基于 Benders-Dual Cutting Plane 算法的两阶段鲁棒优化 Matlab 复现

文献来源：B. Zeng, L. Zhao. *Solving two-stage robust optimization problems using a column-and-constraint generation method*. Operations Research Letters 41 (2013) 457–461.

参考解读：鲁棒优化| C&CG 算法求解两阶段鲁棒优化：全网最完整、最详细的【入门-完整推导-代码实现】笔记

一、 两阶段鲁棒优化（Two-stage Robust Optimization）

【两阶段鲁棒优化】

在传统的确定性优化模型中，我们通常假设所有参数在决策前都是已知的，例如需求、负荷、风光出力等。然而在实际系统中，这些参数往往存在显著不确定性。**鲁棒优化（Robust Optimization, RO）**通过在一个不确定集内寻找**“最坏情形”**的最优解，从而提高方案的抗扰性。但单阶段鲁棒优化把所有决策一次性做完，往往过于保守。为此，引入了**两阶段鲁棒优化（Two-stage RO）**框架：

- **第一阶段：** 在不确定性揭晓前做出的决策。在电力中对应机组组合（Unit Commitment, UC），即决定哪些发电机开机。
- **第二阶段：** 在不确定性（如风电出力实际值）揭晓后做出的调整决策。在电力中对应经济调度（Economic Dispatch, ED），即调整发电机的具体出力。

【两阶段鲁棒模型的典型形式】

$$\min_{y} c^{\mathrm{T}} y + \max_{u\in\mathcal{U}} \min_{x\in F(y,u)} b^{\mathrm{T}} x \tag{1-1}$$

决策变量：y为第一阶段决策（如容量、选址），x为第二阶段决策（如潮流、运输流）

参数不确定性：u为不确定参数（如负荷、需求、新能源等），$\mathcal{U}$为有限的不确定集。

【参考论文所完成的工作】

Zeng & Zhao (2013)的工作，选取**鲁棒选址–运输（Location–Transportation）问题**作为算例，采用**列与约束生成算法（Column-and-Constraint Generation, C&CG）**精确求解两阶段鲁棒模型，本文用 Matlab+YALMIP 复现。

二、 数学模型

1. 通用线性两阶段鲁棒优化模型 Two-Stage RO

【目标函数】

最小化“第一阶段投资成本”$c^{\mathrm{T}}y$和“最恶劣场景下第二阶段运输成本”$b^{\mathrm{T}}x$之和。

$$\min_{y} c^{\mathrm{T}}y + \max_{u\in\mathcal{U}} \min_{x\in F(y,u)} b^{\mathrm{T}}x \tag{2-1-1}$$

【约束方程】

$$Ay \geq d,\ y \in \mathcal{S}_y \tag{2-1-2}$$

$$F(y,u) \coloneqq \{x \in \mathcal{S}_x : Gx \geq h - Ey - Mu\} \tag{2-1-3}$$

其中，约束（2-2）是第一阶段的“先天约束”（如预算、二进制变量等）；约束（2-3）是第二阶段在给定y,u下x的可行性约束（功率平衡、容量限制等）。

2. 选址–运输 Two-Stage RO

【目标函数】

$$\min_{y,z} \sum_i f_i y_i + \sum_i a_i z_i + \sum_i \sum_j c_{ij} x_{ij} \tag{2-2-1}$$

f_i为固定建设成本（fixed cost），a_i为单位容量建设成本（unit capacity cost），c_{ij}为单位运输成本。

第一阶段变量：$y_i \in \{0,1\}$，表示是否在节点i建设设施（facility location variable），z_i表示在节点i的设施容量（the capacity variable）。

