = \max_{x \in X} (1 + 3\lambda)x_1 + (2 - 4\lambda)x_2 + 4\lambda, \quad X = \{x \in \mathbb{Z}_+^2 \mid 3x_1 + 2x_2 \leq 11, 2x_1 - x_2 \leq 5\}$$

利用几何关系可知 X 中整点的凸包为 $(0,0)$、$(0,5)$、$(1,4)$、$(3,1)$、$(2,0)$ 构成的五边形，从而根据讲义定理可知 Z_D 为此凸包与 $-3x_1 + 4x_2 \leq 4$ 交集中的线性规划最优值，计算得为 $x_1 = 2$、$x_2 = 2.5$ 时取到的 7。

(f) 对应的 Lagrange 函数

$$Z(\lambda) = \max_{x \in X} (1 - 2\lambda)x_1 + (2 + \lambda)x_2 + 5\lambda, \quad X = \{x \in \mathbb{Z}_+^2 \mid 3x_1 + 2x_2 \leq 11, -3x_1 + 4x_2 \leq 4\}$$

利用几何关系可知 X 中整点的凸包为 $(0,0)$、$(0,1)$、$(2,2)$、$(3,1)$、$(3,0)$ 构成的五边形，从而根据讲义定理可知 Z_D 为此凸包与 $2x_1 - x_2 \leq 5$ 交集中的线性规划最优值，计算得为 $x_1 = 2$、$x_2 = 2$ 时取到的 6。

9 次模函数作业

1. 教材 2 习题 5.3

   * 这里写出 $\{s,v\}$ 时默认了 $s\neq v$，否则结论未必成立。此外，还需要假设 X 元素个数**有限**，否则可在 $\mathbb{N}$ 上构造函数 F 当子集为有限集时值等于势，否则值为 0，此函数满足题中命题 (讨论 A 有限或可数)，但不为次模函数 (A 为奇数，B 为偶数集合与 1 并集)。事实上，与此类似，教材命题 5.1(2) 也需要假设 X 为有限集。

   若 F 是次模函数，由 $s\neq v$ 且 $s,v\notin A$ 可知 $A\subset A\cup\{v\}$ 且 $s\notin A\cup\{v\}$，从而根据教材命题 5.1(2) 可知

   $$F(A\cup\{s\})-F(A)\geq F(A\cup\{s,v\})-F(A\cup\{v\})$$

   若原命题成立，我们来证明教材命题 5.1(2) 以说明其为次模函数。由 $B-A$ 为有限集，设其为 $\{v_1,\dots,v_k\}$，则由条件 $v_1,\dots,v_k,s$ 都是 $X\backslash A$ 中的不同元素，从而

   $$F(A\cup\{s\})-F(A)\geq F(A\cup\{v_1,s\})-F(A\cup\{v_1\})\geq F(A\cup\{v_1,v_2,s\})-F(A\cup\{v_1,v_2\})\geq\dots$$

   以此重复，直到 v_k 也加入，即得到 $F(A\cup\{s\})-F(A)\geq F(B\cup\{s\})-F(B)$。

2. 教材 2 习题 5.4

   * 与上题类似须假设 X 为有限集。

   根据定义，差分函数单调递减也即对任何 $A\subset B$、$x\in X\backslash(A\cup B)=X\backslash B$ 有

   $$F(A\cup\{x\})-F(A)\geq F(B\cup\{x\})-F(B)$$

   这即是命题 5.1(2)，从而等价性成立。

3. 教材 2 习题 5.6

   不妨设 $X=\{1,\dots,n\}$。

   先证明引理，若 $s\in\mathcal{B}(F)$，其第 p 个分量 s_p 最小值即为 $F(X)-F(X\backslash\{p\})$。

   首先，由 $\mathcal{B}(F)$ 定义可知

   $$s_p=\sum_{i=1}^{n}s_i-\sum_{i\neq p}s_i=F(X)-\sum_{i\neq p}s_i\geq F(X)-F(X\backslash\{p\})$$

   只要证明其可取到即可。下面设 $s_p=F(X)-F(X\backslash\{p\})$，由于 F 限制在 $X\backslash\{p\}$ 上也是次模函数，$\mathcal{B}(F_{X\backslash\{p\}})$ 非空，因此可以得到 $n-1$ 个分量 $s_i,i\neq p$ 满足总和为 $F(X\backslash\{p\})$ 且其中下标集合 I 的部分和不超过 $F(I)$，我们下面证明这 $n-1$ 个分量与 s_p 拼成的向量 s 符合要求。首先，根据定义可知 s 的所有分量总和为 $F(X\backslash\{p\})+F(X)-F(X\backslash\{p\})=F(X)$，符合要求。此外，对任何 $I\subset X$，若 $I\subset X\backslash\{p\}$ 则已经成立，否则设 $I=I'\cup\{p\}$，$I'\subset X\backslash\{p\}$，利用次模函数性质有

   $$F(I)\geq F(I')+F(X)-F(X\backslash\{p\})=F(I')+s_p\geq\sum_{i\in I'}s_i+s_p=\sum_{i\in I}s_i$$

   从而得证。

   利用引理，若 $\mathcal{B}(F)\not\subset\mathbb{R}_+^n$，必有某个 s_p 最小值小于 0，于是对应的 $F(X)$，与单调增矛盾；反之，若 $\mathcal{B}(F)\subset\mathbb{R}_+^n$，类似可得所有 s_p 最小值 ≥ 0，从而所有 $F(X)-F(X\backslash\{q\})\geq 0$，利用次模性，对任何不包含 q 的 A 有

   $$F(A\cup\{q\})-F(A)\geq F(X)-F(X\backslash\{q\})\geq 0$$

   由于上述讨论对任何 q 成立，若 $A\subset B$，考虑将 $B-A$ 中的元素逐个添入 A 中可知单调性成立。

10 随机优化作业与实验报告

10.1 书面作业

1. (a) 我们用 E_k 表示 x^k 固定时的期望，直接展开计算有

$$E_k[\|x^{k+1}-x^*\|^2] = \|x^k-x^*\|^2 + 2\alpha_k E_k[g^k]^T(x^*-x^k) + \alpha_k^2 E_k[\|g^k\|^2]$$

根据随机次梯度法定义，$E_k[g^k]$ 即为 x^k 处的真实次梯度，即其属于 $\partial f(x^k)$，从而根据闭凸函数性质

$$E_k[g^k]^T(x^*-x^k) \le f(x^*) - f(x^k)$$

再利用条件最后一项可放大为 M^2，从而

$$E_k[\|x^{k+1}-x^*\|^2] \le \|x^k-x^*\|^2 + 2\alpha_k(f(x^*)-f(x^k)) + \alpha_k^2 M^2$$

利用全期望公式，两边再对 x^k 取期望即得到真实期望，再移项得到

$$E[\|x^{k+1}-x^*\|^2] - E[\|x^k-x^*\|^2] \le 2\alpha_k E[(f(x^*)-f(x^k))] + \alpha_k^2 M^2$$

对 k 从 1 到 K 求和即得到

$$E[\|x^{K+1}-x^*\|^2] - E[\|x^1-x^*\|^2] \le 2\sum_{k=1}^K \alpha_k E[(f(x^*)-f(x^k))] + \sum_{k=1}^K \alpha_k^2 M^2$$

移项并将 $E[\|x^{K+1}-x^*\|^2]$ 放缩为 0 得证。

(b) 利用凸函数定义可发现

$$A_K(f(\bar{x}_K)-f(x^*)) \le \sum_{k=1}^K \alpha_k(f(x^k)-f(x^*))$$

从而根据 (a) 有

$$E[f(\bar{x}_K)-f(x^*)] \le \frac{1}{A_K}\sum_{k=1}^K \alpha_k E[f(x^k)-f(x^*)] \le \frac{E[\|x^1-x^*\|^2] + \sum_{k=1}^K \alpha_k^2 M^2}{2A_K}$$

取 $D = E[\|x^1-x^*\|^2]$ 即符合结论形式。

10.2 算法实现

10.2.1 问题与算法

首先，根据背景，这是一个 SVM 模型，因此我们假定**训练集**样本数量为 n，每个样本 $x_i \in \mathbb{R}^d$ 有 d 个特征，其对应的标签 $y_i \in \{-1, 1\}$ (covtype 数据集的标签为 1、2，将其映射为 -1 与 1)。求解优化问题

$$\min_{w\in\mathbb{R}^d} \frac{1}{n}\sum_{i=1}^n f_i(w) + \lambda\|w\|^2, \quad f_i(w) = 1 - \tanh(y_i w^T x_i)$$

得到最优的 w 后，我们最终对样本 x 标签预测为 $y = \mathrm{sign}(w^T x)$，这里当 $w^T x = 0$ 时规定 sign 结果为 1 以保证标签在 ± 1 中。我们以预测正确的标签占所有标签的比例表示准确率，并考虑训练集准确率与测试集准确率。

为了利用随机下降方法进行优化，我们将目标函数改写为

$$\min_{w\in\mathbb{R}^d} h(w), \quad \frac{1}{n}\sum_{i=1}^n h_i(w), \quad h_i(w) = f_i(w) + \lambda\|w\|^2$$

计算可得

$$\nabla h_i(w) = y_i(\tanh^2(y_i w^T x_i) - 1)x_i + 2\lambda w$$

下面介绍实现的四种算法的迭代过程，为方便起见，假设初始向量 w^0 取为零向量，下方的 $\odot$ 表示逐元素乘法，向量作为分母也表示逐元素取倒数：

1. Adagrad

   - 给定批次大小 b，保正系数 $\varepsilon > 0$，学习率 $\alpha > 0$。
   - 初始化迭代次数 k 为 0，G^{-1} 为零向量。
   - 从 1 到 n 中随机抽取 b 个下标构成集合 $\mathcal{I}_k$，将 w^k 处梯度 $\nabla h(w^k)$ 的估计值 g^k 取为

   $$g^k = \frac{1}{b}\sum_{i\in\mathcal{I}_k} \nabla h_i(w^k)$$

   - 计算累计的 $G^k = G^{k-1} + g^k \odot g^k$。
   - 更新 ($\mathbf{1}$ 表示全 1 向量)

   $$x^{k+1} = x^k - \frac{\alpha}{\sqrt{G^k + \varepsilon\mathbf{1}}} \odot g^k$$

   - 若 $\|g^k\|$ 充分小或达到最大次数则返回，否则令 $k \to k+1$，回到第三步。

2. Adam

   - 给定批次大小 b，保正系数 $\varepsilon > 0$，更新系数 $\rho_1, \rho_2 \in (0,1)$，学习率 $\alpha > 0$。
   - 初始化迭代次数 k 为 0，动量项 S^{-1}、二阶项 M^{-1} 均为零向量。
   - 从 1 到 n 中随机抽取 b 个下标构成集合 $\mathcal{I}_k$，将 w^k 处梯度 $\nabla h(w^k)$ 的估计值 g^k 取为

   $$g^k = \frac{1}{b}\sum_{i\in\mathcal{I}_k} \nabla h_i(w^k)$$

   - 更新动量项与二阶项

   $$S^k = \rho_1 S^{k-1} + (1-\rho_1)g^k, \quad M^k = \rho_2 M^{k-1} + (1-\rho_2)g^k \odot g^k$$

   - 计算修正 (ρ_1、ρ_2 上标表示次方)

   $$\hat{S}^k = \frac{1}{1-\rho_1^k} S^k, \quad \hat{M}^k = \frac{1}{1-\rho_2^k} M^k$$

   - 更新 ($\mathbf{1}$ 表示全 1 向量)

   $$x^{k+1} = x^k - \frac{\alpha}{\sqrt{\hat{M}^k + \varepsilon\mathbf{1}}} \odot \hat{S}^k$$

   - 若 $\|g^k\|$ 充分小或达到最大次数则返回，否则令 $k \to k+1$，回到第三步。

3. SVRG

   - 给定批次大小 b，重置次数 m，学习率 $\alpha > 0$。
   - 初始化迭代次数 k 为 0。
   - 设 $j = [k/m]$，代表第 j 次重置后。
   - 若 k 是 m 的倍数，记录当前的 $w_M^j = w^k$，并计算其全梯度

   $$g_M^j = \frac{1}{n}\sum_{i=1}^{n} \nabla h_i(w_M^j)$$

- 从 1 到 n 中随机抽取 b 个下标构成集合 $\mathcal{I}_k$，将 w^k 处梯度 $\nabla h(w^k)$ 的估计值 g^k 取为

$$g^k = \frac{1}{b}\sum_{i\in\mathcal{I}_k}\nabla h_i(w^k)$$

  并计算 w_M^j 处梯度的估计值

$$\hat{g}^k = \frac{1}{b}\sum_{i\in\mathcal{I}_k}\nabla h_i(w_M^j)$$

- 用修正的梯度进行更新

$$w^{k+1} = w^k - \alpha(g^k - (\hat{g}^k - g_M^j))$$

- 若 $\|g^k\|$ 充分小或达到最大次数则返回，否则令 $k \to k+1$，回到第三步。

4. 随机拟牛顿法

由于往往训练样本个数极多，记录矩阵进行运算的开销是无法接受的，因此采取有限记忆拟牛顿法，即 L-BFGS 策略进行迭代。总体迭代策略为：

- 给定批次大小 b，记忆长度 m，更新周期 f，学习率 $\alpha > 0$。
- 初始化迭代次数 k 为 0，记忆 s_m、y_m 为 m 个 d 维向量构成的向量组 (s_m^i、y_m^i 表示其中第 i 个向量)，当前有效长度 $l = 0$。
- 从 1 到 n 中随机抽取 b 个下标构成集合 $\mathcal{I}_k$，将 w^k 处梯度 $\nabla h(w^k)$ 的估计值 g^k 取为

$$g^k = \frac{1}{b}\sum_{i\in\mathcal{I}_k}\nabla h_i(w^k)$$

- 利用 L-BFGS 方法从 s_m、y_m、l 与 g^k 生成的拟牛顿步 $H^k g^k$ (注意下述方法中只会涉及向量的计算，无需真的存储矩阵)：
  - 若 $l = 0$，$H^k = I$，于是 $H^k g^k = g^k$。
  - 否则，设

$$\hat{g}^k = \left(I - \frac{y^k(s^k)^T}{(s^k)^T y^k}\right)g^k = g^k - \frac{(s^k)^T g^k}{(s^k)^T y^k}y^k$$

    递归计算

$$H^k g^k = \left(I - \frac{s^k(y^k)^T}{(s^k)^T y^k}\right)\hat{H}^k\hat{g}^k + \frac{s^k(s^k)^T}{(s^k)^T y^k}g^k = \hat{H}^k\hat{g}^k - \frac{(y^k)^T(\hat{H}^k\hat{g}^k)}{(s^k)^T y^k}s^k + \frac{(s^k)^T g^k}{(s^k)^T y^k}s^k$$

    这里 $\hat{H}^k\hat{g}^k$ 表示由 s_m、y_m、$l-1$ 与 $\hat{g}^k$ 生成的拟牛顿步。

- 更新

$$x^{k+1} = x^k + s^k, \quad s^k = -\alpha H^k g^k$$

- 若 $k-1$ 是 f 的倍数，更新记忆：
  - 计算

$$y^k = \frac{1}{b}\sum_{i\in\mathcal{I}_k}\nabla h_i(w^{k+1}) - g^k$$

  - 若 $l \neq m$，令 $l = l+1$，否则将 s_m 的第 2 到 m 个向量移至第 1 到 $m-1$ 个，空出最后一个，并对 y_m 相同操作。
  - 更新

$$s_m^l = s^k, \quad y_m^l = y^k$$

- 若 $\|g^k\|$ 充分小或达到最大次数则返回，否则令 $k \to k+1$，回到第三步。

10.2.2 代码文件结构

adagrad.m	Adagrad 算法实现
adam.m	Adam 算法实现
covtype_scale.data	Covtype 数据集
gisette_scale.data	Gisette 训练集
gisette_scale_t.data	Gisette 测试集
lbfgs.m	LBFGS 算法实现
libsvmread.mexw64	数据集读取函数（来自 LIBSVM 包）
svrg.m	SVRG 算法实现
test_c.m	Covtype 上的训练结果
test_g.m	Gisette 上的训练结果

