

# Group LASSO Problem 编程作业报告

车保骏 2501110040

2025 年 12 月 21 日

## 1 问题描述

算法需要求解的 Group LASSO 问题为：

$$\min_{x \in \mathbb{R}^{n \times l}} \frac{1}{2} \|Ax - b\|_F^2 + \mu \|x\|_{1,2} \quad (1)$$

其中  $A \in \mathbb{R}^{m \times n}$ ,  $b \in \mathbb{R}^{m \times l}$ ,  $\mu > 0$ ,

$$\|x\|_{1,2} := \sum_{i=1}^n \|x_{i,:}\|_2,$$

其中  $x_{i,:}$  是矩阵  $x$  的第  $i$  行。

## 2 测试数据生成

本文使用算法 1 生成测试数据，取 `numpy.random.seed` 随机种子为 97006855，取  $n = 512$ ,  $m = 256$ ,  $l = 2$ ,  $\mu = 0.01$ ,  $sparse = 0.1$ 。

## 3 评价指标及结果总览

项目完成了多个算法的实现，为了实现不同结果效率之间的对比，本项目将使用以下指标来评价各个算法的性能：

- **Fval**: 算法得到的最终目标函数值
- **Errfun**: 算法的解  $x$  和 `cvx-mosek` 的解  $x^*$  的相对距离  $\frac{\|x - x^*\|_F}{1 + \|x^*\|_F}$
- **ErrExact**: 算法的解  $x$  和用于生成数据的  $u$  的相对距离  $\frac{\|x - u\|_F}{1 + \|u\|_F}$
- **Time**: 算法运行时间（单位：秒）

---

**Algorithm 1** 生成测试数据的伪代码

---

**输入:** 种子  $seed$ , 维度  $n, m, l$ , 正则项系数  $\mu$ , 稀疏度  $sparse$

**输出:** 矩阵  $A$ , 真实解  $u$ , 观测值  $b$ , 初始值  $x_0$

- 1: 设置随机种子: `np.random.seed(seed)`
  - 2: 计算非零元素个数:  $k \leftarrow \text{round}(n \times sparse)$
  - 3: 生成随机矩阵:  $A \leftarrow \text{np.random.randn}(m, n)$
  - 4: 随机选择支撑集:  $p \leftarrow \text{np.random.permutation}(n)[:k]$
  - 5: 初始化零矩阵:  $u \leftarrow \text{np.zeros}((n, l))$
  - 6: 在支撑集上赋值:  $u[p, :] \leftarrow \text{np.random.randn}(k, l)$
  - 7: 计算观测值:  $b \leftarrow Au$
  - 8: 生成初始值:  $x_0 \leftarrow \text{np.random.randn}(n, l)$
  - 9: **return**  $A, u, b, x_0$
- 

- **Iter:** 算法迭代次数<sup>1</sup>
- **Sparsity:** 算法得到的最终解的稀疏度, 使用

$$\text{Sparsity} = \frac{1}{nl} \#\{(i, j) : |x_{ij}| > 10^{-6} \cdot \max_{k,l} |x_{kl}|\}$$

计算, 数值越小说明越稀疏。

本文涉及的所有算法在上述评价指标下的表现见表 1。其中, 由于 **Errfun** 以 `cvx-mosek` 为参考, 所以 `cvx-mosek` 一行未标出 **Errfun**; 调用 `mosek` 或 `gurobi` 求解器的算法均无法获得迭代次数。此外, 以 `cvx-mosek` 的最优值  $f^*$  为参考, 我们还计算了部分算法迭代中相对误差  $\frac{f^k - f^*}{f^*}$  的变化, 见图 1。