第二阶段变量：x_{ij}表示从i到j的运输量（transportation variable），x是一个（源节点数，目的节点数）大小的矩阵变量。

不确定参数：d_j表示客户j的需求，D为多面体不确定集。

该问题决策分为两个阶段：

第一阶段：在需求未知时，决定设施选址和容量。

第二阶段：在需求不确定集揭示后，决定运输方案以满足需求并最小化成本。

【约束方程】

容量建设约束 $z_i \leq K_i y_i, \forall i$ （2-2-2）

其中K_i为节点i的最大允许建设容量。

容量限制（流出量不超过容量） $\sum_j x_{ij} \leq z_i, \forall i$ （2-2-3）

需求限制（流出量满足需求） $\sum_i x_{ij} \geq d_j, \forall j$ （2-2-4）

运输量为正 $$x_{ij} \geq 0 \quad (2\text{-}2\text{-}5)$$

【不确定集D】

论文采用如下方式构建不确定集：

$$d_j = \underline{d}_j + \hat{d}_j g_j \quad (2\text{-}2\text{-}6)$$

$$\text{s.t. } g_j \in \{0,1\}, j = 1,\ldots,n \quad (2\text{-}2\text{-}7)$$

$$\sum_j g_j \leq \Gamma, j = 1,\ldots,n \quad (2\text{-}2\text{-}8)$$

其中，d_j为实际需求，$\underline{d}_j$为基准需求，$\hat{d}_j$为最大需求偏差，g_j为不确定系数（Perturbation Factor），表示偏差发生的程度。

式（2-11）中Γ为“不确定性预算”参数，用于控制保守程度。如果$\Gamma = 0$，表示我们认为所有的$g_j = 0$，完全忽略不确定性；如果Γ等于节点总数，表示允许所有节点同时达到峰值，即考虑最坏的情况，此时通常导致结果过于保守，成本极高。

※不确定集D通过参数Γ让决策者可以在风险中性和极端保守之间找到平衡点，同时利用线性约束来描述变量间的相关性。

3. Two-Stage RO 和 Benders-Dual Cutting Plane Method

【主问题（Master Problem)】

包含第一阶段的决策y、只与y有关的约束、子问题返回的割$\eta \geq 0$，评估第二阶段目标函数取值的辅助变量η

$$\min_y c^{\mathrm{T}} y + \eta \quad (2\text{-}3\text{-}1)$$

$$\text{s.t. } Ay \geq d, \quad (2\text{-}3\text{-}2)$$

$$\mathcal{S}_y \subseteq \mathbb{R}^n_+, \quad (2\text{-}3\text{-}3)$$

$$\eta \geq 0. \quad (2\text{-}3\text{-}4)$$

【子问题（Subproblem）】

$$\max_{u \in \mathcal{U}} \min_{x \in F(y,u)} b^{\mathrm{T}} x \quad (2\text{-}3\text{-}5)$$

$$\text{s.t. } Gx \geq h - Ey - Mu, u \in \mathcal{U} \quad (2\text{-}3\text{-}6)$$

$$\mathcal{S}_x \subseteq \mathbb{R}^m_+, \quad (2\text{-}3\text{-}7)$$

子问题的外层是$\max_{u \in \mathcal{U}}$，决策变量是不确定量u；内层是$\min_{x \in F(y,u)}$，决策变量是x。

从而内层模型是关于x的线性规划，假定已知$\bar{y}$、$\bar{u}$，可以写出内层模型的对偶问题：

$$\max_{\lambda}(h-E\bar{y}-M\bar{u})^{\mathrm{T}}\lambda \tag{2-3-8}$$

$$\text{s.t. } G^{\mathrm{T}}\lambda\le b, \tag{2-3-9}$$

$$\lambda\ge 0. \tag{2-3-10}$$

此时子问题的外层和内层写在一起，可以转化为单层模型：

$$\max_{\lambda,u}(h-E\bar{y}-Mu)^{\mathrm{T}}\lambda \tag{2-3-11}$$

$$\text{s.t. } G^{\mathrm{T}}\lambda\le b, \tag{2-3-12}$$

$$\lambda\ge 0, \tag{2-3-13}$$

$$u\in\mathcal{U}. \tag{2-3-14}$$

写成函数形式：

SP1
$$Q(y)=\max_{\lambda,u}\{(h-E\bar{y}-Mu)^{\mathrm{T}}\lambda: G^{\mathrm{T}}\lambda\le b,\lambda\ge 0,u\in\mathcal{U}\} \tag{2-3-15}$$