由于数据集文件过大，不附在邮件中，其为实验要求中所给网页里的文件下载后解压、重命名得到。此外，libsvmread 函数来自网页里链接中的 LIBSVM 包，编译其 MATLAB 接口即得到，并非个人完成。

将三个 data 文件准备好放入文件夹后，直接在目录中运行两个 test 脚本即可得到运行结果，各参数含义见四个算法文件的注释。对于 Contype 测试集，test 脚本首先进行了预处理：从数据中随机选出了 70% 作为训练集，剩余 30% 作为测试集。

10.3 结果展示

Covtype 数据集的训练集大小，测试集大小，特征个数为 54；Gisette 数据集的训练集大小 6000，测试集大小 1000，特征个数为 5000。由于 Covtype 数据集的特征较少，无需很强的正则化，取定 $\lambda = 0.001$，而 Gisette 的特征极多，且有干扰特征，需要较强的正则化，取定 $\lambda = 10$。下面先介绍四种算法的特点。

对 Adagrad 算法，取定合适参数迭代效果如图 12 。上方为 Covtype，下方为 Gisette，左侧为准确率曲线，右侧为目标函数值曲线。蓝色实线、虚线为训练集与测试集的情况，黄线为迭代时间。在 Covtype 的最终准确率达到 0.7，Gisette 的最终准确率接近 0.9，且函数值确实下降至收敛，可以验证正确性。

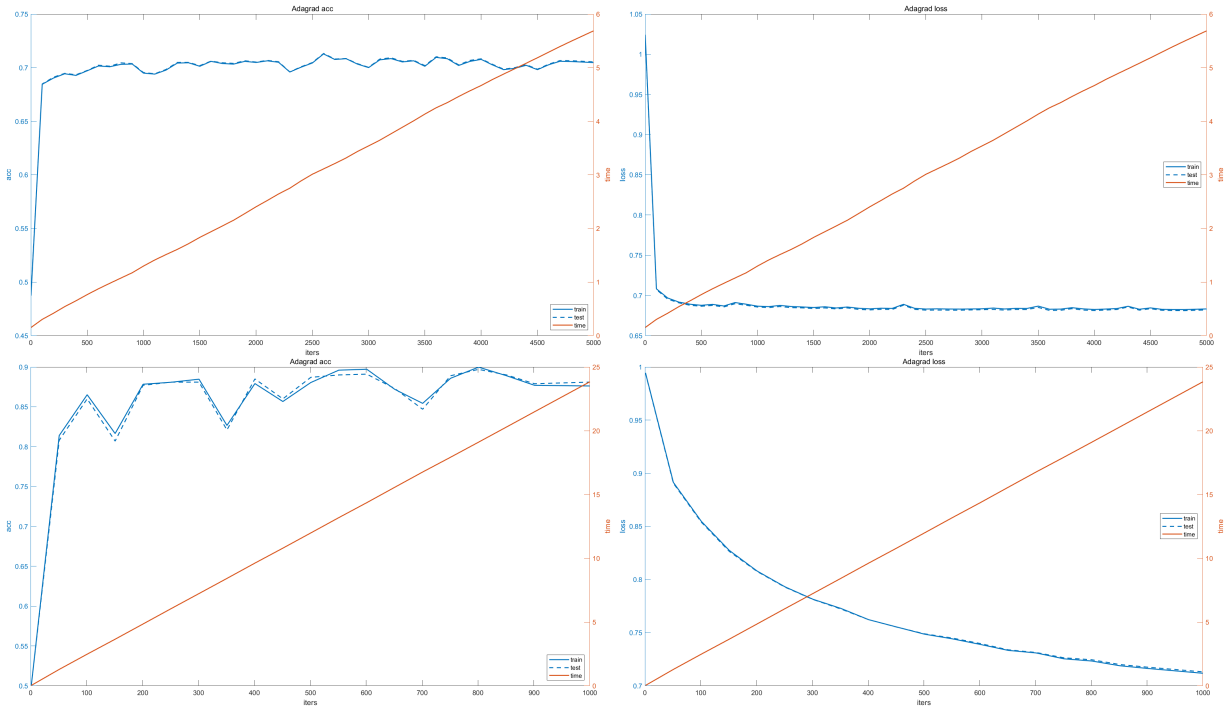

图 12: Adagrad 迭代准确率与误差曲线

选取的参数如下表 (由于 Gisette 数据集特征较多且实际稀疏，保正系数需要稍大，学习率则应减小以避免过度振荡，误差计算间隔指每隔多少次迭代计算准确率与误差，这部分也计入时间)：

表 6: Adagrad 迭代参数选择

	Covtype	Gisette
b	32	32
ε	0.0001	0.001
α	0.5	0.0001
迭代次数	5000	1000
误差计算间隔	100	50

可以发现，Adagrad 算法的函数值下降较稳定，能够稳定收敛，步长可以取稍大，单次迭代时间分别在 0.001 秒与 0.024 秒左右。

对 Adam 算法，取定合适参数迭代效果如图 13 ，图像含义同上，可验证正确性。

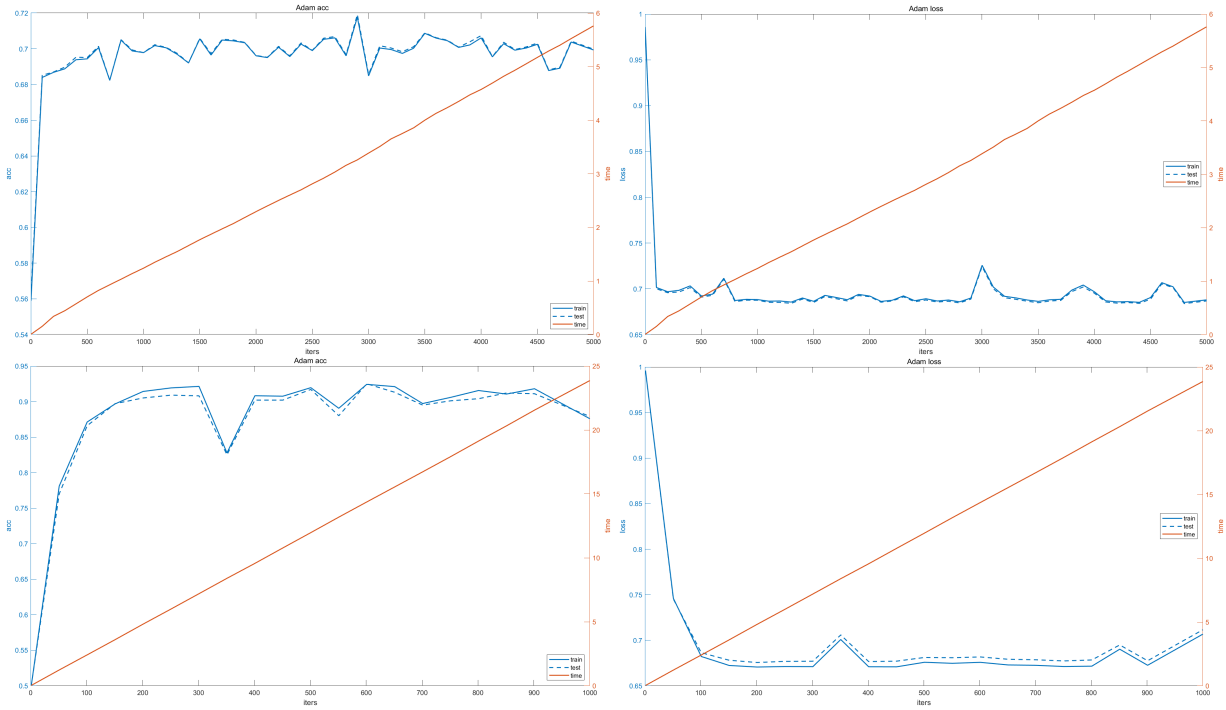

图 13: Adam 迭代准确率与误差曲线

选取的参数如下表 (ρ_1、ρ_2 采用了通常选择)：

表 7: Adam 迭代参数选择

	Covtype	Gisette
b	32	32
ε	0.0001	0.001
ρ_1	0.9	0.9
ρ_2	0.9	0.9
α	0.05	0.0001
迭代次数	5000	1000
误差计算间隔	100	50

可以发现，在与 Adagrad 迭代相同的学习率、保正系数与批次大小选取下，Adam 迭代拥有更快的收敛速度 (Gisette 上)，但有着更多的振荡，这与其更激进的迭代策略是相符的。事实上其学习率不能取得太高，如 Covtype 上将学习率取为了 Adagram 算法的十分之一，若更大则容易使结果不收敛。其单次迭代时间比 Adagrad 略微多一点，但总差别不到 10%，几乎可视为相同。由此，Adam 迭代在对稳定性要求不太高的场景比 Adagrad 应用更加广泛。

对 SVRG 算法，取定合适参数迭代效果如图 14 ，图像含义同上，可验证正确性。

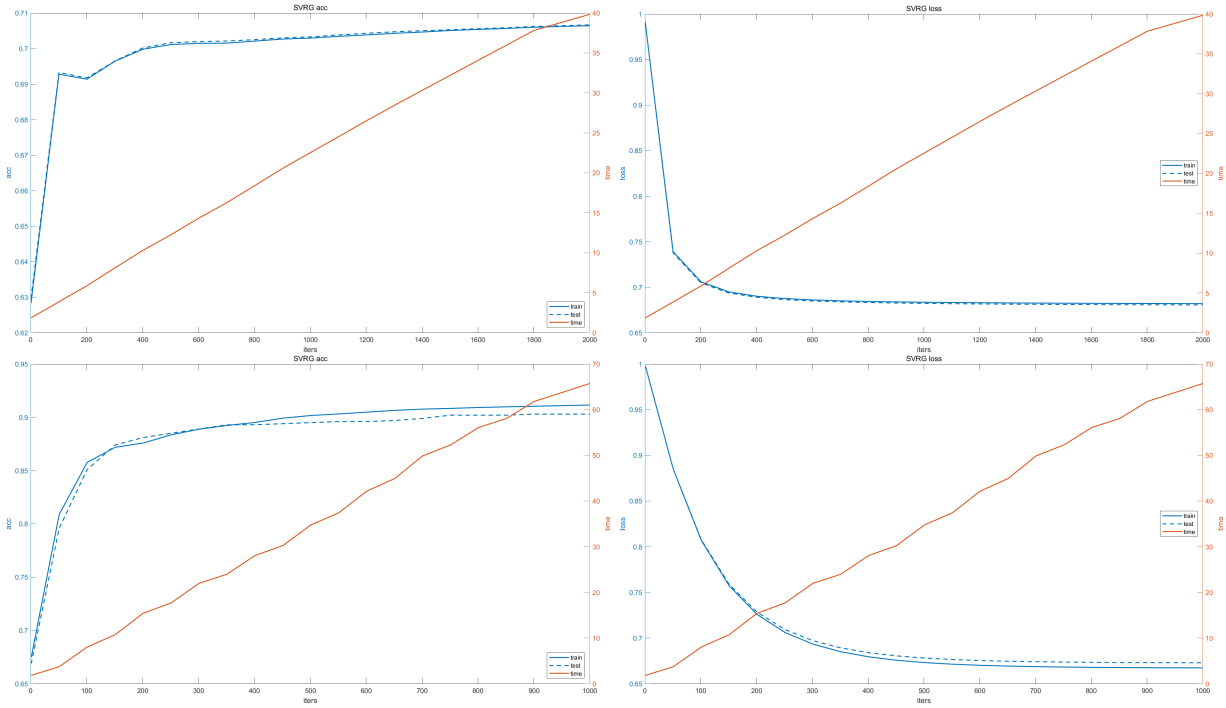

图 14: SVRG 迭代准确率与误差曲线

选取的参数如下表 (实际可选取更大的 m，此处为表现 SVRG 迭代的特性取得稍小)：

表 8: SVRG 迭代参数选择

	Covtype	Gisette
b	32	32
m	100	100
α	0.5	0.0001
迭代次数	2000	1000
误差计算间隔	100	50

可以发现，其表现出了非常出色的收敛稳定性，尤其是 Covtype 中它达到了所有算法中最高的最终准确率。不过，这样效果的代价是极高的平均单次迭代时间，分别为 0.020 秒、0.066 秒左右——这是由于其每隔一段时间就要进行开销巨大的全梯度估算。考虑到时间因素，此参数下其收敛速度是远比其他算法慢的。由此，只有当样本的分布非常差时，SVRG 算法才可能表现出很好的效果。

最后，对于 LBFGS 迭代，在尝试了非常多的参数后，可以发现结果较为随机：在 “运气较好” 时容易得到很好的结果，但时常会在某个时刻准确率骤降，误差骤升。分析原因可以发现，问题主要来自于，由于目标函数非凸，H^k 的估算不保证正定性，当估算的 $(s^k)^T y^k$ 接近 0 甚至为负时，其结果可能变得很奇异。由此，想要得到更好的 LBFGS 方法，还必须对此问题给出解决方案。一个可行的选择是仅在 $(s^k)^T y^k$ 较大时再进行更新，时间因素，未对此进行实现。

采用直接的 LBFGS 方法，取定合适参数迭代效果如图 15 ，图像含义同上，可验证正确性。

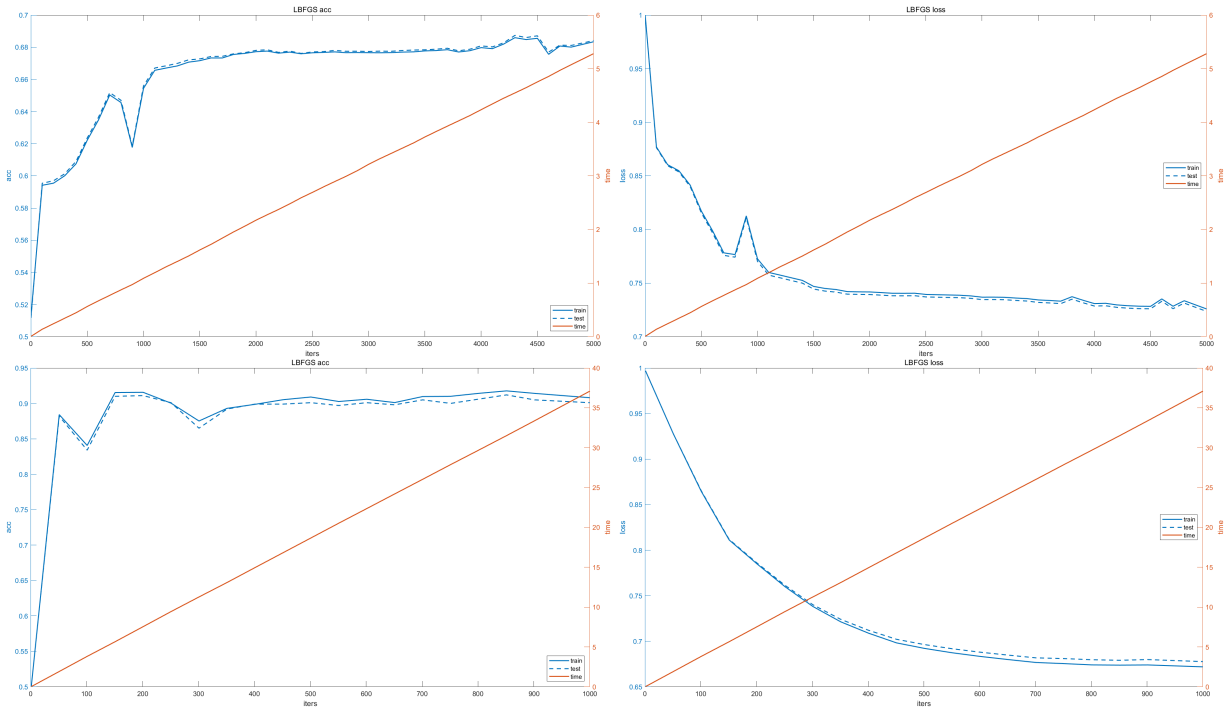

图 15: LBFGS 迭代准确率与误差曲线

选取的参数如下表 (m 采用了通常选择)：

表 9: LBFGS 迭代参数选择

	Covtype	Gisette
b	32	32
m	5	5
α	0.005	0.0001
迭代次数	5000	1000
误差计算间隔	100	50

改变参数可以发现，其学习率必须取得非常低才能保证算法大部分情况下的稳定性，而即使如此，其收敛性在不出现奇异情况时仍然很好。对特征较少的 Covtype，其单次迭代时间与 Adagrad 或 Adam 算法基本相同，收敛结果也不错；对特征很多的 Gisette，其单次迭代时间约为 0.037 秒，并未大很多，且到达 0.9 准确率的迭代次数或时间也较少。