## 4 调用求解器求解

### 4.1 CVX 调用 Mosek 和 Gurobi 求解器

由表 1 可见, 使用 CVX 调用 Mosek 和 Gurobi 求解器的效率相当。两者在最优值上的计算结果几乎相同, 在运行时间和解的稀疏度上也很相近。

---

<sup>1</sup>对于增广拉格朗日法解对偶问题 (ALM-dual), 由于内循环中不更新  $x$ , 所以迭代次数指外循环的迭代次数。

Solver	Fval	Errfun	ErrExact	Time(s)	Iter	Sparsity
cvx_mosek	0.67057522	–	3.56e-04	1.8220	–	0.1035
cvx_gurobi	0.67057522	7.66e-06	3.61e-04	2.1429	–	0.1055
mosek_direct	0.67057522	4.33e-07	3.56e-04	0.4263	–	0.1025
gurobi_direct	0.67057522	1.66e-06	3.57e-04	0.9032	–	0.1025
SGD_primal	0.67059517	6.93e-04	9.66e-04	4.7122	2357	0.2852
GD_primal	0.67058665	3.30e-04	6.53e-04	2.7558	1381	0.5156
ProxGD_primal	0.67057599	1.26e-04	4.65e-04	0.1868	643	0.1357
FProxGD_primal	0.67057528	3.22e-05	3.37e-04	0.1202	253	0.1025
ALM_dual	0.67058907	3.71e-04	6.71e-04	0.7970	9	0.0996
ADMM_dual	0.67058733	3.33e-04	6.44e-04	0.0704	70	0.0996
ADMM_primal	0.67060069	1.62e-04	4.46e-04	0.2093	569	0.6055

表 1: 优化算法性能比较

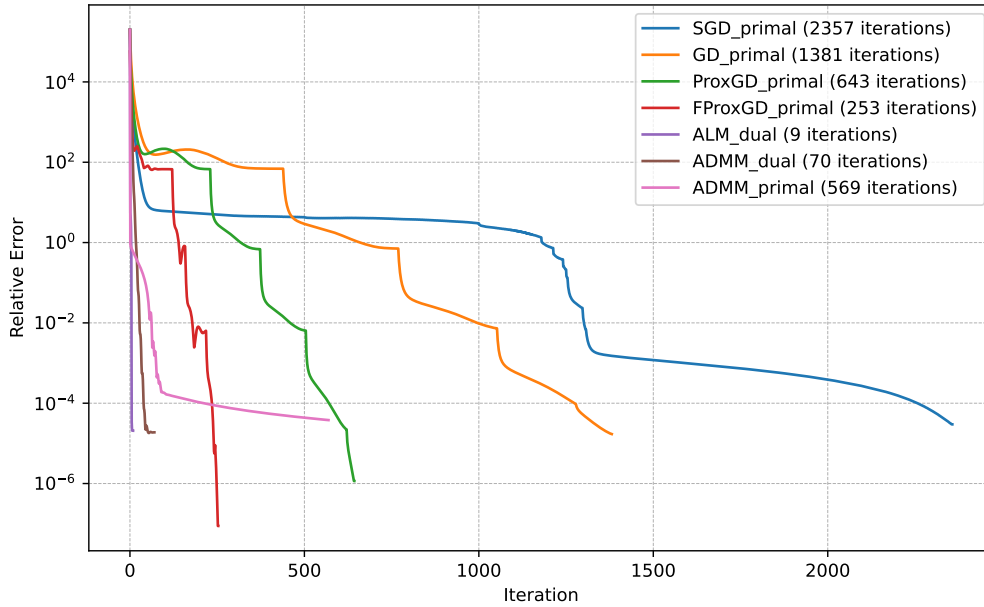


图 1: 不同算法迭代中相对误差  $\frac{f^k - f^*}{f^*}$  的变化

## 4.2 直接调用 Mosek 和 Gurobi 求解

为了使原问题能直接被 Mosek 和 Gurobi 求解（不使用 CVXPY），我们引入辅助变量  $t$ ，构造二阶锥规划。等价的 SOCP-QP 模型为：

$$\begin{aligned} \min_{x \in \mathbb{R}^{nl}, t \in \mathbb{R}^n} \quad & \frac{1}{2} \|Ax - b\|_F^2 + \mu \sum_{i=1}^n t_i \\ \text{s.t.} \quad & \|x_{i,:}\|_2 \leq t_i, \quad i = 1, \dots, n. \end{aligned}$$

这就可以直接调用 Mosek 和 Gurobi 求解。表 1 的结果显示，直接调用两种求解器的求解速度明显快于借助 CVX，且获得的解的稀疏性稍高。相对而言，直接调用 Mosek 比直接调用 Gurobi 求解的速度更快。