其中，$\bar{y}$为固定参数，决策变量是不确定量u和对偶变量λ。如果满足强对偶性，则求解该单层问题得到的最优值即为原子问题的最优值，否则，由弱对偶性，该单层问题的最优值提供了一个下界。【参见9602PPTDuality①】

【求解子问题SP1】

SP1目标函数含有$u^{\mathrm{T}}\lambda$，决策变量之积，引入**双线性项**，是一个bilinear optimization problem。文献采用KKT条件引入Big-M线性化，求解子问题，本文采用Gurobi求解非线性问题。对于给定的第k步迭代中第一阶段得到的y_k，通过求解SP1，得到相应的子问题的解(λ_k^*,u_k^*)，生成割平面：

$$\eta\ge(h-E\bar{y}-Mu_k^*)^{\mathrm{T}}\lambda_k^* \tag{2-3-16}$$

【更新主问题】

将割平面加入到主问题中：

MP1
$$\min_{y} c^{\mathrm{T}}y+\eta \tag{2-3-17}$$

$$\text{s.t. } Ay\ge d, \tag{2-3-18}$$

$$\mathcal{S}_y\subseteq\mathbb{R}_+^n, \tag{2-3-19}$$

$$\eta\ge(h-E\bar{y}-Mu_l^*)^{\mathrm{T}}\lambda_l^*,\forall l\le k \tag{2-3-20}$$

$$\eta\in\mathbb{R}. \tag{2-3-21}$$

其中$\forall l\le k$表示在前k步迭代中，每次产生的割平面都被加入到MP1中。

【求解主问题】

※通过计算全局上界和下界，使上下界收敛，得到最优解。

4. 选址–运输 Two-Stage RO 和 Benders-Dual Cutting Plane Method

【主问题 MP】

$$\min_{y,z} [400\ 414\ 326]\begin{bmatrix}y_0\\y_1\\y_2\end{bmatrix} + [18\ 25\ 20]\begin{bmatrix}z_0\\z_1\\z_2\end{bmatrix} + \eta \tag{2-4-1}$$

$$\text{s.t. } z_i \le K_i y_i, \forall i \tag{2-4-2}$$

$$z \ge 0 \tag{2-4-3}$$

$$\eta \ge 0. \tag{2-4-4}$$

【子问题 SP】

$$\min_x \sum_i \sum_j c_{ij} x_{ij} \tag{2-4-5}$$

容量限制（流出量不超过容量） $$\text{s.t. } \sum_j x_{ij} \le z_i, \forall i \tag{2-4-6}$$

需求限制（流出量满足需求） $$\sum_i x_{ij} \ge d_j, \forall j \tag{2-4-7}$$

运输量为正 $$x_{ij} \ge 0 \tag{2-4-8}$$

子问题的对偶问题SPDP为：

$$\max_{\pi,\lambda} \sum_j d_j \lambda_j - \sum_i z_i \pi_i \tag{2-4-5}$$

$$\text{s.t. } \lambda_j - \pi_i \le c_{ij}, \forall (i,j) \tag{2-4-6}$$

$$\pi_i \ge 0, \tag{2-4-7}$$

$$\lambda_j \ge 0. \tag{2-4-8}$$

推导：构建子问题的对偶问题

引入对偶变量π_i和λ_j，构造拉格朗日函数$\mathcal{L}(x,\pi,\lambda)$：

$$\mathcal{L}(x,\pi,\lambda) = \underbrace{\sum_{i,j} c_{ij}x_{ij}}_{\text{原始目标}} + \underbrace{\sum_i \pi_i \left(\sum_j x_{ij} - z_i\right)}_{\text{供给约束惩罚}} \underbrace{- \sum_j \lambda_j \left(\sum_i x_{ij} - d_j\right)}_{\text{需求约束惩罚}}$$

$$\mathcal{L}(x,\pi,\lambda) = \sum_{i,j} \left(c_{ij} + \pi_i - \lambda_j\right) x_{ij} + \left(\sum_j \lambda_j d_j - \sum_i \pi_i z_i\right)$$