总的来说，四种算法里 Adam 算法在单次迭代开销可能很大时效果是相对最好的，而 Adagrad 则相对保持了较好的稳定性与收敛速度。最后，我们用较稳定的 Adagrad 算法来分析正则化系数的作用。其他参数保持与上方相同，Gisette 的迭代次数改为 3000 次保证收敛，将结果列表如下：

表 10: 正则化系数对结果的影响

正则化系数	Covtype 准确率		Gisette 准确率	
	训练集	测试集	训练集	测试集
0.001	0.7088	0.7093	0.9202	0.9130
0.01	0.6847	0.6856	0.9202	0.9130
0.1	0.6295	0.6308	0.9202	0.9130
1	0.6140	0.6154	0.9195	0.9120
10	0.5655	0.5662	0.9078	0.9000

从中能够看出，我们最开始的分析是合理的，由于 Covtype 特征很少，添加更强的正则化并不能提升泛化能力，还增加了拟合误差，实际观察准确率曲线可以发现剧烈的振荡。不过，对 Gisette，加强正则化似乎并没有显著效果，对泛化能力的提升仍然有限，可能在训练集、测试集的划分更加不好时才能展现效果。

11 奇异值分解实验报告

11.1 算法实现

11.1.1 问题与算法

我们实现两种算法，线性时间 SVD 算法 (下称 LSVD 算法) 与随机 SVD 原型算法 (下称 RSVD)。下文中的简化奇异值分解是指，对行数 z_m 大于列数 z_n 的矩阵 Z，将其分解为 $Z = Z_U Z_D Z_V^T$，其中 Z_D 为 n 阶对角阵，且所有对角元从大到小排列，均非负；Z_V 为 n 阶正交阵，Z_U 为 $m \times n$ 矩阵且列相互正交。

LSVD 算法的过程如下：

- 给定矩阵 $A \in \mathbb{R}^{m\times n}$、近似奇异值个数 r、过采样个数 p。
- 记采样权重 $q_i = \frac{1}{\|A\|_F}\|\alpha_i\|$，这里 α_i 表示 A 的第 i 列。
- 考虑离散随机变量 X，以 q_i 的概率取到 i，进行 $r+p$ 次采样，得到 $i_1, \ldots, i_{r+p}$。
- 构造 $C \in \mathbb{R}^{m\times(r+p)}$，使得 C 的第 t 列为 A 的第 i_t 列乘 $((r+p)q_{i_t})^{-1/2}$。
- 对 C^TC 进行正交相似对角化 (MATLAB 自带的 Schur 分解)，得到 $C^TC = PDP^T$，P 正交、D 对角。
- 对 D 进行置换得到对角元从大到小排列的 D_0，同时对 P 的列进行置换得到 P_0，使得 $C^TC = P_0D_0P_0^T$。
- 设 P_0 的前 r 列构成矩阵 P_1，将 P_1 的第 i 列除以 D_0 的第 i 个对角元得到 P_2，记 $H = CP_2$。
- 取 $Y = A^TH$，设 Y 的简化奇异值分解为 $Y = VSW^T$。
- 设 $U = HW$，最终得到 A 的近似简化奇异值分解 $A \approx USV^T$ (这里的还原方式为讲义中第一种)。

RSVD 算法的过程如下：

- 给定矩阵 $A \in \mathbb{R}^{m\times n}$、近似奇异值个数 r、过采样个数 p 与强化系数 q (一般为 1 或 2)。
- 生成每个元素独立服从标准正态分布的矩阵 $\Omega \in \mathbb{R}^{n\times(r+p)}$。
- 取 $Y = (AA^T)^qA\Omega$，计算时从右往左以保证运算量较低。
- 设 Y 的 QR 分解为 $Y = QR$，设 Q^TA 的简化奇异值分解为 $Q^TA = U_0S_0V_0^T$。
- 取 U 为 Q 乘 U_0 的前 r 列，S 为 S_0 的前 r 行 r 列，V 为 V_0 的前 r 列，最终得到 A 的近似简化奇异值分解 $A \approx USV^T$。

11.1.2 代码文件结构

`a.jpg`	示例图片
`admm.m`	普通 ADMM 算法
`admm_acc.m`	用随机 SVD 加速的 ADMM 算法
`lsvd.m`	线性 SVD 近似算法
`rsvd.m`	随机 SVD 原型算法
`test_n.m`	核范数优化问题测试
`test_svd.m`	随机 SVD 算法效果测试

11.2 结果展示

对于随机生成的 2048×512 阶秩为 20 的矩阵，我们以 $\|A - USV^T\|_2$ 作为误差度量，取定 $p = r$，对不同 r 的近似结果如下：

表 11: 随机 SVD 算法低秩矩阵时间与效果对比表

算法	LSVD		RSVD ($q = 1$)		RSVD ($q = 2$)	
	误差	时间 (s)	误差	时间 (s)	误差	时间 (s)
$r = 5$	1.20×10^3	3.7×10^{-3}	1.14×10^3	7.1×10^{-3}	1.13×10^3	8.2×10^{-3}
$r = 10$	1.15×10^3	3.0×10^{-3}	1.01×10^3	7.4×10^{-3}	1.01×10^3	7.5×10^{-3}
$r = 15$	1.10×10^3	4.0×10^{-3}	9.15×10^2	9.9×10^{-3}	9.15×10^2	1.2×10^{-2}
$r = 20$	5.07×10^{-12}	4.2×10^{-3}	7.03×10^{-12}	1.2×10^{-2}	5.01×10^{-12}	1.4×10^{-2}

对于更大的 3072×4096 自选图片的例子，仍取定 $p = r$，对不同 r 的近似结果如下：

表 12: 随机 SVD 算法一般矩阵时间与效果对比表

算法	LSVD		RSVD ($q = 1$)		RSVD ($q = 2$)	
	误差	时间 (s)	误差	时间 (s)	误差	时间 (s)
$r = 5$	2.35×10^4	2.5×10^{-2}	1.86×10^4	2.5×10^{-1}	1.85×10^4	2.8×10^{-1}
$r = 10$	2.30×10^4	2.8×10^{-2}	1.23×10^4	4.1×10^{-1}	1.23×10^4	4.6×10^{-1}
$r = 15$	2.20×10^4	5.4×10^{-2}	9.17×10^3	5.8×10^{-1}	9.17×10^3	5.7×10^{-1}
$r = 20$	1.63×10^4	4.2×10^{-2}	7.04×10^3	6.9×10^{-1}	7.04×10^3	7.5×10^{-1}

综合两张表格可以看出，RSVD 算法较 LSVD 算法要消耗显著更多的时间，但也有着明显更好的近似效果 (除了第一张表格最后一行已经精确的情况，此时主要有数值精度决定，具有随机性)。此外，取 $q = 2$ 比起 $q = 1$ 改善的近似效果相对有限，不过时间消耗的增长也很有限。对自选图片，作出原图与按照不同随机奇异值分解方法分解后恢复的图片如图 16 。

图 16: 随机 SVD 算法图片压缩效果

从图中也能明显看到，RSVD 算法的误差更小体现为恢复的效果更好，人物的轮廓更加清晰。

11.3 应用

最后，我们用它来加速 ADMM 算法。对于优化问题

$$\min_{X\in\mathbb{R}^{m\times n}} \frac{1}{2}\sum_{(i,j)\in\Omega}(X_{ij}-M_{ij})^2+\mu\|X\|_*$$

在期中实验中已经给出了 ADMM 算法的表达式 (形式存在微小区别，已进行对应调整)，第 k 次迭代为

$$X^{k+1}=(\rho Z^k-Y^k+2M*\Omega)\div(\rho\mathbf{1}+2\Omega)$$

$$Z^{k+1}=S_{2\mu/\rho}(X^{k+1}+Y^k/\rho)$$

$$Y^{k+1}=Y^k+\tau\rho(X^{k+1}-Z^{k+1})$$

这里 $*$ 与 $\div$ 表示逐元素乘除，Ω 表示在 Ω 中的分量为 1，其他为 0 的矩阵，$S_\alpha(X)$ 表示将 X 奇异值分解后每个奇异值若绝对值小于 α 则缩小到 0，否则向 0 的方向收缩 α，然后还原得到的矩阵。

为了进行加速，我们将 $S_\alpha(X)$ 加入参数 r，将奇异值分解改为用 $q=1$、$p=r$ 的 RSVD 算法得到的近似奇异值分解。测试中，我们取定 M 为随机生成的 200×200 阶秩为 10 的矩阵，Ω 为随机 4800 个分量，且保持不变。

取定 $\rho=0.01$、$\tau=1$，最大迭代 10000 次，误差下降的容差 10^{-8}，近似秩 $r=10$，两方法效果对比如下：

表 13: ADMM 与加速 ADMM 结果对比表

参数		ADMM			加速 ADMM		
μ	$\mu\|M\|_*$	迭代次数	时间 (s)	结果	迭代次数	时间 (s)	结果
0.1	203.18	2516	24.8	202.93	10000	39.9	203.52
0.01	20.318	1817	18.3	20.315	950	3.92	20.316
0.001	2.0318	3077	30.0	2.0318	1549	6.32	2.0318

其收敛曲线如图 17 。

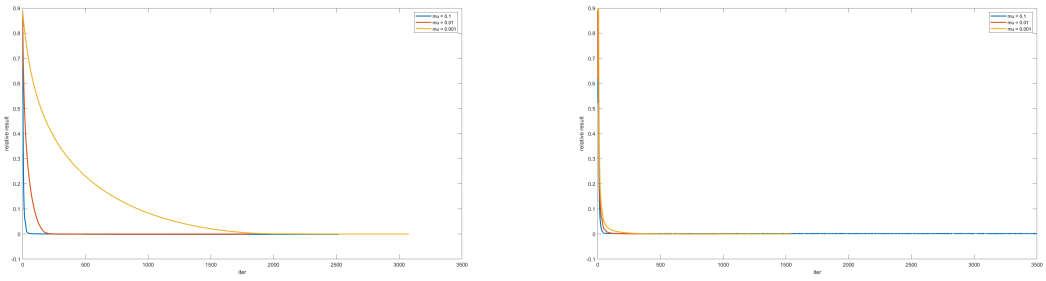

图 17: ADMM 与加速 ADMM 收敛曲线对比

与期中完全类似，$\mu\|M\|_*$ 为 $X=M$ 时优化函数取值，因此真解应小于它且在 μ 减小时逼近于它。可以发现，加速 ADMM 在 $\mu=0.1$ 时，由于真解与低秩距离相对远，出现了振荡，无法收敛到合理结果，也即其稳定性不如 ADMM 算法。不过，当真解的确接近低秩时，其时间消耗远低于 ADMM 算法，且同样能得到可接受的结果。因此，随机奇异值分解算法的确对于加速核范数正则化相关迭代有很好的效果。

12 相位恢复实验报告

12.1 算法实现

12.1.1 问题与算法

考虑一般的相位恢复问题，寻找 $z \in \mathbb{C}^n$ 使得

$$\forall r = 1, \dots, m, \quad y_r = |\langle a_r, z\rangle|^2$$

这里 $y_r \in \mathbb{R}$、$a_r \in \mathbb{C}^n$ 均给定。我们将所有 y_r 拼成的列向量记为 y，所有 a_r 作为行向量拼成的矩阵记为 A，则上述限制可以写为

$$y = |Az|^2$$

这里取模与平方都表示逐元素进行。为了能 (在相差相位意义下) 确定唯一的 z，我们至少保证 A 列满秩，从而 A^*A 正定。

考虑如下算法：

- 相位提升

  用上标星号表示共轭转置，直接计算有

  $$y_r = (a_r^* z)^*(a_r^* z) = z^* a_r a_r^* z = \mathrm{tr}(z^* a_r a_r^* z) = \mathrm{tr}(a_r a_r^* z z^*)$$

  由此，设 $Z = zz^*$，由线性代数知识，所有 Z 的集合即为秩不超过 1 的半正定阵，于是问题可化为找 n 阶半正定阵 Z 使得 $\mathrm{tr}(a_r a_r^* Z) = y_r$ 且 $\mathrm{rank}\, Z$ 最小。

  将 rank 松弛为核范数，由于半正定阵特征值即为奇异值，所有奇异值之和为特征值之和，从而即转化为找 n 阶半正定阵 Z 使得 $\mathrm{tr}(a_r a_r^* Z) = y_r$ 且 $\mathrm{tr}(Z)$ 最小。

  不过，此问题利用 SDP 的对偶问题进行 ADMM 构造需要计算一个 m 元二次函数的最小值 (且各分量并非独立)，计算开销很大，并不适合以此构造 ADMM 算法求解。

- 相位切割

  我们用 $b = \sqrt{y}$ 表示 y 每个分量开根号的结果，则有 $|Az| = b$。此问题可转化为，寻找 $x \in \mathbb{C}^m$ 使得 $|x| = b$，且 $\|Az - x\|^2$ 最小。进一步转化为

  $$\min_{z\in\mathbb{C}^n, u\in\mathbb{C}^m} \frac{1}{2}\|Az - b * u\|^2 \qquad \text{s.t.} \quad |u_i| = 1, \forall i$$

  这里星号代表逐分量相乘。

  由于这对 z 即为最小二乘问题，给定 u 时 z 的解可以写为 $z = (A^*A)^{-1}A^*(b * u)$。由此，代入计算可知问题最终化为

  $$\min_{u\in\mathbb{C}^m} u^* M u \qquad \text{s.t.} \quad |u_i| = 1, \forall i$$
  $$M = \mathrm{diag}(b)(I - A(A^*A)^{-1}A^*)\,\mathrm{diag}(b)$$

  这里 $\mathrm{diag}(b)$ 表示将 b 的元素作为对角元的对角阵。与之前类似，目标函数也可写为 $\mathrm{tr}(UM)$ 的形式，这里 $U = uu^*$ 为秩 1 半正定 Hemite 阵，约束即成为 U 的对角元均为 1。

  为进行 ADMM 算法构造，引入 $\phi = u$，且 $|\phi_i| = 1, \forall i$，目标函数仍为 $u^* M u$，则增广 Lagrange 函数为

  $$L_\rho(u, \phi, \lambda) = u^* M u + \mathcal{I}(\phi) + \langle \lambda, u - \phi\rangle + \frac{\rho}{2}\|u - \phi\|^2$$

  这里 $\mathcal{I}$ 在 ϕ 符合每个分量模长为 1 时为 0，否则为正无穷。ADMM 算法即

  $$u^{k+1} = \arg\min_u L_\rho(u, \phi^k, \lambda^k)$$

$$\phi^{k+1} = \arg\min_{\phi} L_\rho(u^{k+1}, \phi, \lambda^k)$$
$$\lambda^{k+1} = \lambda^k + \rho\tau(u^{k+1} - \phi^{k+1})$$