## 5 其它算法

### 5.1 连续化策略

对于 Group Lasso 原始问题 (1)，我们采用连续化策略：从一个较大的正则化参数  $\mu_t$  开始，逐步减小至目标值  $\mu$ ，并在每个阶段求解对应的 LASSO 问题：

$$\min_{x \in \mathbb{R}^{n \times l}} f_{\mu_t}(x) := \frac{1}{2} \|Ax - b\|_F^2 + \mu_t \|x\|_{1,2}. \quad (2)$$

该策略的优势在于：在求解当前  $\mu_t$  对应的问题时，可以将前一阶段（对应  $\mu_{t-1}$ ）得到的最优解作为高质量的初始点（仅对第一个子问题  $\mu_1$  使用随机初始点）。由于较大的  $\mu_t$  会使问题具有更强的正则性，解更稀疏且优化问题更易求解。通过这种方式，连续化策略将原始困难问题转化为一系列由易到难的子问题，利用前序解提供良好初值，从而显著加速整体求解过程。

在本项目中，SGD 算法、GD 算法、ProxGD 算法和 FProxGD 算法使用连续化策略，具体结果见下面的小节。

### 5.2 原问题的对偶问题

对于 Group Lasso 原始问题 (1)，可以考虑通过求解其对偶问题求解原问题。下面做一下对偶问题的推导。首先引入辅助变量  $z = Ax - b$ ，则原问题等价于

$$\min_{x, z} \frac{1}{2} \|z\|_F^2 + \mu \|x\|_{1,2} \quad \text{s.t.} \quad z = Ax - b.$$

构造拉格朗日函数（对偶变量为  $y \in \mathbb{R}^{m \times l}$ ）：

$$\mathcal{L}(x, z; y) = \frac{1}{2} \|z\|_F^2 + \mu \|x\|_{1,2} + \langle y, Ax - b - z \rangle,$$

并对  $(z, x)$  求极小：

- 对  $z$  最小化:  $\min_z \left\{ \frac{1}{2} \|z\|_F^2 - \langle y, z \rangle \right\}$  在  $z = y$  处取得最小值, 代入得  $-\frac{1}{2} \|y\|_F^2$ ;
- 对  $x$  最小化:  $\min_x \left\{ \mu \|x\|_{1,2} + \langle A^\top y, x \rangle \right\} = \begin{cases} 0, & \text{if } \|(A^\top y)_{i,:}\|_2 \leq \mu, \forall i, \\ -\infty, & \text{otherwise.} \end{cases}$

将上述结果代入拉格朗日对偶函数可得

$$g(y) = \inf_{x,z} \mathcal{L}(x, z; y) = \begin{cases} -\frac{1}{2} \|y\|_F^2 - \langle y, b \rangle, & \text{if } \|(A^\top y)_{i,:}\|_2 \leq \mu, \forall i, \\ -\infty, & \text{otherwise.} \end{cases}$$

因此, (1) 的对偶问题为:

$$\max_{y \in \mathbb{R}^{m \times l}} -\frac{1}{2} \|y\|_F^2 - \langle y, b \rangle \quad \text{s.t.} \quad \|(A^\top y)_{i,:}\|_2 \leq \mu, \quad i = 1, \dots, n.$$

等价地, 可写为最小化形式:

$$\min_{y \in \mathbb{R}^{m \times l}} \frac{1}{2} \|y\|_F^2 + \langle y, b \rangle \quad \text{s.t.} \quad \|(A^\top y)_{i,:}\|_2 \leq \mu, \quad i = 1, \dots, n. \quad (3)$$

### 5.3 次梯度算法

为使用次梯度法 (Subgradient Descent) 求解, 我们把正则项  $\|x\|_{1,2}$  的次梯度取为:

$$[\partial \|x\|_{1,2}]_{i,:} = \begin{cases} \frac{x_{i,:}}{\|x_{i,:}\|_2} & \text{if } \|x_{i,:}\|_2 > \epsilon = 10^{-6}, \\ \frac{x_{i,:}}{1 + \|x_{i,:}\|_2} & \text{otherwise.} \end{cases} \quad (4)$$