定义对偶函数$g(\pi,\lambda)$：

$$g(\pi,\lambda) = \inf_{x \ge 0} \mathcal{L}(x,\pi,\lambda)$$

$$g(\pi,\lambda) = \begin{cases} \left(\sum_j \lambda_j d_j - \sum_i \pi_i z_i\right), if\ c_{ij} + \pi_i - \lambda_j \ge 0 \\ -\infty, otherwise \end{cases}$$

由于$\inf_{x\geq 0}$，因此只需要x的系数≥ 0，$\inf_{x\geq 0}$即可为有限值。

从而得到子问题的对偶问题（Dual Problem）为：

$$\max_{\pi,\lambda} \sum_j d_j\lambda_j - \sum_i z_i\pi_i$$

$$\text{s.t. } \lambda_j - \pi_i \leq c_{ij}, \forall(i,j)$$

$$\pi_i \geq 0,$$

$$\lambda_j \geq 0.$$

在SPDP的目标函数中，d为不确定参数，并与对偶变量λ乘积，即**双线性项**$d_j\lambda_j$，导致了非凸性。

代入（2-9）$d_j = \underline{d}_j + \widehat{d}_j g_j$，并加上不确定集约束（2-2-7）（2-2-8）：

$$\max_{\pi,\lambda} \sum_j \left(\underline{d}_j + \widehat{d}_j g_j\right)\lambda_j - \sum_i z_i\pi_i \quad (2\text{-}4\text{-}9)$$

$$\text{s.t. } \lambda_j - \pi_i \leq c_{ij}, \forall(i,j) \quad (2\text{-}4\text{-}10)$$

$$\pi_i \geq 0, \quad (2\text{-}4\text{-}11)$$

$$\lambda_j \geq 0. \quad (2\text{-}4\text{-}12)$$

$$0 \leq g_j \leq 1, \forall j \quad (2\text{-}4\text{-}13)$$

$$\sum_j g_j \leq \Gamma, \forall j \quad (2\text{-}4\text{-}14)$$

三、 编写代码

1. 选址–运输 Two-Stage RO 和 Benders-Dual Cutting Plane Method

【参数定义】

```
%% 1. 参数定义 (Parameter Definition)
num_nodes = 3; %3 个节点
f = [400; 414; 326];    % 固定建设成本 Fixed cost
a = [18; 25; 20];       % 单位容量建设成本 Unit capacity cost
C_trans = [             % 单位运输成本 Transportation cost
    22, 33, 24;
    33, 23, 30;
    20, 25, 27
];
K = [800; 800; 800];% 最大允许建设容量

% 不确定集参数
d_bar = [206; 274; 220]; % 基准需求
d_hat = [40; 40; 40]; % 最大需求偏差
```

```
Gamma = 1.8;
```

由于后文将讨论在问题中加入容量下限约束，因此设置惩罚系数，并计算最大可能总需求：

```
%  惩罚系数 (Penalty Cost): 代表切负荷的高昂成本。
%  它的作用是保证子问题永远可行。
%  只要这个值比正常的运输成本(20-30)大很多即可。
PENALTY_COST = 5000;
% [附录 A-4.3] 计算最大可能得总需求，用于主问题预约束
% 简单估算：所有基准值 + Gamma * 最大偏差 (最保守估计)
Max_Total_Demand_Est = sum(d_bar) + Gamma * max(d_hat);
```

【主问题：初始化】

```
%% 2. 初始化
LB = -inf;
UB = inf;
epsilon = 1e-3;
Max_Iter = 30;

Cut_Pi = []; % = pi_sp 由子问题对偶问题求解得到 pi_sp 具体数值，用于主问题的割平面
Cut_Constant = []; % = sum(lambda_sp .* d) 由子问题对偶问题求解得到 lambda_sp、g 具体数值，用于主问题的割平面

% 主问题变量
y = binvar(num_nodes, 1);% 0,1 变量(3,1)
z = sdpvar(num_nodes, 1);% 连续决策变量(3,1)
eta = sdpvar(1, 1);% 子问题辅助变量(1,1)

fprintf('Starting Robust Benders-Dual (Penalized Method)...\n');
fprintf('%-5s | %-12s | %-12s | %-10s\n', 'Iter', 'LB', 'UB', 'Gap');
```