前两步优化中，后两项添加 $\|\lambda\|^2$ 适当倍数可以合并写为

$$\frac{\rho}{2}\|u - \phi + \rho^{-1}\lambda\|^2$$

从而直接计算可得到迭代

$$u^{k+1} = (2M + \rho I)^{-1}(\rho\phi^k - \lambda^k)$$
$$\phi^{k+1} = N(u^{k+1} + \rho^{-1}\lambda^k)$$

这里 N 代表将每个分量进行归一化模长，0 直接归一化为 1。

- 梯度下降

  考虑优化问题

  $$f(z) = \frac{1}{m}\sum_{r=1}^{m} |\langle a_r, z\rangle^2 - y_r|$$

  利用 Wirtinger 梯度直接计算可得其一个次梯度为

  $$\nabla f(z) = \frac{1}{m}\sum_{r=1}^{m} \operatorname{sign}(\langle a_r, z\rangle^2 - y_r) a_r a_r^* z$$

  利用矩阵运算，此形式事实上可写为

  $$\nabla f(z) = \frac{A^*}{m}(\operatorname{sign}(|Az|^2 - y) * Az)$$

  这里 sign 代表对向量每个元素进行，星号为逐元素相乘。

  最终的迭代方式为

  $$z_{k+1} = z_k - \mu_k \nabla f(z_k)$$

  这里 μ_k 为步长因子。

  由于此问题并非凸优化 (如 $z = 0$ 即为驻点)，为保证不落入局部最优，我们需要给出步长因子与初值的确定方式：

  – 初值选取

    根据讲义的理论推导，我们需要设定 z_0 是矩阵

    $$Y = \frac{1}{m}\sum_{r=1}^{m} y_r a_r a_r^*$$

    的最大特征值对应的特征向量。

    类似之前计算可知

    $$Yz = \frac{A^*}{m}(y * (Az))$$

    从而可以直接由幂法迭代得到 z_0 的方向。

    我们估算结果的模长为 $\sqrt{\|y\|_1/m}$，并以此作为 z_0 的模长。

  – 步长因子

    根据讲义的理论推导与实验要求中给出的网页上算法的实现，我们定义步长因子为

    $$\mu_k = \frac{\rho(k)}{\|z_0\|^2}$$

    其中

    $$\rho(k) = \min\left\{1 - \mathrm{e}^{-k/330}, 0.2\right\}$$

综合以上讨论，我们考虑相位切割与梯度下降两种算法构造。

12.1.2 代码文件结构

`ADMM.m` ADMM 算法实现
`gd.m` 梯度下降算法实现
`test.m` 测试程序

其中梯度下降算法的实现部分参考了实验要求中给出的网页代码。

12.2 结果展示

首先，随机生成 128 阶复向量 z，并随机生成 576 个复采样向量 a_r，用相位切割与梯度下降方法进行迭代的损失曲线如图 18，左侧为取定 $\rho = \tau = 1$ 的相位切割 ADMM 算法，右侧为梯度下降算法。由于真解可以相差相位，我们默认 z 与迭代结果的第一个分量都是正实数 (接下来的例子中不会出现第一个分量是 0 的情况) 以计算损失。

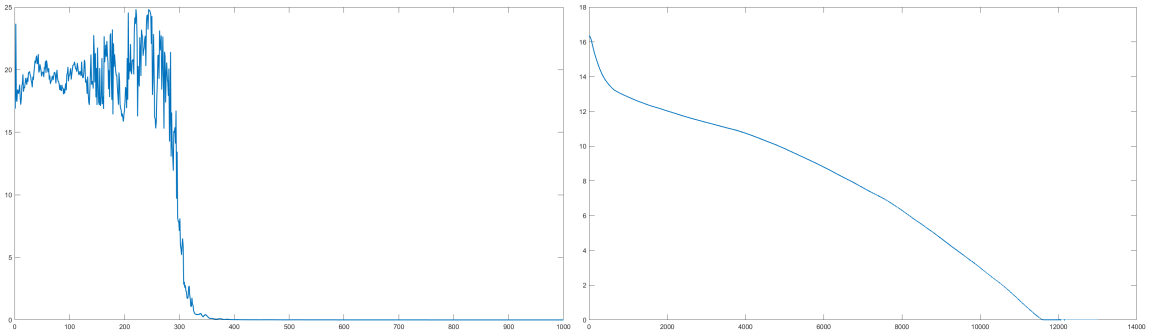

图 18: 随机生成采样损失曲线对比

可以看到，两种方法都能够收敛到最优结果，且相位切割进行 1000 次迭代的总时间为 1.79 秒，而梯度下降 14000 次迭代共 1.94 秒，由此，相位切割算法在此例子上是收敛更快的。

不过，在 A 并不容易显式表示时，相位切割算法就会暴露出缺点。考虑更为真实的采样例子：以一定要求生成掩码矩阵 M_a，并将向量 z 复制 L 份后与 M_a 逐元素相乘，并进行离散 Fourier 变换，得到一个新的 $n \times L$ 矩阵。

这个过程的确是进行了 $\mathbb{C}^n \to \mathbb{C}^{n\times L}$ 的线性变换，但变换矩阵 A 的构造相对复杂，想要像前一种情况一样显式计算 $2M_a + \rho I$ 的逆则更为困难。幸运的是，由于 Az 与 A^*z (事实上，可以直接构造出 A^*z/m) 仍然较容易计算，梯度下降算法仍然可以进行。我们取定 $L = 6$，仍然可以通过梯度下降算法得到如图 19 的下降结果。

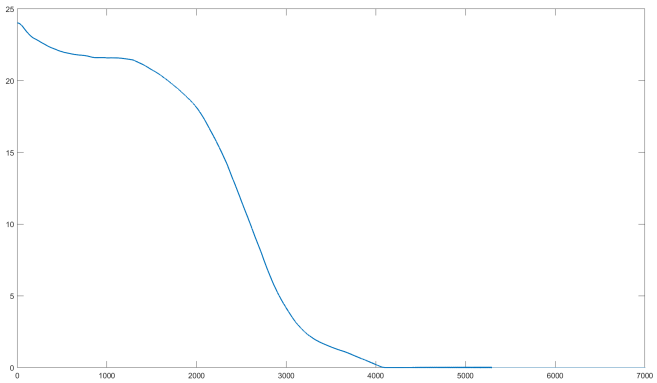

图 19: Fourier 采样损失曲线

考虑更复杂的例子：给定一张 $n_1 \times n_2$ 的灰度图，与上一种情况类似构造 L 种掩码进行放缩后进行离散 Fourier 变换，这时想要显式构造 A 还需要张量积技术，存储、运算的开销都过于大了，但梯度下降还是可以进行。考虑 $n_1 = 650$、$n_2 = 600$ 的特定图片，取定 $L = 21$。每进行 300 次迭代都导出结果，将其每个分量的灰度视为模长收缩到 $[0, 255]$ 区间并取整的值，作图后效果如图 20 。

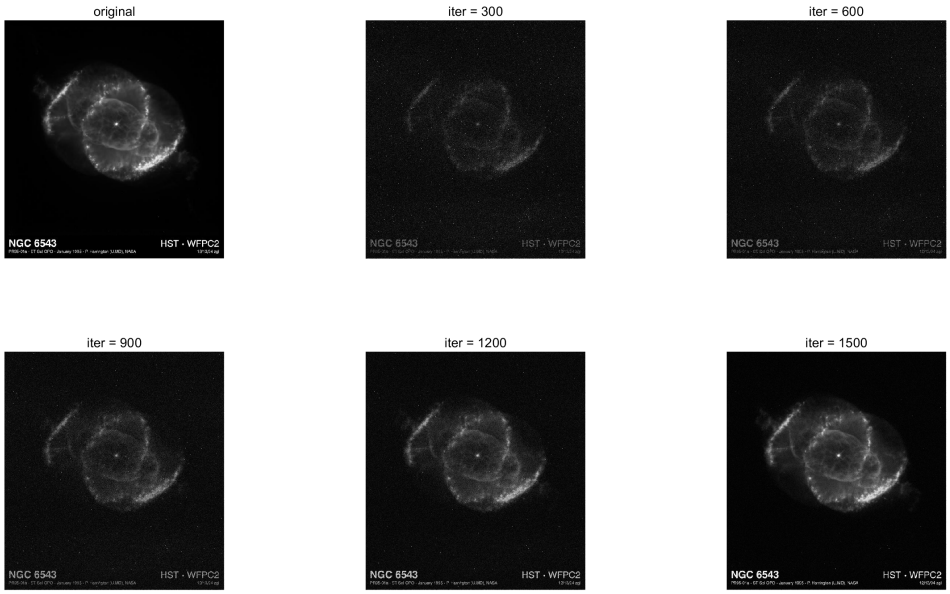

图 20: Fourier 采样图片恢复效果

由此可以看出梯度下降算法较好的收敛效果。

13 马尔可夫决策过程作业

1. (a) 记

$$f_V(s,a) = R(s,a) + \gamma \sum_{s' \in S} P(s' \mid s,a) V(s')$$

对任何 $s \in S$，若 $BV(s) \geq BV'(s)$，设 $BV(s) = f_V(s,a_0)$，根据定义有 $BV'(s) \geq f_{V'}(s,a_0)$，从而

$$|BV(s) - BV'(s)| = BV(s) - BV'(s) \leq f_V(s,a_0) - f_{V'}(s,a_0) = \gamma \sum_{s' \in S} P(s' \mid s,a_0)(V(s') - V'(s'))$$

根据条件概率定义有

$$\sum_{s' \in S} P(s' \mid s,a_0) = 1$$

且每项均为正，从而将所有 $V(s') - V'(s')$ 都放大到 $V(s) - V'(s)$ 最大处有 (利用 $\gamma > 0$)

$$|BV(s) - BV'(s)| \leq \gamma \|V - V'\|_\infty$$

同理，当 $BV'(s) > BV(s)$ 时，完全对称可得成立，由于此式对任何 s 成立，即得结论。

(b) 由 (a) 迭代可知

$$\|B^n V - B^n V'\|_\infty \leq \gamma^n \|V - V'\|_\infty$$

代入 $V = V_1$、$V' = V_0$ 即得

$$\|V_{n+1} - V_n\|_\infty \leq \gamma^n \|V_1 - V_0\|_\infty$$

再求和可知

$$\|V_{n+k} - V_n\|_\infty \leq \sum_{l=0}^{k-1} \|V_{n+l+1} - V_{n+l}\|_\infty \leq \sum_{l=0}^{k-1} \gamma^{n+l} \|V_1 - V_0\|_\infty$$

左侧系数为

$$\gamma^n \frac{1-\gamma^k}{1-\gamma}$$

由 $\gamma \in (0,1)$，其不超过，$\frac{\gamma^n}{1-\gamma}$，得证。

2. (a) 直接计算可知

$$D^\pi(\tau) = \prod_{t=1}^{H-1} \pi(s_t,a_t) P(s_{t+1} | s_t,a_t)$$

(b) 由定义可得

$$\rho(\pi) = \mathbb{E}_\tau[R(\tau) \mid \pi, s_1] = \sum_\tau R(\tau) D^\pi(\tau)$$

(c) 直接计算有

$$\nabla_\theta \rho(\pi_\theta) = \sum_\tau R(\tau) \nabla_\theta D^{\pi_\theta}(\tau) = \sum_\tau R(\tau) D^{\pi_\theta}(\tau) \nabla_\theta \log D^{\pi_\theta}(\tau)$$

将其重新写为期望形式即得到 (这里省略条件期望的条件 s_1)

$$\nabla_\theta \rho(\pi_\theta) = \mathbb{E}_\tau[R(\tau) \nabla_\theta \log D^{\pi_\theta}(\tau)]$$

再由 (a) 可知

$$\nabla_\theta \log D^{\pi_\theta}(\tau) = \nabla_\theta \sum_{t=1}^{H-1} (\log \pi_\theta(s_t,a_t) + \log(P(s_{t+1}|s_t,a_t))) = \sum_{t=1}^{H-1} \nabla_\theta \log \pi_\theta(s_t,a_t)$$

代入即得证。

(d) 我们先证明

$$\mathbb{E}\left[\frac{\partial}{\partial\theta_j}\log\pi_\theta(s_t,a_t)\mid s_t\right]=0$$

直接计算可知其为

$$\sum_a \pi_\theta(s_t,a)\frac{\partial}{\partial\theta_j}\log\pi_\theta(s_t,a)=\sum_a\frac{\partial}{\partial\theta_j}\pi_\theta(s_t,a)=\frac{\partial}{\partial\theta_j}1=0$$

由此，利用全期望定理，原式为 (外侧省略条件期望的条件 s_1,θ)

$$\sum_{t=1}^{H-1}\mathbb{E}_\tau\left[b_t(s_t)\frac{\partial}{\partial\theta_j}\log\pi_\theta(s_t,a_t)\right]=\sum_{t=1}^{H-1}\mathbb{E}_\tau\left[\mathbb{E}\left[b_t(s_t)\frac{\partial}{\partial\theta_j}\log\pi_\theta(s_t,a_t)\mid s_t\right]\right]$$

由条件独立性，其可以进一步拆分为

$$\sum_{t=1}^{H-1}\mathbb{E}_\tau\left[\mathbb{E}[b_t(s_t)\mid s_t]\,\mathbb{E}\left[\frac{\partial}{\partial\theta_j}\log\pi_\theta(s_t,a_t)\mid s_t\right]\right]$$

由于第二项恒为 0 即得求和为 0，得证。

(e) 可取 $b_t(s)=V^{\pi_\theta}_{H-t}(s)$，即 t 时刻开始执行策略 π_θ 所得到的价值。

14 期末实验报告

14.1 基本求解方法

14.1.1 问题描述

我们考虑的模型由如下方式描述：给定一些列车 $G_1,\ldots,G_N$，一些站点 $A_1,\ldots,A_n$，其中 A_1 为首发站，A_n 为终点站。已知每个列车的速度为 $v_1,\ldots,v_N$，速度为 v 的列车从站点 A_j 开到 A_{j+1} 的时间记为 $f(v)_j$。此外，取值为 0 或 1 的 b_{ij} 表示列车 i 是否在第 j 站停靠，其为 1 代表停靠，假定所有 $b_{i1}=b_{in}=1$，也即所有列车都在首发站与终点站停靠。额外给定总时间 T、最短间距 s、停靠最短时间 m、停靠最长时间 M，以上所有时间单位均为分钟，且为整数。

最后，还有三项信息也将提供，不过只用于作图：每个列车的名称、每个站点的名称、每个站点到始发站的距离。上述所有信息均来自实验要求中的**简单例子**。

我们要求在一个总时间为 T 的安排中给出每辆车的时刻表 (未必每辆车都能进入安排，若其无法进入则其时刻表为空)，以从起点站发车开始，到到达终点站为止。例如，可要求 G_1 在第 3 分钟从 A_1 发车，第 12 分钟到达 A_2，第 17 分钟从 A_2 发车..... 第 120 分钟到达 A_n。要求时刻表满足如下的约束：

1. 时刻表中的所有时间都是 0 到 T 范围中的**整数**。
2. **车头时距**类约束，此处相当于假设只有一条轨道：
   (a) 两辆不同列车从 A_1 的发车至少间隔 s 分钟。
   (b) 两辆不同列车到达 A_n 至少间隔 s 分钟。
   (c) 对 $j=2,\ldots,n-1$，不能有两辆不同列车同时停靠在第 j 站，且前一辆车从 A_j 出发与后一辆车到达 A_j 至少间隔 s 分钟。
   (d) 对 $j=1,\ldots,n-1$，若列车 G_{i_1} 从第 j 站出发比列车 G_{i_2} 早，则其到达第 $j+1$ 站也须比 G_{i_2} 早。