我们使用连续化策略, 采用外层循环调整参数、内层执行次梯度更新的双层循环策略。对于内层循环, 我们使用前期固定、后期消失的步长  $\alpha_j = \min(\alpha_{\max}, \frac{\alpha}{j+1})$ , 并在相邻两次  $f_{\mu_t}$  差值小于  $tol$  时停止内循环。每次内循环结束, 我们缩小参数  $tol$  和  $\alpha_{\max}$ 。经尝试, 选取第  $i$  轮外循环 (设最多  $N$  轮) 的  $\mu_t$  为

$$\begin{aligned} \mu_t &= \text{cos\_annealing}(i, N, \mu, \mu_{\max}) \\ &:= \mu + \frac{\mu_{\max} - \mu}{2} \left( 1 + \cos \left( \frac{i}{N} \pi \right) \right). \end{aligned} \quad (5)$$

算法 2 描述了 SGD 的更新过程。我们取  $\mu_{\max} = 2.0$ , 内循环最大步数  $N_{\text{inner}}$  为 500 (但当  $\mu_t = \mu$  时允许迭代 5000 步), 外循环最大步数  $N_{\text{outer}}$  为 10,  $\alpha_{\max}$  初值为 0.001,  $\alpha = 0.5$ ,  $tol$  初值为  $10^{-3}$ 。其它细节详见代码。

从图 1 和表 1 中可以看出次梯度法收敛较慢, 且最终得到的解稀疏性较差。

---

**Algorithm 2** Group LASSO 随机次梯度下降算法 (SGD)

---

**输入:** 初始点  $x_0$ , 矩阵  $A$ , 观测值  $b$ , 正则参数  $\mu$

**输出:** 最优解  $x_{\text{opt}}$ , 迭代次数, 函数值序列

```
1: 初始化  $x \leftarrow x_0, x_{\text{opt}} \leftarrow x_0, f_{\text{best}} \leftarrow f(x_0), k \leftarrow 0$ 
2: for out_iter = 0 to  $N_{\text{outer}} - 1$  do
3:    $\mu_t \leftarrow \text{cos\_annealing}(\text{out\_iter}, N_{\text{outer}}, \mu, \mu_{\text{max}})$ 
4:   if out_iter ==  $N_{\text{outer}} - 1$  then
5:      $flag \leftarrow \text{True}, tol \leftarrow 10^{-7}, N_{\text{inner}} \leftarrow 5000$ 
6:   else
7:      $flag \leftarrow \text{False}, N_{\text{inner}} \leftarrow 500$ 
8:   end if
9:   for  $j = 0$  to  $N_{\text{inner}} - 1$  do
10:    计算当前目标值  $f^k \leftarrow f(x)$ 
11:    if  $f^k < f_{\text{best}}$  then
12:       $x_{\text{opt}} \leftarrow x, f_{\text{best}} \leftarrow f^k$ 
13:    end if
14:    if  $j \geq 1$  and  $|f_{\mu_t}(x) - f_{\mu_t}(x_{\text{prev}})| < tol$  then
15:      break
16:    end if
17:     $x_{\text{prev}} \leftarrow x$ 
18:    计算次梯度  $g_k \leftarrow A^\top(Ax - b) + \mu_t \cdot \partial\|x\|_{1,2}$  ▷ 见式 (4)
19:    计算步长  $\alpha_j \leftarrow \min(\alpha_{\text{max}}, \frac{\alpha}{j+1})$ 
20:    更新  $x \leftarrow x - \alpha_j g_k$ 
21:     $k \leftarrow k + 1$ 
22:    if  $k \geq 5000$  then
23:       $flag \leftarrow \text{True}$ 
24:      break
25:    end if
26:  end for
27:  if  $flag$  then
28:    break
29:  end if
30:  更新参数:  $tol \leftarrow \max(tol \cdot 0.8, 10^{-6}), \alpha_{\text{max}} \leftarrow \alpha_{\text{max}} \cdot 0.9$ 
31: end for
32: return  $x_{\text{opt}}, k$ , 函数值序列  $\{f^k\}$ 
```

---

## 5.4 光滑化梯度法

我们考虑对正则项  $\|x\|_{1,2}$  做光滑化后，取梯度为：

$$[\nabla g(x, \epsilon)]_{i,:} = \begin{cases} \frac{x_{i,:}}{\|x_{i,:}\|_2} & \text{if } \|x_{i,:}\|_2 > \epsilon, \\ \frac{x_{i,:}}{\epsilon} & \text{otherwise.} \end{cases} \quad (6)$$