【开始迭代】

```
for iter = 1:Max_Iter
```

【主问题：目标函数】

```
    %% 3. 求解主问题 (Master Problem)
    % 目标函数
    Obj_MP = f'*y + a'*z + eta;
```

【主问题：约束】

```
% 主问题约束
Constraints_MP = [];
% 容量建设约束
Constraints_MP = [Constraints_MP, z <= K .* y];
% 设施容量非负
Constraints_MP = [Constraints_MP, z >= 0];
% 第二阶段运输成本非负
Constraints_MP = [Constraints_MP, eta >= 0];
% [附录 A-4.3] 添加总容量下限约束
% 这能避免第一轮 z=0 导致的无界，加速收敛
Constraints_MP = [Constraints_MP, sum(z) >= Max_Total_Demand_Est];
% 添加 Benders Cuts
% eta >= pi_k * z + sum(lambda_k * d_k)
for k = 1:length(Cut_Constant)
    pi_vec = Cut_Pi(:, k);
    const_val = Cut_Constant(k);
    %
    Constraints_MP = [Constraints_MP, eta >= pi_vec' * z + const_val];
end
```

※ 主问题约束必须添加一个总容量的下限约束。否则会出现子问题对偶无界!

若无，总容量的下限约束，求解问题，在matlab代码求解中出现了报错：

```
Starting Robust Benders-Dual (Penalized Method)...
Iter  | LB          | UB          | Gap
错误使用 TwoStageROBendersDual (第 180 行)
Subproblem Failed. Status: 2. Info: Unbounded objective function (learn to
debug) (GUROBI-NONCONVEX). Please check
Solver!
```

子问题对偶无界⇔子问题不可行。表示主问题求解得到的容量方案z^*无法满足最坏场景下的需求，即出现了**切负荷（未满足的需求）**。在标准 Benders 分解中，这需要添加 Feasibility Cut（可行性割）。

论文的处理方法就是增加一个总容量下限约束，防止主问题在第一轮迭代时给出$z^* = 0$这种显而易见的不可行解。

【主问题：求解】

```
% 求解
ops = sdpsettings('solver', 'gurobi', 'verbose', 0);
```

```
    sol_MP = optimize(Constraints_MP, Obj_MP, ops);
```

【主问题：得到求解结果，更新主问题下界】

```
    if sol_MP.problem == 0
        % 求解成功，更新 y、z 值传递给子问题
        y_star = value(y);
        z_star = value(z);
        eta_star = value(eta);
        % 更新主问题的下界
        Current_LB = value(Obj_MP);
    else
        % 第一代若无界(unbounded)，手动给初值
        if iter == 1
            y_star = [1;1;1]; z_star = K; eta_star = 0; Current_LB = -inf;
        else
            error('Master Problem Failed. Error Code: %d. Msg: %s',
sol_MP.problem, sol_MP.info);
        end
    end
    LB = Current_LB;
```

【子问题：初始化】

```
    %% 4. 求解子问题 (Subproblem)

    % 对偶变量
    % pi_sp: 对应容量约束 z - sum(x) >= 0
    % lambda_sp: 对应需求约束 sum(x) - d >= 0
    pi_sp = sdpvar(num_nodes, 1);
    lambda_sp = sdpvar(num_nodes, 1);

    % 不确定变量 g(3,1)
    g = sdpvar(num_nodes, 1);
    % 实际需求
    d = d_bar + d_hat .* g;
```

【子问题：目标函数】

```
       % 子问题对偶问题的目标函数
    Obj_SPDP=sum(d.*lambda_sp)-sum(z_star.*pi_sp);
```