   出于模型简化的需要，我们将约束 (c)(d) 简化为约束 (c')：

   (c') 对 $j=2,\ldots,n-1$，两辆不同列车到达 A_j 至少间隔 s 分钟，两辆不同列车从 A_j 发车至少间隔 s 分钟。

   必须指出的是，对于一条轨道的模型，约束 (c') 并不具有实际意义，例如给出的参考解 Figure 2 中，在第 80 到 100 分钟的 C 站，两辆列车 “穿过” 了正停靠在此站的列车，除此以外，即使 $M>s$，约束 (c') 也并没有避免两辆列车间隔 s 分钟到达某站，都停靠 M 分钟后离开的情况，这意味着停靠时间可能出现重合。实验要求参考的原论文 [Caprara et al., 2002] 中，实际上也将上述 (c)(d) 给出的约束进行了弱化，**去除了约束 (c) 中不能同时停靠的要求，只保留不能交叉**，也即只避免了上面说的第一种情况，并未避免第二种情况。而实验要求则进一步简化，**忽略了不能交叉的约束**。
3. 对 $j=1,\ldots,n-1$，G_i 从第 j 站开到第 $j+1$ 站的时间**必须**为 $f(v_i)_j$。
4. 对 $j=2,\ldots,n-1$，若 $b_{ij}=1$，G_i 在第 j 站最少需要停靠 m 分钟，最长停靠 M 分钟。
5. 对 $j=2,\ldots,n-1$，若 $b_{ij}=0$，G_i **不能**在第 j 站停靠，也即到达时间与发车时间相同。

最后，我们需要考虑满足不同要求的可行时刻表，这里考虑三类要求：

1. 要求安排尽量多的列车进入。
2. 安排尽量多的列车进入，并要求所有列车的总到达时间最短。
3. 安排尽量多的列车进入，并要求所有列车的总停靠时间最短。
4. 安排尽量多的列车进入，并要求所有列车的总停靠时间最长。

14.1.2 直接求解法

本部分对应实验要求中第一题。

我们用图论方式建立模型，并对此进行直接求解。用上标 t 表示时间，考虑一个共有 $(T+1)(n-1)+2$ 个节点的**有重边**图。添加虚拟起点 σ、虚拟终点 τ，A_1 对应 $T+1$ 个出发节点 $\alpha_1^0,\ldots,\alpha_1^T$，$A_2$ 到 A_{n-1} 均对应 $T+1$ 个到达节点 $\beta_j^0,\ldots,\beta_j^T$ 与 $T+1$ 个出发节点 $\alpha_j^0,\ldots,\alpha_j^T$，$A_n$ 只有 $T+1$ 个到达节点。将顶点集记为 V。

对于每辆列车 G_i，规定属于它的**有向**边集 E_i (注意每个 E_i 不交，即使不同 E_i 有相同的连接，也**创建多条边**)：σ 连接所有 a_1^t 表示可能的出发时间；每个 α_j^t 连接到 $\beta_{j+1}^{t+f(v_i)_j}$ 表示从第 j 站到第 $j+1$ 站经历的时间；若 $b_{ij}=1$，每个 β_j^t 连接到 $\alpha_j^{t+m},\alpha_j^{t+m+1},\ldots,\alpha_j^{t+M}$ 表示可能的停靠到出发的间隔，否则 β_j^t 直接连接到 α_j^t；所有 β_n^t 连接 τ 表示到站。

我们将所有 E_i 的无交并记作整个图的边集 E，优化目标是在 E 中选出一些边作为时刻表。对每条边 $e\in E$，考虑 $x_e\in\{0,1\}$，0 代表选中，1 代表不选。由于每辆列车或无法安排，或有一条路径 σ 到 τ 的路径作为时刻表，对每辆车 A_i 的约束条件可以写为

$$\sum_{e\in\delta_i^+(\sigma)} x_e \le 1$$

$$\sum_{e\in\delta_i^-(v)} x_e = \sum_{e\in\delta_i^+(v)} x_e,\quad \forall v\in V\backslash\{\sigma,\tau\}$$

这里 $\delta_i^{\pm}(v)$ 代表 E_i 中离开/进入 v 的边集合。此外，虽然实验要求中给出了到终点的 x_e 总和不超过 1 这一约束，它在上述类似流函数的约束条件满足时是自然满足的。

冲突条件可以统一写为

$$z_{i,v} = \sum_{e\in\delta_i^-(v)} x_e$$

$$y_v = \sum_{i=1}^{N} z_{i,v}$$

$$\forall v\in V,\quad \sum_{v'\in\mathcal{N}(v)} y_{v'} \le 1$$

这里 $z_{i,v}$ 表示第 i 辆列车进入了 (初始节点以外的) 节点 v，y_v 表示节点 v 有车占用 (结合下一个约束，y_v 事实上也为 0 或 1)，$\mathcal{N}(v)$ 表示所有与 v 相互冲突的节点，具体来说，在我们的例子里，根据车头时距类约束，a_j^t 将与 a_j^{t-s+1} 到 a_j^{t+s-1} 冲突，b_j^t 将与 b_j^{t-s+1} 到 b_j^{t+s-1} 冲突。

此问题可以看作一个庞大的 01 整数规划问题，由此可以直接调用 Gurobi 求解。为了进行构造，需要给出每条边到序号的双射，以此刻画条件。我们假设边集的顺序是 $E_1,E_2,\ldots,E_N$，每个 E_i 中的边按照 σ 到 α_1、α_1 到 β_2、β_2 到 α_2，直到 β_n 到 τ 的顺序排列。每组中，以先 α 时间上标后 β 时间上标排列，例如 β_2 到 α_2 的边的排序为

$$\beta_2^0\to\alpha_2^m,\quad \beta_2^0\to\alpha_2^{m+1},\quad \ldots,\beta_2^0\to\alpha_2^M,\quad \beta_2^1\to\alpha_2^{m+1},\quad \ldots$$

由此，通过每一部分的边数构造偏移可将给定的边映射到序号，也同理可将序号映射回边。有映射后，即可以构造对应的约束条件。

接下来我们给出代码实现的更多细节，分为三个函数，其中问题构造与结果展示在接下来的算法实现中也将使用：

1. 问题构造

   这部分代码主要目的是搭建问题所需的数据结构。分为五步：构造顶点到序号的双射、统计边集数量、构造边集、构造冲突顶点集、构造权重。

   对于顶点到序号的双射，我们这里只实现一个函数

```
function bias = vertex_to_bias(vertex)
    switch vertex.type
        case "begin"
            bias = 1 + (vertex.station * 2 - 2) * (T + 1);
        case "arrival"
            bias = 1 + (vertex.station * 2 - 3) * (T + 1);
    end
end
```

输入任何一个给定的非 σ 或 τ 的顶点 (顶点的信息包含在何时到达何站/从何站出发)，输出一个偏移量，此顶点对应的序号即偏移量减其对应的时间再减 1。

接下来，我们将每辆列车对应的边分为 $2n-1$ 组，即之前所说的 σ 到 α_1、α_1 到 β_2、β_2 到 α_2，直到 β_n 到 τ。不同类型的边会具有不同的数量：σ 到 α_1 或 β_n 到 τ 的边数量为 $T+1$，α_i 到 β_{i+1} 的边数量会与速度相关，不超过 $T+1$ 条，而 β_i 到 α_i 的边若不停靠则为 $T+1$ 条，否则大约为 $(T+1)(M-m+1)$ 条，但实际上时间边界处会因无法取到而减少。

统计完边集的数量后，我们即可以对边进行构造了，我们的图的存储方式是一个 $|V|\times|E|$ 的关联矩阵 $\mathcal{E}$ (稀疏矩阵)，第 n_e 条边的起点为第 n_1 个顶点，终点为第 n_2 个顶点，则 $\mathcal{E}_{n_1n_e}=-1$、$\mathcal{E}_{n_2n_e}=1$。为了从边的逻辑含义定位到连接的顶点，即需要之前构造的 vertex to bias 函数。除此以外，我们还将记录第 i 辆列车对应的边集 E_j 对应 $\mathcal{E}$ 中的哪一段下标，用变量 bias train 进行记录。

在将上述简化的约束 (c') 转化为图论语言后，可发现冲突顶点集也即所有满足上下标在定位范围内的 $\alpha_j^t,\dots,\alpha_j^{t+s-1}$ 与 $\beta_j^t,\dots,\beta_j^{t+s-1}$，换句话说，任何一个站的连续 s 个出发节点与连续 s 个到达节点均至多只有一个被使用。由此，可得所有冲突顶点类约束的个数为 $N_C=(2n-2)(T+2-s)$，第一项表示层数，第二项表示每层的组数。每组冲突顶点都恰好 s 个，因此可用一个 $N_C\times s$ 的矩阵进行表示。

最后，我们需要根据问题类型在每条边对应的 x_e 上构造权重 p_e，最终优化目标为

$$\max\sum_{e\in E}p_ex_e$$

使得 x_e 满足之前所说的约束。根据问题的四种类型，权重分别构造为：

(a) 若要求列车尽量多，将每辆列车 σ 到 α_1 的边权重设置为 1，其余设置为 0 即可。

(b) 若要求列车尽量多的情况下总到达时间最短，将每辆列车 β_n^t 到 τ 的边权重设置为 $-t$，且每辆列车 σ 到 α_1 的边权重设置为某充分大数。由此，可以优先保证列车数量最多，再保证总到达时间最短。

(c) 若要求列车尽量多的情况下总停靠时间最短，将每辆列车 α_j^t 到 β_j^{t+l} 的边权重设置为 $-l$，且每辆列车 σ 到 α_1 的边权重设置为某充分大数。

(d) 若要求列车尽量多的情况下总停靠时间最长，将每辆列车 α_j^t 到 β_j^{t+l} 的边权重设置为 l，且每辆列车 σ 到 α_1 的边权重设置为某充分大数。

2. Gurobi 求解

为了能够进行 Gurobi 求解，我们要在上方的数据结构中额外将约束转化为矩阵。从定义中可以看出，只要确定了所有 $\delta_i^{\pm}(v)$ 中边的序号，即可利用约束条件构造矩阵。

将 $\mathcal{E}$ 中的所有 -1 变为 0，只保留 1，得到的矩阵记为 $\mathcal{M}$，可以发现，$\mathcal{M}$ 每行的非零元素位置即为所有以对应顶点为终点的边。同理，将 $\mathcal{E}$ 中的所有 1 变为 0，只保留 -1，得到的矩阵记为 $\mathcal{P}$，其每行的非零元素位置即为所有以对应顶点为终点的边。

除了基本的路径约束以外，将此性质结合之前得到的冲突顶点集，也可刻画车头时距类约束，最终得到线性规划对应的矩阵 A、向量 b 的表示 (这里有的行对应 $a_i^T x \leq b_i$，有的行对应 $a_i^T x = b_i$，需要在 Gurobi 中对每行的 sense 进行不同的设置)。

由于 Gurobi 内置了零一整数规划类型的求解 (变量类型设置为 B)，将权重的相反数作为优化目标 (由于默认为最小化，事实上也可将模型改为最大化) 输入，即可自动实现优化求解。

3. 结果展示

为了展示结果，我们需要先构造之前提到的顶点到序号的双射的另一方向：给定序号，提取出对应顶点的具体信息。这个映射的构造是直接的，通过取模、求余得到对应的站点与时间即可。

利用此映射我们可以将优化输出转化为时刻表结果：将优化输出中所有为 1 的项提取出来，逐个解释，可得 G_i 在第 t_1 分钟从 A_1 站出发，t_2 分钟到达 A_2 站..... 由此即得文字版时刻表，再作图得到图片。

值得一提的是，由于我们的边实际上是按照顶点顺序进行排列的，这又自然符合车站的时间顺序，只需要顺次提取输出的 x_e 中为 1 的项，即可得到 $G_1, G_2, \ldots, G_N$ 的时刻表。作图时，由于每个顶点实际上出现了两次，只要提取时刻表每条边的终点，再顺次连接，即得到了折线图。

下面，先考虑最基本的问题：安排尽量多的列车进入，对此，Gurobi 在经过了 231 秒的求解后给出了如图 21 的最优结果 (此图像与实验要求中的参考解图像不一致，这是由于参考解对应的数据事实上并不是实验要求中给出的数据)。从图像的合理性中即可验证算法的正确性，也可直接测试是否满足条件。此

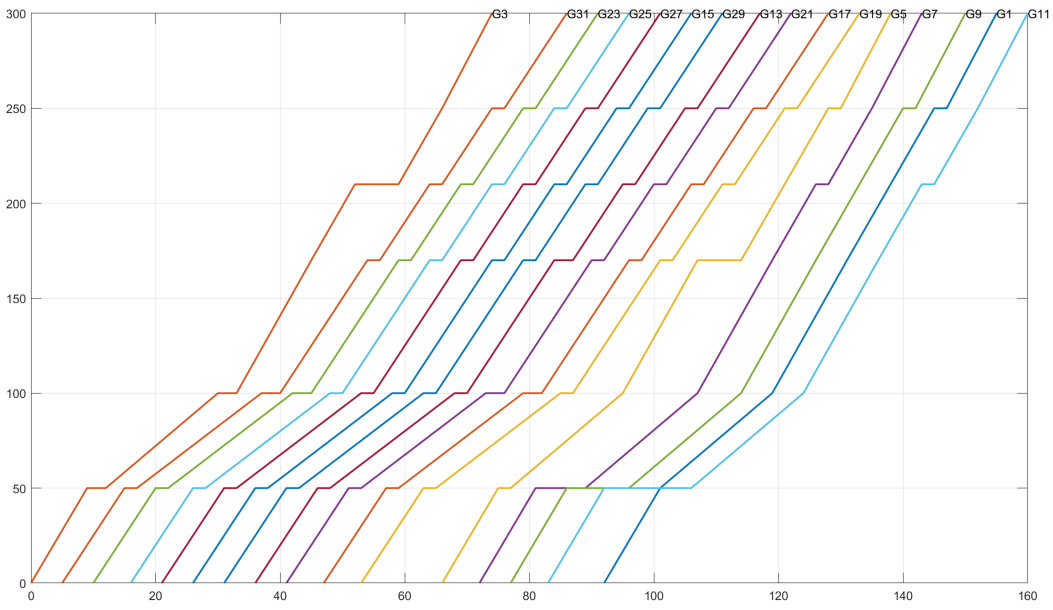

图 21: Gurobi 求解最大火车数

外，图像中安排了全部列车，也即我们的例子允许将所有列车一起放入，此后三个问题的最优解中应当都有全部列车。不过，如 80 分钟到 120 分钟附近的情况，我们的简化条件的确无法避免重叠或交叉的发生。

若考虑最早到达或最短停靠，给定对应的权重后，Gurobi 分别在 8 秒、11 秒后求解出了如图 22 、图 23 的结果。观察结果可以发现，事实上是可以所有停靠时间均为 2 分钟，无论是最早到达还是最短停靠都满足此最短停靠条件。此外，从求解时间关系也可以看出，对这类整数规划问题，由于其本质 NP 性，求解时间是非常不稳定的，如果分割平面较好，随着求解约束增加，结果可能反而变好。

另一个有趣的事实是，可以发现，指定了最早到达或最短停靠后，得到的图像的确几乎符合原始要求(c)(d)。这是由于，两类冲突情况的产生本质都是由于停靠时间过长，因此在停靠时间尽量短时很可能能得到没有冲突的结果，这就体现了近似的合理性。