光滑化梯度法的迭代过程类似于次梯度法，但在部分细节上有区别：

- $tol$  和  $\mu_t$  每次内循环开始时缩小到原来的 0.1，其中  $\mu_t$  控制为不小于  $\mu$ 。  $\mu_t$  的初值选为 200。
- 采用式 6 的光滑化梯度替代次梯度，其中  $\epsilon = \epsilon_t = 10^{-3}\mu_t$ 。此时子问题的梯度 Lipschitz 常数可选为  $L_t = \|A\|_2^2 + \frac{\mu_t}{\epsilon_t} = \|A\|_2^2 + 10^3$ 。因此所有内循环均固定步长为  $\frac{1}{L_t}$ 。
- 超参数变化：设置所有内循环总数上限为 5000，每次内循环不超过 500 次。设置  $tol$  初值为 0.01。

从表 1 中可以看出光滑化梯度法尽管收敛比次梯度法快，但最终得到的解稀疏性更差，这可能是光滑化处理所导致的。

## 5.5 近端点梯度算法及其 Nesterov 加速

在本项目求解的问题形式下，近似点算子可以逐行计算，解析地写出。记  $h(x) = \mu\|x\|_{1,2}$ ，则

$$[\text{prox}_{th}(z)]_{i,:} = \begin{cases} \left(1 - \frac{t\mu}{\|z_{i,:}\|_2}\right) z_{i,:}, & \text{if } \|z_{i,:}\|_2 > t\mu, \\ 0, & \text{otherwise.} \end{cases} \quad (7)$$

对于近端点梯度算法（ProxGD）我们仍然采用连续化策略：

- 每次内循环做更新：  $z^{k+1} = x^k - tA^\top(Ax^k - b)$ ,  $x^{k+1} = \text{prox}_{th}(z^{k+1})$ ，其中固定步长  $t = \frac{1}{\|A\|_2^2}$ 。
- $tol$  和  $\mu_t$  每次内循环开始时缩小到原来的 0.1，其中  $\mu_t$  控制为不小于  $\mu$ 。  $\mu_t$  的初值选为 200。
- 超参数：设置所有内循环总数上限为 5000，每次内循环不超过 500 次。设置  $tol$  初值为 0.01。

Nesterov 加速是一种常基于动量的加速方法。这里我们选取著名的 FISTA 加速算法做更新，得到 FProxGD 的迭代格式：

$$\begin{aligned} y^k &= x^{k-1} + \frac{k-2}{k+1}(x^{k-1} - x^{k-2}), \\ x^k &= \text{prox}_{th}(y^k - tA^\top(Ay^k - b)), \end{aligned}$$

其中固定步长  $t$  仍取  $\frac{1}{\|A\|_2^2}$ 。

从图 1 和表 1 中可以看出，ProxGD 和 FProxGD 的各方面性能均优于 SGD 和 GD。两者相比，加速后的 FProxGD 算法在各方面性能均优于 ProxGD。

## 5.6 增广拉格朗日法和 ADMM 求解对偶问题

对于原问题的对偶问题 3，我们引入变量  $z = A^\top y$ ，将对偶问题化为：

$$\min_{y \in \mathbb{R}^{m \times l}} \frac{1}{2} \|y\|_F^2 + \langle y, b \rangle \quad (8)$$

$$\text{s.t. } A^\top y + z = 0, \quad \|z_{i,:}\|_2 \leq \mu, \quad i = 1, \dots, n. \quad (9)$$

我们对约束  $A^\top y + z = 0$  引入乘子  $\lambda$ ，定义增广拉格朗日函数

$$\mathcal{L}_\rho(y, z, \lambda) = \frac{1}{2} \|y\|_F^2 + \langle y, b \rangle + \langle \lambda, A^\top y + z \rangle + \frac{\rho}{2} \|A^\top y + z\|_F^2. \quad (10)$$

我们固定  $\mu$ （即不再采用连续化策略），对于每个子问题，以上一个子问题的  $(y, z)$  为初值求解

$$(y, z) = \arg \min_{\|z_{i,:}\|_2 \leq \mu} \mathcal{L}_{\rho_k}(y, z, \lambda^k).$$

求解方式为固定  $z$ ，优化  $y$ ，此时有解析解：

$$y = (I + \rho_k A A^\top)^{-1} (-A \lambda^k + \rho_k A(-z) - b). \quad (11)$$

再将  $y$  固定，优化  $z$ ，此时即为做投影：

$$z = \text{proj}_{\{\|z_{i,:}\|_2 \leq \mu\}} \left( -\frac{1}{\rho_k} \lambda^k - A^\top y \right). \quad (12)$$