【子问题：约束】

```
    % 约束
```

```
    Constraints_SPDP=[];
    % 对偶约束: C_ij + pi_i - lambda_j >= 0
    for i = 1:num_nodes
        for j = 1:num_nodes
            Constraints_SPDP = [Constraints_SPDP, C_trans(i,j) + pi_sp(i) -
lambda_sp(j) >= 0];
        end
    end
    % 对偶变量非负
    Constraints_SPDP = [Constraints_SPDP,pi_sp>=0];
    Constraints_SPDP = [Constraints_SPDP,lambda_sp>=0];
    % 【关键修正】 对偶变量上界 (防止 Unbounded)
    % 只要系统出现不可行（容量不足），lambda 和 pi 就会顶到这个上界
    % 这相当于告诉主问题：由于容量不足，产生了巨大的惩罚成本
    Constraints_SPDP = [Constraints_SPDP, lambda_sp <= PENALTY_COST];
    Constraints_SPDP = [Constraints_SPDP, pi_sp <= PENALTY_COST];

    % 不确定集约束
    Constraints_SPDP = [Constraints_SPDP,g>=0,g<=1];% 范围
    Constraints_SPDP = [Constraints_SPDP,sum(g)<=Gamma];% 总预算
    Constraints_SPDP = [Constraints_SPDP,g(1)+g(2)<=1.2];% 局部约束
```

※ 子问题中对应主问题的容量下限约束，需要增加对偶变量上限约束。

给对偶变量加**上界**$Penalty$，等价于给原问题加**松弛变量**，使得子问题的需求约束（2-4-7）变为：

$$\sum_i x_i + s \geq d, s > 0$$

上式的s为切负荷量。子问题的目标函数增加了惩罚项，变为$\min cost + Penalty \times s$。

场景还原：

当主问题传递过来的容量方案z^*很小（例如第一轮迭代$z^* = 0$），而子问题发现最坏情况下的需求$d > 0$时：

- *原问题视角： 要求$\sum_i x_i \geq d$，但容量$z^* = 0$，这在物理上是不可能的。因此，原问题不可行。*
- *对偶问题视角： 需求的影子价格（Shadow Price，即对偶变量λ）代表了“每增加一单位供给能减少多少违约”。既然完全无法满足，$\lambda \to \infty$，导致对偶目标函数$\max_{\pi,\lambda} \sum_j d_j\lambda_j - \sum_i z_i\pi_i \to \infty$，即对偶无界。*

【子问题：求解】

```
    % 求解
    ops = sdpsettings('solver', 'gurobi', 'verbose', 0);
    ops.gurobi.NonConvex = 2;% 解决 g*lambda 的非凸性
    sol_SP = optimize(Constraints_SPDP, -Obj_SPDP, ops); % Maximize

    if sol_SP.problem ~= 0
        % 若报错，显示原因
        error('Subproblem Failed. Status: %d. Info: %s. Please check
Solver!', sol_SP.problem, sol_SP.info);
    end
```

【子问题：得到求解结果】

```
    % 提取结果
    eta_val = value(Obj_SPDP);

    % 注意：我们在 Cut 里需要的 pi 是对应 "sum(x) <= z" 的对偶变量（通常 <= 0）。
    % 我们的 KKT 里定义的是 "z - sum(x) >= 0" 的对偶变量 pi_sp >= 0。
    % 它们的关系是：Value(Cut_pi) = -Value(pi_sp)
    % 物理意义：增加容量 z，会降低成本，所以 z 的系数应为负。
    pi_val = -value(pi_sp);
    lambda_val = value(lambda_sp);
    d_val = value(d);
```

【更新主问题上界】

```
    % 更新 UB
    Current_UB = f'*y_star + a'*z_star + eta_val;
    UB = min(UB, Current_UB);
```

【检查主问题上下界是否收敛，若收敛，求解成功】

```
    % 主问题上下界是否收敛
    Gap = abs(UB - LB)/abs(LB);
    fprintf('%-5d | %-12.2f | %-12.2f | %-10.4f\n', iter, LB, UB, Gap);

    if Gap <= epsilon
        % 主问题上下界收敛，完成求解
        fprintf('Converged successfully!\n');
        break;
```

```
    end
```