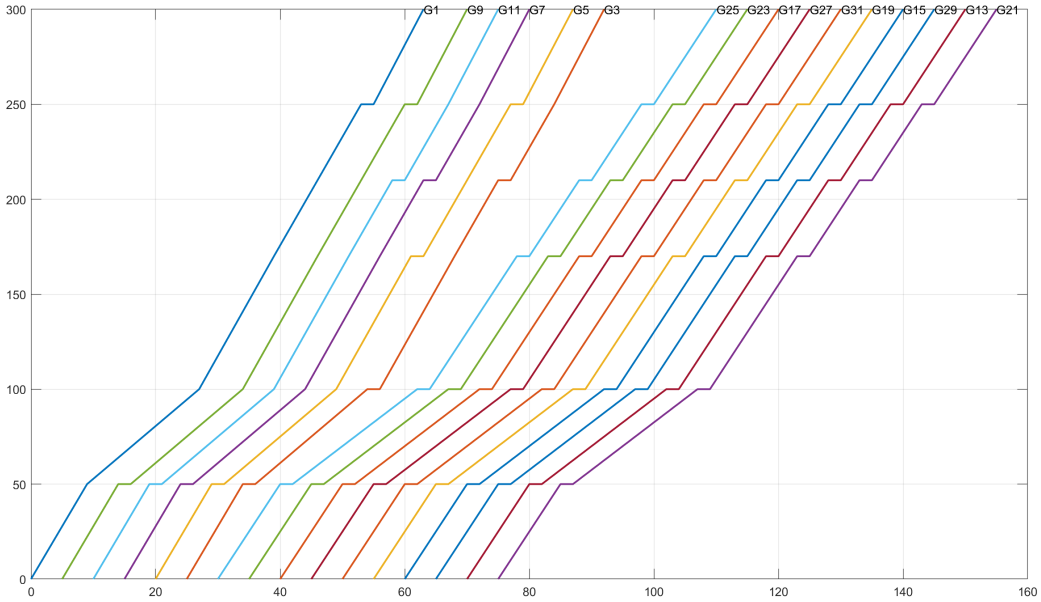

图 22: Gurobi 求解最早到达

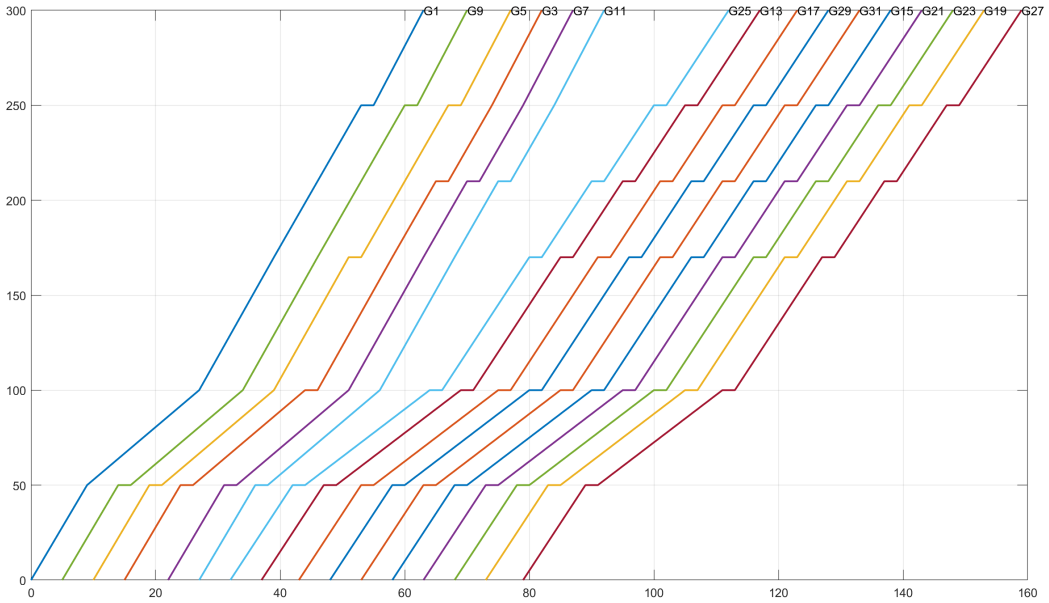

图 23: Gurobi 求解最短停靠

由于各个约束条件之间的相互包含性，我们之后针对 Lagrange 算法将主要采用**最早到达时间**与**最短停靠时间**作为约束进行测试，并观察能否得到所有停靠全为 2 分钟的结果。

反之，若我们需要停靠时间比较长，可以想象，结果将会无法成为一个合理的时刻表。Gurobi 对最长停靠问题的求解效果如图 24 ，此时，大量的重叠与交叉让这个时刻表无法成为合理的时刻表。事实上，

由于收敛所需时间较长，此结果仅是一个中间结果，可以想象，迭代收敛后将产生更多的交叉。

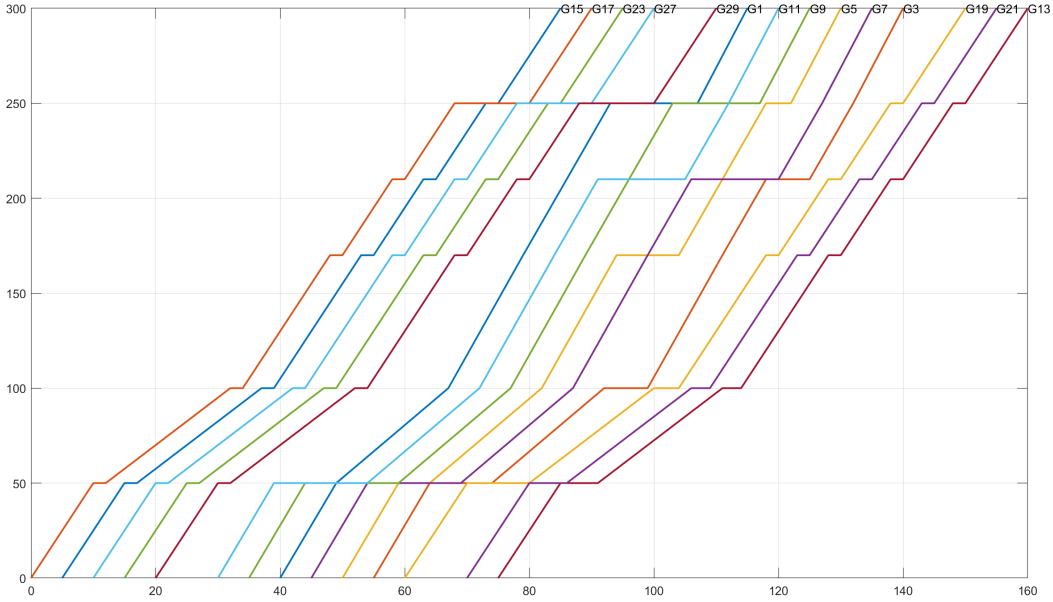

图 24: Gurobi 求解最长停靠

14.1.3 JSP 问题

本部分对应实验要求中第二题。

一个 JSP 问题 (**车间作业调度**问题) 是指如下形式的整数规划问题：给定 $p_{ik} \geq 0$、$a_{ihk} \in \{0,1\}$，找一组 x_{ijk}、c_{ik} 使得 c_{ik} 满足

$$\max_{i,k} c_{ik}$$

最小，且有约束

$$c_{ik} - p_{ik} + M(1 - a_{ihk}) \geq c_{ih}$$

$$c_{jk} - c_{ik} + M(1 - x_{ijk}) \geq p_{jk}$$

$$c_{ik} \geq 0, \qquad x_{ijk} \in \{0,1\}$$

这里 M 是充分大的正数，保证只要 $a_{ijk} = 0$ 或 $x_{ijk} = 0$ 即有约束成立。i、j 的范围是 1 到 n，k、h 的范围是 1 到 m，上述约束均对任何 h, i, j, k 成立。

具体来说，$p_{ik} \geq 0$ 是第 i 个工件在第 k 个机器上加工所需的时间，$a_{ihk} \in \{0,1\}$，其为 1 表示第 i 个工件在第 k 机器上加工前需要在第 j 个机器上加工 (由此所有 a_{ihk} 给出了所有工件的加工流程)。我们需要给出的 x_{ijk} 表示第 i 个工件是否先于第 j 个工件在第 k 个机器上加工，而 c_{ik} 即表示第 i 个工件在第 k 个机器上加工的完成时间。

为了将列车调度问题转化为 JSP 问题，我们需要先**假定列车停站时间固定**。沿用之前的记号，我们假设列车的停站时间恒定为 m，并构造工件与机器如下：每个列车看作一个工件 $G_1, \ldots, G_N$，将每站看作一个机器 $A_1, \ldots, A_n$，且任何两站之间都有 N 个机器 $M_1^1, \ldots, M_1^N$。每个工件 G_i 需要经历的机器为

$$A_1, \quad M_1^i, \quad A_2, \quad M_2^i, \quad \ldots, \quad A_{n-1}, \quad M_{n-1}^i, \quad A_n$$

且每个机器的时间为 (我们假设所有 $f(v_i)_j \geq m$，也即列车任何两站之间需要的时间比最短停站时间要长，对此问题是合理的假设)

$$s, \quad f(v_i)_1 - m, \quad s, \quad f(v_i)_2 - m, \quad \ldots, \quad s, \quad f(v_i)_{n-1} - m, \quad s$$

可以发现，这样的设计事实上代表了每个列车在两站之间互不干扰，而对每站都会产生一个时间为 s 的占用，即车头时距约束。由此，即可通过 JSP 问题的算法进行求解。

此建模与我们之前的建模有如下相同与不同：

1. JSP 建模的目标函数事实上是所有 x_e 的非线性函数，它考虑的是安排所有列车的最短时间，而之前建模只能处理类似最短时间求和的目标函数。
2. JSP 建模的模型更小，因为其无需将每个时间都设置为节点，由此其求解也比直接建模更快。不过，在进行真正求解时，若希望进行精确求解，实际复杂度还是将与最大时间相关 (类似背包问题复杂度与最大重量相关)。
3. 即使简化为 JSP 问题，其仍然是 NP 难的，需要通过整数规划算法进行相对复杂的求解。
4. JSP 建模的最大缺点在于，它需要保证每个工件在机器上的时间固定，难以直接处理动态完成时间。在所有停站时间都可取为最短时，此约束不会有太大的影响，但对更复杂的问题则无法解决。若想将原问题等价转化为 JSP 问题 (理论来说由 NPC 性质，这是可能的)，得到的 JSP 问题则将与时空网格求解类似，不再具有模型简单的性质。

14.2 Lagrange 方法

14.2.1 Lagrange 松弛

本部分对应实验要求中第四题。

为了应用 Lagrange 松弛求解之前实现的模型中，我们对每个 $\mathcal{N}(v)$ 引入**非负**乘子 λ_v。在第 k 次迭代中，考虑将不等式约束进行松弛，也即考虑优化问题

$$x^{(k)} = \arg\max \left\{ \sum_{e \in E} p_e x_e - \sum_{v \in V} \lambda_v^{(k-1)} \left(\sum_{v' \in \mathcal{N}(v)} y_{v'} - 1 \right) \right\}$$

对应约束为

$$\sum_{e \in \delta_i^+(\sigma)} x_e \leq 1$$

$$\sum_{e \in \delta_i^-(v)} x_e = \sum_{e \in \delta_i^+(v)} x_e, \quad \forall v \in V \backslash \{\sigma, \tau\}$$

且 y_v 定义同前。

乘子的更新遵循

$$\lambda_v^{(k)} = \max \left\{ 0, \lambda_v^{(k-1)} + \eta \left(\sum_{v' \in \mathcal{N}(v)} y_{v'}^{(k)} - 1 \right) \right\}$$

可以发现，原不等式约束可某种意义上看作 $\lambda_v \to +\infty$ 的情况，而此时的每步迭代中，目标函数为 x 的线性函数，于是对 $x^{(k)}$ 的求解成为了带负权的最短路径问题。由于这是一个有向无环图，且我们的顶点已经进行了排序，事实上只需逐个进行一次松弛即能得到 $x^{(k)}$，通过计算反向邻接表的预处理，这可以很快进行。

不过，上述方法存在两个需要改进的地方：

1. 首先，迭代后终止得到的 x 未必是一个可行解。为了让其可行，我们将进行贪婪递归删除的操作。具体来说，只要 x 不是可行解，计算所有产生冲突的约束，并计算哪一辆列车与其他列车冲突的次数最多，删除这辆列车对应的所有边。由于列车数量有限，总会在删除某辆列车后得到可行解。

2. 其次，直接进行递推会出现如图 25 的情况 (这里展示的是某一步迭代的结果)。可以发现，G15 到 G31 始终重合在一起，这是由于迭代过程对它们完全对称，并没有分离的过程。

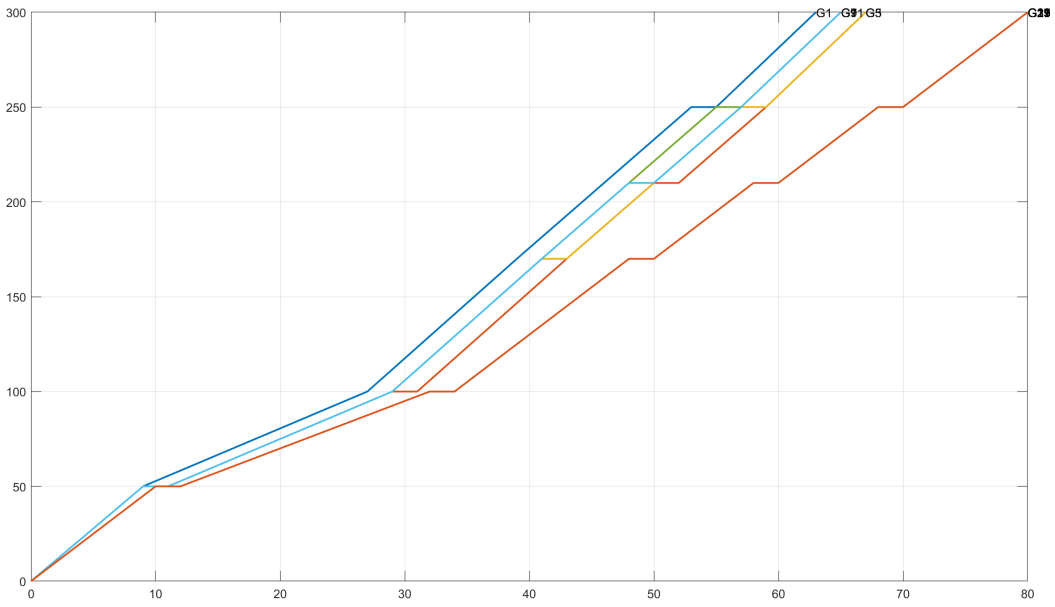

图 25: 未调整的 Lagrange 方法求解最早到达

由此，想要进行第 k 次迭代，实际上需要每对一辆列车的部分计算完 $x^{(k)}$ 后就更新一次乘子，这样才能保证后更新的部分能与先更新的部分错开。

解决这两个问题以后，即可以实现算法 (计算 x_e 在 y_v 中权重的方式与之前类似，可先构造冲突表再计算) 进行更新，例如，对最短考虑取 $\eta = 10^{-5}$，迭代 100 次，递归删除后得到的可行解如图 26 。