ALM 算法交替进行 11 式和 12 式，直到  $z$  的变化值小于  $tol_{\text{inner}}$ 。接着做乘子更新：

$$\lambda^{k+1} = \lambda^k + \rho_k (A^\top y + z). \quad (13)$$

我们选取初值  $tol_{\text{inner}} = 10^{-6}$ ， $\rho_1 = 10$ ，并在每次外循环结束后增大  $\rho_{k+1} = 2\rho_k$ ，将  $tol_{\text{inner}}$  减半。内外循环上限分别为 200 轮、20 轮，相邻两次外循环的目标函数值差小于  $10^{-7}$  则停止循环。

与 ALM 类似，ADMM 求解对偶问题也使用增广拉格朗日函数，区别在于不区分内外循环，而是交替进行 11、12 和 13 的更新。此时， $\rho$  固定为 100，终止条件设最近 20 轮内目标函数值最大值和最小值差小于  $10^{-6}$  或运行 1000 轮则停止循环。

事实上，上述增广拉格朗日方程 10 中的乘子  $\lambda$  就对应原始变量  $x$ 。因此我们可以根据  $\lambda$  的更新作为  $x$  的更新。从表 1 中可以看出，ADMM-dual 的求解效率相当高，且求得的解具有很好的稀疏性。

## 5.7 ADMM 求解原始问题

类似地，我们可以写出原始问题求解的 ADMM 算法：

$$\begin{aligned}x^{k+1} &= (\rho I + A^\top A)^{-1}(A^\top b + \rho z^k - y^k), \\z^{k+1} &= \text{proj}_{\{\|z_{i,:}\|_2 \leq \mu/\rho\}} \left( x^{k+1} + \frac{1}{\rho} y^k \right), \\y^{k+1} &= y^k + \rho(x^{k+1} - z^{k+1}).\end{aligned}$$

此时， $\rho$  固定为 0.5，终止条件设最近 20 轮内目标函数值最大值和最小值差小于  $10^{-6}$  或运行 3000 轮则停止循环。然而，从表 1 和图 1 中可以看出，ADMM-primal 的求解效率和精度都不如 ADMM-dual。

## 6 总结与说明

**声明：**本项目的代码和本报告的书写有借助 Deepseek、Qwen3-max 两个 AI 工具，具体使用场景包括询问其商用求解器调用方法、优化代码结构、搭建本实验报告的基础 LaTeX 框架等。如上述算法描述有细节缺失，请参考提交代码。

本文围绕 Group Lasso 优化问题，系统实现了包括商业求解器调用、一阶优化算法（次梯度法、光滑化梯度法、近端梯度法及其 Nesterov 加速版本）、对偶方法（增广拉格朗日法 ALM 和 ADMM）以及原始问题的 ADMM 算法在内的多种求解策略，并在统一的测试样例下进行了全面性能评估。

实验结果表明：

- 商业求解器 MOSEK、Gurobi 在精度上具有显著优势，尤其在直接建模为二阶锥规划时效率远高于通过 CVXPY 接口调用，说明了问题结构利用的重要性。
- 次梯度法与光滑化梯度法格式简单，但即使精心选取参数和连续化策略仍收敛缓慢，获得的解稀疏性较差，说明这两种方法对于 Group Lasso 优化问题适用性差。
- ProxGD 及其加速版本 FProxGD 表现出极优的综合性能：收敛速度快、解的质量高、稀疏性良好。
- 对偶方法（ALM-dual 与 ADMM-dual）展现出极高的计算效率和良好的稀疏恢复能力。特别是 ADMM-dual 仅需 70 次迭代、0.08 秒即可获得接近最优的解，说明将复杂非光滑项移至约束侧后，投影操作变得简洁高效。
- 相比之下，ADMM-primal 在实际运行中收敛较慢、稀疏性控制不佳（Sparsity 高达 0.6055）。

综上所述，对于中等规模的 Group Lasso 问题，若追求高精度解，应当使用直接调用 MOSEK 的 SOCP 方式；若注重算法实现灵活性，FProxGD 或者 ADMM-dual 是较为高效的算法方案。