【若不收敛，更新割平面变量，继续迭代】

```
    % 更新 Cut
    Cut_Pi = [Cut_Pi, pi_val];
    Cut_Constant = [Cut_Constant, lambda_val' * d_val];

end
```

【求解结果】

```
Starting Robust Benders-Dual (Penalized Method)...
Iter  | LB          | UB          | Gap
1     | 14296.00    | 35238.00    | 1.4649
2     | 14473.67    | 35238.00    | 1.4346
3     | 14636.55    | 35238.00    | 1.4075
4     | 14762.39    | 35238.00    | 1.3870
5     | 14801.29    | 35238.00    | 1.3807
6     | 14903.22    | 35238.00    | 1.3645
7     | 15103.67    | 35238.00    | 1.3331
8     | 15240.38    | 35238.00    | 1.3121
9     | 30532.00    | 34556.00    | 0.1318
10    | 30936.84    | 34556.00    | 0.1170
11    | 30948.50    | 34556.00    | 0.1166
12    | 31015.98    | 34556.00    | 0.1141
13    | 31252.16    | 34556.00    | 0.1057
14    | 31354.12    | 34556.00    | 0.1021
15    | 31374.69    | 34556.00    | 0.1014
16    | 33126.91    | 33680.00    | 0.0167
17    | 33130.93    | 33680.00    | 0.0166
18    | 33545.26    | 33680.00    | 0.0040
19    | 33680.00    | 33680.00    | 0.0000
Converged successfully!
=== Final Results ===
Facilities Built (y): [1 0 1]
Capacities Built (z): [255 0 517]
Worst-Case Total Cost: 33680.00
```

相较于论文结果（Table 1），本文 BD 方法的迭代次数更多（在第 9 次迭代才对应论文的第 2 次迭代），最终收敛值一致，完成 BD 割平面方法的复现

2.

3. Two-Stage RO 和 C&CG

若不确定集$\mathcal{U}$是有限离散集合$\{u^1, ..., u^r\}$，可写成“场景展开”的形式：

$$\min_{y,\eta,\{x^l\}} c^{\mathrm{T}}y + \eta$$

$$\text{s.t. } Ay \geq d$$

$$\eta \geq b^{\mathrm{T}}x^l, l = 1, ..., r$$

$$Ey + Gx^l \geq h - Mu^l, l = 1, ..., r$$

$$y \in \mathcal{S}_y, x^l \in \mathcal{S}_x$$

其中η代表“最坏场景下的二阶段成本”，即：

$$\eta = \max_l b^{\mathrm{T}}x^l$$

■ 如果**直接枚举所有场景**，就得到一个“大但单体”的混合整数规划：

$$\min_y c^{\mathrm{T}}y + \max_{l=1,...,r} \min_{x^l} b^{\mathrm{T}}x^l = \min_y c^{\mathrm{T}}y + \max\left\{\min_{x^1} b^{\mathrm{T}}x^1, ..., \min_{x^l} b^{\mathrm{T}}x^l\right\}$$

这里有三层：

- 最外层：$\min_y$：第一阶段决策；
- 中间层：$\max_{l=1,...,r}$：选一个“**最坏场景**”；
- 最内层：$\min_{x^l}$：在场景l下做第二阶段修正决策。

既然场景是有限个 $l = 1, ..., r$，那么中间层的 max 可以通过一个辅助变量η展开成常见的线性形式，即式（1-）。此时问题将所有场景的变量和约束都放在一个模型中，由同一个求解器一次性解决。

■ C&CG 的思想是：**只为当前“重要场景”生成对应的二阶段变量和约束，逐步扩充场景集合，直到上下界收敛。**

【主问题】

$$\min_{y,\eta,\{x^l\}_{l\in\mathcal{O}}} c^{\mathrm{T}}y + \eta \quad (1\text{-}1)$$

$$\text{s.t. } Ay \geq d \quad (1\text{-}1)$$

$$\eta \geq b^{\mathrm{T}}x^l, \forall l \in \mathcal{O} \quad (1\text{-}1)$$

$$Ey + Gx^l \geq h - Mu^l, \forall l \in \mathcal{O} \quad (1\text{-}1)$$

$$y \in \mathcal{S}_y, x^l \in \mathcal{S}_x, \eta \in \mathbb{R} \quad (1\text{-}1)$$