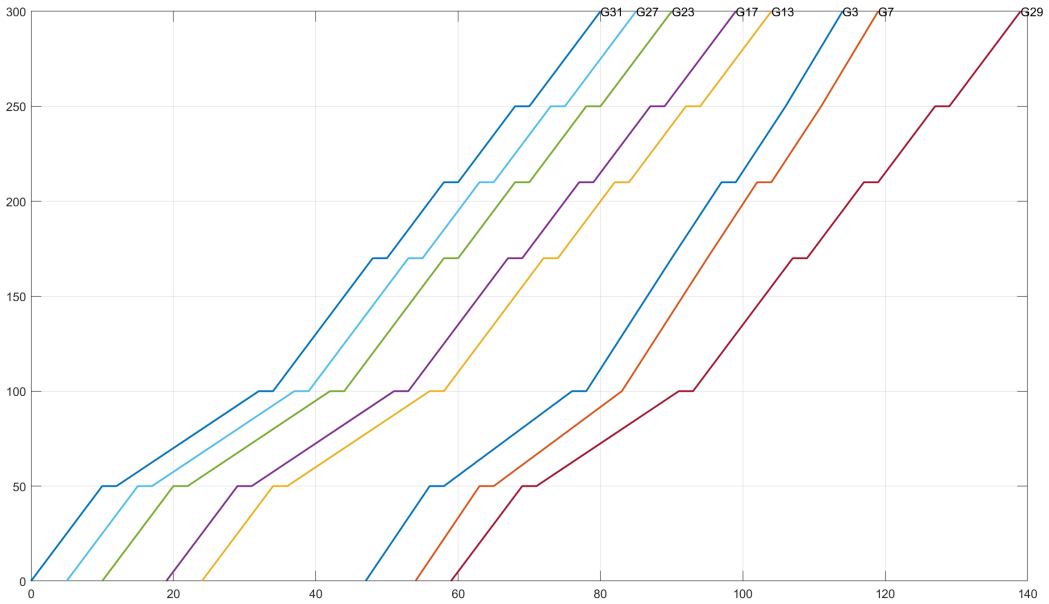

图 26: 递归删除后的 Lagrange 方法求解最短停靠

无论如何进行参数调整，结果都几乎为此。这个时刻表中只安排了 8 辆列车，效果并不好。为了研究其问题所在，我们先绘制删除前的原始时刻表如图 27 。

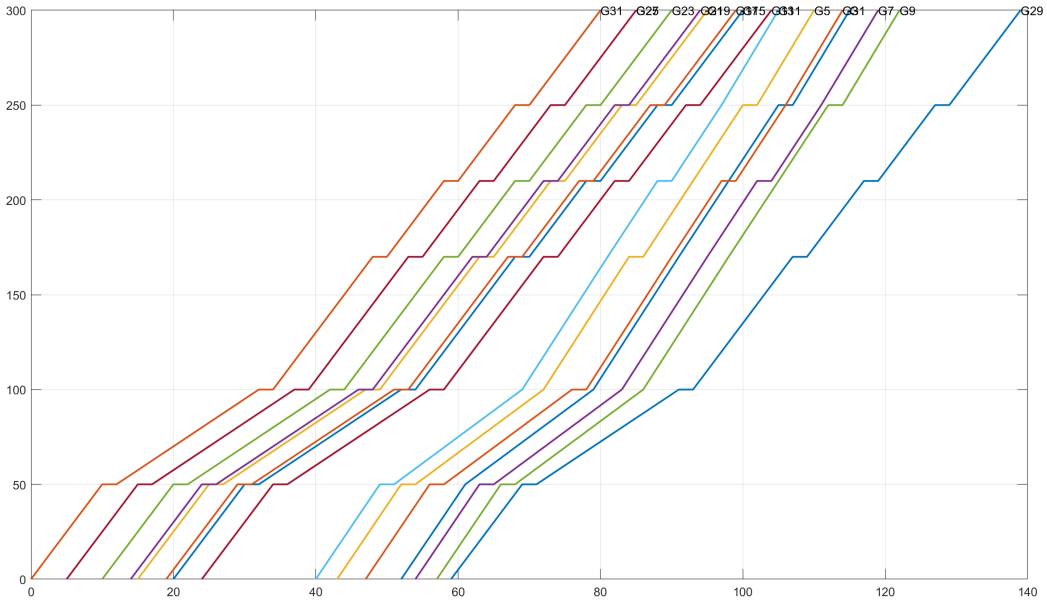

图 27: 递归删除前的 Lagrange 方法求解最短停靠

经过测试可以发现，步长无论取多短，几次迭代以后也将稳定到此情况。分析可发现，这是由于最短停靠时间对应的最优解 (不考虑冲突) 只与开始时间有关，而最短路径达到最优以后，剩余的优化完全由 λ 决定，因此 η 的值只决定 λ 与原本权重的比例，此时并不重要。另一方面，最短路径的变化相当于开始时间点进行移动，而图 27 事实上对应一个 “局部最优解”，任何一辆车的开始时间向左或右移动一分钟都会导致冲突更强。

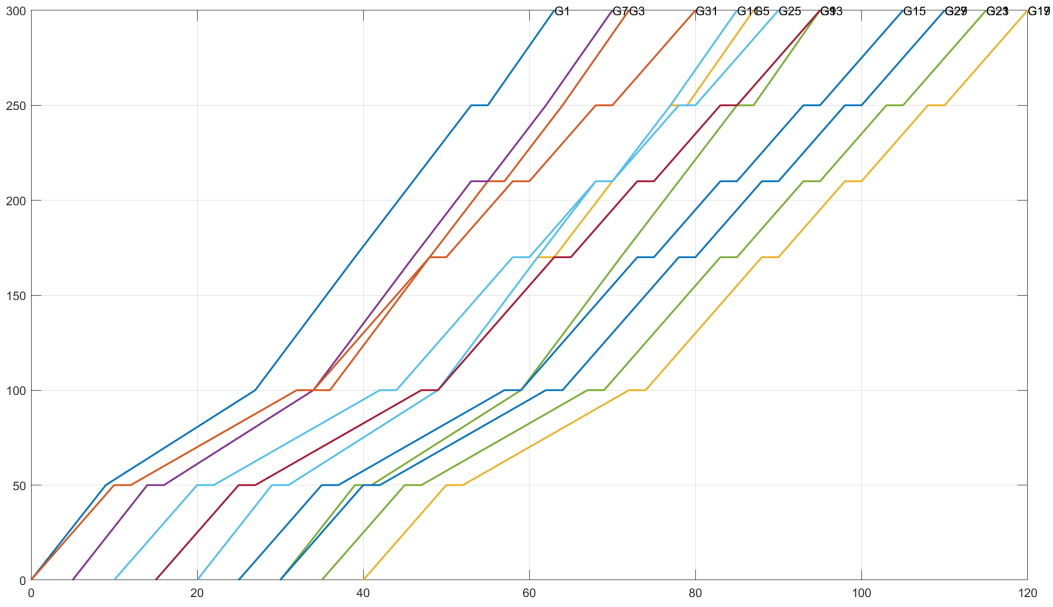

图 28: 递归删除前的 Lagrange 方法求解最早到达

不过，即使最短路问题对应的最优解唯一，如考虑最早到达问题，效果也并不好，一个删除前的迭代结果如图 28 。观察迭代过程可发现，即使将 η 调整到较小，增加迭代次数，结果也并没有很好的收敛性。

14.2.2 增广 Lagrange 函数

本部分对应实验要求中第五题。

为了解决上述 Lagrange 算法存在的问题，我们会希望能增添一个**近端项**，保证 x 的变化不会太大，由此，保持 λ_v 的更新过程与 x_e 的约束不变，我们将 Lagrange 方法的目标函数改写为增添近端项的**增广 Lagrange 函数**

$$x^{(k)} = \arg\max \left\{ \sum_{e\in E} p_e x_e - \sum_{v\in V} \lambda_v^{(k-1)} \left(\sum_{v'\in\mathcal{N}(v)} y_{v'} - 1 \right) - \frac{1}{2} \sum_{v\in V} \rho_v^{(k-1)} \left(\sum_{v'\in\mathcal{N}(v)} y_{v'} - 1 \right)^2 \right\}$$

可以发现，当 ρ 越大时，第三项的控制力将越强，由此 x_e 的更新应会更小，从而能起到控制步长的作用。但是，由于第三项不再是一次函数，此问题无法直接以最短路问题进行求解。

不过，由于我们已知最优解中每个 $\mathcal{N}(v)$ 里最多一条边对应的 x_e 为 1，我们强行**规定**所有冲突的 x_e 交叉项为 0，这样再利用 $x_e^2 = x_e$，上述问题就可以**线性化**为 (注意剩下的 x_e 相关的项只有 $x_e^2 - 2x_e = -x_e$)

$$x^{(k)} = \arg\max \left\{ \sum_{e\in E} p_e x_e - \sum_{v\in V} \lambda_v^{(k-1)} \left(\sum_{v'\in\mathcal{N}(v)} y_{v'} - 1 \right) - \frac{1}{2} \sum_{v\in V} \rho_v^{(k-1)} \left(- \sum_{v'\in\mathcal{N}(v)} y_{v'} \right) \right\}$$

由此重新构造系数进行最短路径求解可得到增广 Lagrange 函数法。

实际做法中，我们设定初始的 ρ 为 0，在每步迭代中对其进行更新：

$$\rho_v^{(k+1)} = \min \left\{ \rho_v^{(k)} + \mu \max \left\{ 0, \sum_{v'\in\mathcal{N}(v)} y_{v'}^{(k)} - 1 \right\}^2, \lambda_v^{(k+1)} \right\}$$

这里与 λ_v^{k+1} 取 min 是为了保证约束的合理性。为了方便计算，我们在每一轮迭代后再对 ρ 进行更新，而不是像 λ 一样每算一组 x_e 就进行更新。

取定 $\eta = 0.1$、$\mu = 0.001$，迭代 200、300、400 次与递归删除后的结果如图 29 。

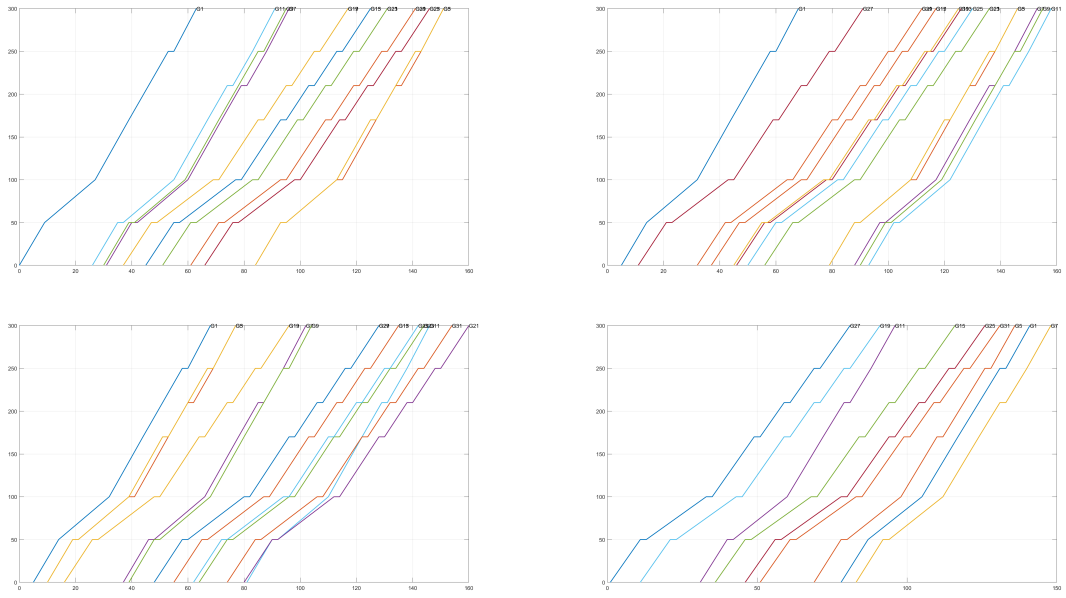

图 29: 增广 Lagrange 方法求解最短停靠

从图中可以看出此方法的确让收敛有了更好的性态。同样，相同参数下对最早到达迭代 500、1000、1500、2000 次后 (不进行递归删除) 的结果如图 30 。

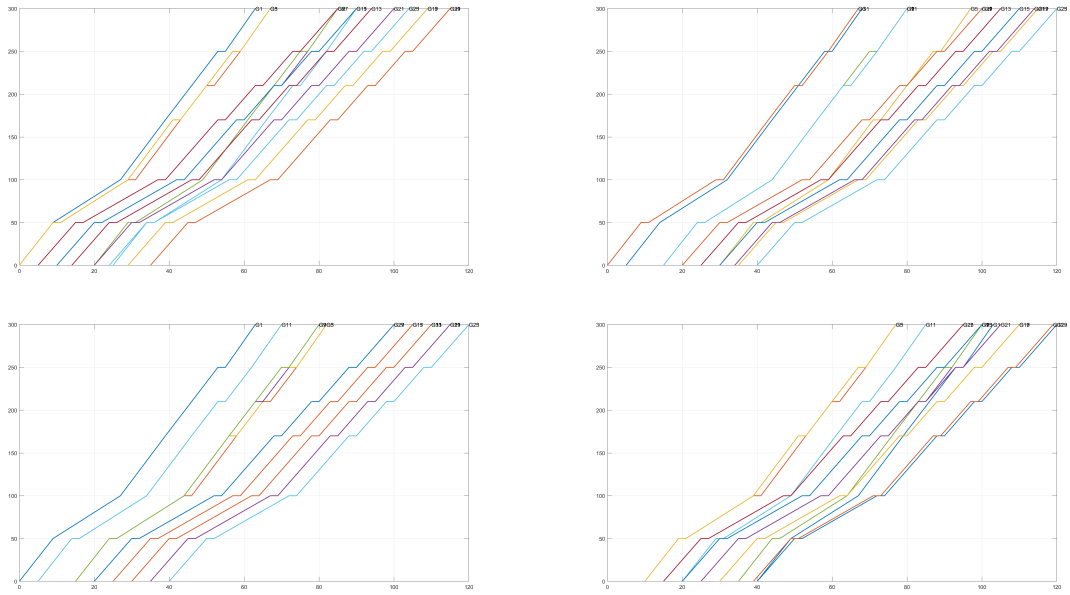

图 30: 增广 Lagrange 方法求解最早到达

由此，增广后的方法确实能比直接使用 Lagrange 函数更好收敛，但在这样的大规模情况中仍然具有局限性。

14.2.3 代码文件结构

`form_problem.m`	建立问题并存储进文件 `problem.mat`
`show_res.m`	将结果打印为时刻表或绘制为图表
`solve_augmented.m`	用增广 `Lagrange` 方法求解模型
`solve_gurobi.m`	用 `Gurobi` 直接求解模型
`solve_lagrange.m`	用 `Lagrange` 方法求解模型
`test.m`	测试算法

直接在文件夹下运行 test 脚本即可。

14.3 机器学习算法

14.3.1 大语言模型

本部分对应实验要求中第三题。

按照要求，我们尝试以大语言模型给出一个解决此优化问题的工作流。以北大官网 DeepSeek 为例，由于我们已经实验报告开始给出了问题的描述，直接复制作为基本提示词，并让其给出用图的数学建模：

我们考虑的模型由如下方式描述：

给定 N 辆列车 G_1 到 G_N，n 个站点 A_1 到 A_n，其中 A_1 为首发站，A_n 为终点站。已知每个列车的速度为 v_1 到 v_N，速度为 v 的列车从第 j 站开到第 $j+1$ 站的时间记为 $f(v)_j$。此外，取值为 0 或 1 的 b_{ij} 表示列车 i 是否在第 j 站停靠，其为 1 代表停靠，假定所有 $b_{i1} = b_{in} = 1$，也即所有列车都在首发站与终点站停靠。

额外给定总时间 T、最短间距 s、停靠最短时间 m、停靠最长时间 M，以上所有时间单位均为分钟，且为整数。

我们要求在一个总时间为 T 的安排中给出每辆车的时刻表 (未必每辆车都能进入安排，若其无法进入则其时刻表为空)。以从起点站发车开始，到到达终点站为止。例如，可要求 G_1 在第 3 分钟从 A_1 发车，第 12 分钟到达 A_2，第 17 分钟从 A_2 发车..... 第 120 分钟到达 A_n。

要求时刻表满足如下的约束：时刻表中的所有时间都是 0 到 T 范围中的整数。车头时距类约束，即两辆不同列车从 A_1 的发车至少间隔 s 分钟；两辆不同列车到达 An 至少间隔 s 分钟；对 j 从 2 到 $n-1$，两辆不同列车到达 A_j 至少间隔 s 分钟，两辆不同列车从 A_j 发车至少间隔 s 分钟。对 j 从 1 到 $n-1$，G_i 从第 j 站开到第 $j+1$ 站的时间必须为 $f(v_i)_j$。对 j 从 2 到 $n-1$，若 $b_{ij}=1$，G_i 在第 j 站最少停靠 m 分钟，最长停靠 M 分钟，否则 G_i 不能在第 j 站停靠，也即到达时间与发车时间相同。

假设我们希望给出的时间表能容纳所有列车，且所有列车到达终点站的时间之和最小。给出以上问题的一个数学建模，用图上的优化问题表示。

DeepSeek 生成的建模为 (调整了部分排版，未改变内容)：该问题可建模为一个基于图的整数线性规划问题，具体如下：

决策变量：

- $a_{i,j} \in \mathbb{Z}^+$：列车 G_i 到达站点 A_j 的时间
- $d_{i,j} \in \mathbb{Z}^+$：列车 G_i 从站点 A_j 发车的时间

目标函数：最小化所有列车到达终点站的时间之和：

$$\min \sum_{i=1}^{N} a_{i,n}$$

约束条件：

1. 行驶时间约束 (轨道物理限制)：

$$a_{i,j+1} = d_{i,j} + f(v_i)_j \quad \forall i \in [1,N],\, j \in [1,n-1]$$

2. 中间站停靠时间约束：若 $b_{ij}=1$ (停靠)，则

$$m \le d_{i,j} - a_{i,j} \le M \quad \forall i \in [1,N],\, j \in [2,n-1]$$

若 $b_{ij}=0$ (通过)，则

$$d_{i,j} = a_{i,j} \quad \forall i \in [1,N],\, j \in [2,n-1]$$

3. 车头时距约束 (安全间隔)：

- 发车站 A_1：

$$|d_{i,1} - d_{k,1}| \ge s \quad \forall i \ne k$$

- 终点站 A_n：

$$|a_{i,n} - a_{k,n}| \ge s \quad \forall i \ne k$$

- 中间站 A_j ($j \in [2,n-1]$)：

$$|a_{i,j} - a_{k,j}| \ge s \quad \text{且} \quad |d_{i,j} - d_{k,j}| \ge s \quad \forall i \ne k$$

4. 时间范围约束：

$$0 \le a_{i,j},\, d_{i,j} \le T \quad \forall i,j$$

$$a_{i,n} \le T \quad \forall i \quad (\text{确保所有列车在总时间内完成})$$

5. 逻辑顺序约束：

   - 发车时间不早于到达时间：

$$d_{i,j} \geq a_{i,j} \quad \forall i, j$$

   - 起点站和终点站必须停靠 (隐含通过 $b_{i1} = 1, b_{in} = 1$ 实现，但停靠时间不受限)。

建模说明：

- 该模型通过整数变量表示离散时间点，约束确保列车运行符合物理规律与调度规则。
- 车头时距约束通过离散事件间的最小间隔保证安全性。
- 目标函数旨在最大化运输效率，缩短总行程时间。
- 最终解必须在满足所有约束的条件下，覆盖所有列车的可行调度方案。

可以发现，虽然其给出的描述并没有用图简化问题，但确实是一个合理的数学描述，尝试通过提示词让其写代码解决：

请写出 Python 或 MATLAB 代码以解决此问题，输入为：trains 表示列车数量，stations 表示站点数量，velocities 表示列车速度，time_by_velocity 表示上述速度映射到向量的函数 f(v)，if_stop 表示上述 b_{ij}，以及整数 T、m、M、s 含义同前。

DeepSeek 最终给出了通过现有的优化器直接求解上述整数规划问题的 Python 代码，不过，它也指出这样直接的建模只能解决小规模的问题，当规模放大时复杂度将指数提升：

```
from pulp import *

def schedule_trains(trains, stations, velocities, time_by_velocity,
    if_stop, T, m, M, s):
    # Precompute travel times for each train and section
    travel_time = [
        [time_by_velocity(v, j) for j in range(stations-1)]
        for v in velocities
    ]

    # Create the problem instance
    prob = LpProblem("Train_Scheduling", LpMinimize)

    # Create decision variables: a[i][j] and d[i][j]
    a = LpVariable.dicts("arrival",
                         [(i, j) for i in range(trains) for j in range(stations)],
                         lowBound=0, upBound=T, cat='Integer')
    d = LpVariable.dicts("departure",
                         [(i, j) for i in range(trains) for j in range(stations)],
                         lowBound=0, upBound=T, cat='Integer')

    # Objective function: Minimize total arrival time at final station
    prob += lpSum(a[(i, stations-1)] for i in range(trains))
```

```
# Constraint 1: Travel time constraints
for i in range(trains):
    for j in range(stations-1):
        prob += a[(i, j+1)] == d[(i, j)] + travel_time[i][j]

# Constraint 2: Dwell time constraints
for i in range(trains):
    for j in range(stations):
        # Departure must occur after arrival
        prob += d[(i, j)] >= a[(i, j)]

        if j in [0, stations-1]:
            # First and last station must have b_ij=1 (handled by dwell time)
            continue
        else:
            if if_stop[i][j]:
                prob += d[(i, j)] - a[(i, j)] >= m
                prob += d[(i, j)] - a[(i, j)] <= M
            else:
                prob += d[(i, j)] == a[(i, j)]

# Constraint 3: Headway constraints with big-M formulation
M_big = T + 1  # Large enough constant

for i in range(trains):
    for k in range(i+1, trains):
        for j in range(stations):
            # Arrival headway for intermediate and final stations
            if j != 0:  # First station only has departure headway
                y_a = LpVariable(f"y_a_{i}_{k}_{j}", cat='Binary')
                prob += a[(i, j)] - a[(k, j)] + M_big*(1 - y_a) >= s
                prob += a[(k, j)] - a[(i, j)] + M_big*y_a >= s

            # Departure headway for intermediate and first stations
            if j != stations-1:  # Last station only has arrival headway
                y_d = LpVariable(f"y_d_{i}_{k}_{j}", cat='Binary')
                prob += d[(i, j)] - d[(k, j)] + M_big*(1 - y_d) >= s
                prob += d[(k, j)] - d[(i, j)] + M_big*y_d >= s

# Solve the problem
prob.solve(PULP_CBC_CMD(msg=False))  # Disable solver messages

# Extract results
```

```
    schedules = []
    for i in range(trains):
        valid = True
        schedule = []
        for j in range(stations):
            a_val = value(a[(i, j)])
            d_val = value(d[(i, j)])

            # Check time validity
            if not (0 <= a_val <= T and 0 <= d_val <= T and a_val <= d_val):
                valid = False
                break

            schedule.append((int(d_val), int(a_val)))  # (departure, arrival)

        if valid and value(a[(i, stations-1)]) <= T:
            # Format: List of (departure, arrival) times per station
            schedules.append(schedule)
        else:
            schedules.append([])  # Invalid schedule

    return schedules

# 示例使用说明
if __name__ == "__main__":
    # 示例参数定义
    trains = 2
    stations = 3
    velocities = [1, 2]

    # 假设的time_by_velocity函数：行驶时间为固定值示例
    def time_by_velocity(v, section):
        # 假设每个区间的行驶时间为速度的倒数（示例）
        return 10 // (v + 1)

    # if_stop矩阵：必须满足始发站和终点站为1
    if_stop = [
        [1, 0, 1],  # 列车0：在中间站不停靠
        [1, 1, 1]   # 列车1：在中间站停靠
    ]

    T = 50  # 总时间限制
    m = 2   # 最小停靠时间
    M = 5   # 最大停靠时间
```

```
s = 3   # 安全间隔

# 运行求解
schedules = schedule_trains(trains, stations, velocities,
    time_by_velocity, if_stop, T, m, M, s)

# 输出结果
for i, schedule in enumerate(schedules):
    if schedule:
        print(f"Train {i} schedule:")
        for j, (depart, arrive) in enumerate(schedule):
            print(f"  Station {j+1}: Depart at {depart}, Arrive at {arrive}")
    else:
        print(f"Train {i}: No valid schedule")
```

检测可以发现此代码的确可以执行，的确有基本正确的结果。

14.3.2 问题生成

本部分对应实验要求中第六题。

用大语言模型给出了求解方法后，还需要构建实例进行验证。我们在之前提示词的基础上，要求其生成从简单到复杂的五组输入，并分别以自然语言、数学形式、代码实现三种方式给出，得到的结果如下：

以下是为该问题设计的五组从简单到复杂的输入样例，分别以自然语言、数学形式和代码实现三种方式呈现：

样例 1 基础案例 (单个列车，无冲突)

- 自然语言描述

  1 辆列车，2 个站点

  速度 $v = 1$，行驶时间函数 $f(v)_j = 5$ 分钟

  必须停靠首末站（中间无停靠）

  $T = 10$ 分钟，$s = 1$ 分钟，$m = 1$ 分钟，$M = 5$ 分钟

- 数学形式

  $N = 1, n = 2$

  $v = [1], f(v)_0 = 5$

  $b = [[1, 1]]$

  $T = 10, s = 1, m = 1, M = 5$

- 代码实现

  ```
  trains = 1
  stations = 2
  velocities = [1]
  def time_by_velocity(v, j): return 5
  if_stop = [[1, 1]]
  T = 10
  m, M, s = 1, 5, 1
  ```

样例 2 冲突约束测试 (两车追尾场景)

- 自然语言描述

  2 辆列车，3 个站点

  速度均为 $v = 2$，行驶时间 $f(v)_j = 3$ 分钟

  列车 1 在中间站停靠，列车 2 不停

  $T = 20$ 分钟，$s = 3$ 分钟，$m = 2$ 分钟，$M = 4$ 分钟

- 数学形式

  $N = 2, n = 3$

  $v = [2, 2]$

  $f(v)_j = 3$ (所有区间)

  $b = [[1, 1, 1], [1, 0, 1]]$

  $T = 20, s = 3, m = 2, M = 4$

- 代码实现

```
trains = 2
stations = 3
velocities = [2, 2]
def time_by_velocity(v, j): return 3
if_stop = [
    [1, 1, 1],  # 列车1在中间站停靠
    [1, 0, 1]   # 列车2中间站不停
]
T = 20
m, M, s = 2, 4, 3
```

样例 3 动态速度优化 (不同速度混合)

- 自然语言描述

  3 辆列车，4 个站点

  速度分别为 $1, 2, 3$，行驶时间 $f(v, j) = 2v/(j + 1)$

  停靠矩阵随机生成 (需满足首末站为 1)

  $T = 50$ 分钟，$s = 2$ 分钟，$m = 1$ 分钟，$M = 5$ 分钟

- 数学形式

  $N = 3, n = 4$

  $v = [1, 2, 3]$

  $f(v, j) = \text{floor}(2v/(j + 1))$

  $b = [[1, 0, 1, 1], [1, 1, 0, 1], [1, 1, 1, 1]]$

  $T = 50, s = 2, m = 1, M = 5$

- 代码实现

```
trains = 3
stations = 4
```

```
velocities = [1, 2, 3]
def time_by_velocity(v, j): return (2*v) // (j+1)
if_stop = [
    [1,0,1,1],
    [1,1,0,1],
    [1,1,1,1]
]
T = 50
m, M, s = 1, 5, 2
```

样例 4 高密度调度 (临界时间限制)

- 自然语言描述

  3 辆列车，5 个站点

  速度相同 $v = 3$，行驶时间 $f(v, j) = j + 1$ 分钟

  所有列车在中间站均停靠

  极限时间 $T = 30$ 分钟，$s = 2$ 分钟，$m = 1$ 分钟，$M = 3$ 分钟

- 数学形式

  $N = 3, n = 5$

  $v = [3, 3, 3]$

  $f(v, j) = j + 1$

  $b = [[1, 1, 1, 1, 1]] * 3$

  $T = 30, s = 2, m = 1, M = 3$

- 代码实现

```
trains = 3
stations = 5
velocities = [3, 3, 3]
def time_by_velocity(v, j): return j+1
if_stop = [[1,1,1,1,1]] * 3
T = 30
m, M, s = 1, 3, 2
```

样例 5 综合压力测试 (大规模复杂场景)

- 自然语言描述

  5 辆列车，5 个站点

  速度梯度分布 $[1, 3, 5, 7, 9]$，行驶时间 $f(v, j) = \text{ceil}(10/(v * (j + 1)))$

  随机生成停靠矩阵（保证首末站为 1）

  $T = 100$ 分钟，$s = 3$ 分钟，$m = 2$ 分钟，$M = 6$ 分钟

- 数学形式

```
N=5, n=5
v=[1,3,5,7,9]
f(v,j) = math.ceil(10/(v*(j+1)))
b = [
    [1,0,1,0,1],
    [1,1,0,1,1],
    [1,0,0,1,1],
    [1,1,1,0,1],
    [1,0,1,1,1]
]
T=100, s=3, m=2, M=6
```

- 代码实现

```
import math
trains = 5
stations = 5
velocities = [1, 3, 5, 7, 9]
def time_by_velocity(v, j):
    return math.ceil(10 / (v*(j+1)))

if_stop = [
    [1,0,1,0,1],
    [1,1,0,1,1],
    [1,0,0,1,1],
    [1,1,1,0,1],
    [1,0,1,1,1]
]
T = 100
m, M, s = 2, 6, 3
```

复杂度分析:

1. 样例 1：仅验证单列车基本逻辑
2. 样例 2：测试车头时距约束
3. 样例 3：引入不同速度与动态行驶时间
4. 样例 4：临界时间下的调度可行性
5. 样例 5：大规模复杂场景下的综合优化

每组样例均覆盖不同的约束组合，可用于逐步验证模型正确性和求解器性能。

可以发现，这里的样例设置的确合理，虽然其中仍然存在一些问题，例如样例 3 的情况并不是真正的“临界”，可以以更极限的情况进行检测，样例 5 的数学描述直接采用了代码等。进行进一步的生成与筛选后，即可得到符合要求的样例。

14.3.3 强化学习

本部分对应实验要求中第七题。

除了大语言模型外，还可以通过强化学习解决此问题。仍然沿用之前的记号，我们将每辆列车 G_i 作为动作的主体，每分钟视为一步对所有列车同时进行，具体状态转移如下：

1. 每辆列车 G_i 具有状态：起始 (未发动) 共 $T+1$ 个、A_j 前往 A_{j+1} 路上共 $f(v_i)_j - 1$ 个状态 (每分钟前进一个)、对 $j = 2, \ldots, n-1$，若其在 A_j 站停则有 $M+1$ 个状态 (刚停下直到上限)，否则只有一个到达状态。最终有唯一一个到达 A_n 状态。额外需要一个失败状态，代表最后一个时间点仍为到达终点。

2. 每辆 G_i 在起始状态或任何 $j = 2, \ldots, n-1$ 的非最后一个停靠状态 (即尚未停留 M 分钟) 以一定概率 (初始如停靠越晚出发概率越大) 选择出发或等待，出发则进入下一个前进状态，等待则进入下一个停靠状态。最后一个停靠状态、前进状态或不停留的到达状态只能选择前进。

我们需要整体的奖励函数以改进列车的策略，更准确来说，整体的奖励函数将承担处理车头时距约束、优化目标函数的任务。上述的转移代表每分钟列车的状态，而若在五分钟以内两辆列车在同一站停靠 (注意这直接处理了**原始**的约束，而非我们简化后的版本)，即给予一定的惩罚，冲突时间越长则惩罚越重。此外，若某个 G_i 在 T 到达上限后仍为到终点，进入失败状态，也给予一定的惩罚。根据不同的约束条件，可给出不同的奖励，如考虑总停留时间最短则在停留时间越短时对转移入出发状态奖励越高。

最终，每辆列车将接收同一时刻其他所有列车的状态 (也即状态实际上是**整体**的) 并进行转移，在奖励函数下学习出最优策略。