【子问题】在当前y下，寻找使二阶段成本最坏的场景u：

$$Q(y) = \max_{u \in \mathcal{U}} \min_{x \in F(y,u)} b^{\mathrm{T}} x \tag{1-1}$$

四、 数学模型

1. 光伏多时段经济调度模型

【目标函数】

$$\min \sum_{t=1}^{T} \sum_{i=1}^{N} \left[\left(C_i^1 P_{g,i,t} + C_i^0 x_i + + C_{\mathrm{pv}} P_{\mathrm{pv},t} \right) \right] \tag{1-1}$$

决策变量为所有发电机**有功出力**$P_{g,i,t}$和**开关机状态**$\boldsymbol{x}$，为 0 表示停机，为 1 表示开机，在只考虑经济调度时，机组状态在所考虑的时段内不变。

$P_{\mathrm{pv},t}$为光伏在第 t 个调度周期内的出力，N为机组数目，T为调度周期数。

【等式约束】

系统负荷平衡约束：

$$\begin{cases} P_{\mathrm{bus},t} = P_{g,t} + \boldsymbol{P_{\mathrm{pv},t}} - D_t \\ \sum P_{\mathrm{bus}} = 0, \forall t \end{cases} \tag{1-2}$$

【不等式约束】

机组出力上下限约束：

$$x_i P_{g,i}^{\min} \le P_{g,i} \le x_i P_{g,i}^{\max} \tag{1-3}$$

机组爬坡约束:

$$\begin{cases} P_{g,i,t+1} - P_{g,i,t} \le R_i^u x_{i,t} \\ P_{g,i,t} - P_{g,i,t+1} \le R_i^d x_{i,t+1} \end{cases} \tag{1-4}$$

R_i^u和R_i^d是机组 i 运行时上下爬坡的最大速率。

线路潮流上下限约束：

$$P_l^{\min} \le H P_{\mathrm{bus},t} \le P_l^{\max} \tag{1-5}$$

H为功率传输分布因子，可由函数 makePTDF(case, refbus)得到。

2. 多时段经济调度模型 + 随机规划（Stochastic Programming，SP）

随机规划将已知的所有场景全部考虑，目标函数为所有场景成本的期望值，即：

$$\min \mathbb{E}\left[\sum_{t=1}^{T} \sum_{i=1}^{N} \left[\left(C_i^1 P_{g,i,t} + C_i^0 x_i + + C_{\mathrm{pv}} P_{\mathrm{pv},t} \right) \right] \right] \tag{1-6}$$

其中，通过分析所有场景的概率分布，得到目标函数期望值的表达式。

3. 样本平均近似（Sample Average Approximation, SAA）

样本平均近似是一种求解随机优化问题的方法，假设不知道场景的概率分布，对采样的 N 个场景求平均值处理，即用样本的平均值代替期望：

$$\min f_N(x) = \frac{1}{N} \sum_{i=1}^{N} F(x, \alpha_i) \tag{1-7}$$

在此式N为场景数，α_i为第 i 次采样得到的场景。在*优化基本理论与方法（16）随机方法之一*【1】中给出了证明，SAA 的最优值和最优解会在 N→∞时收敛到真实问题的最优值和最优解。

4. 机会约束规划（Chance Constrained Programing,CCP）

CCP 允许所作决策在一定程度上不满足约束条件，并根据随机变量的分布，使约束条件成立的概率不小于某一置信水平。

例如，对于约束（1-2）的概率形式：

$$\mathbb{P}\left(\sum_{i=1}^{N}\left(P_{g,i,t}+P_{\mathrm{pv},i,t}\right)\geq\sum_{i=1}^{N}D_{i,t}\right)\geq 1-\varepsilon$$

（1-8）表示，在考虑光伏不确定的情况下，系统总发电量能够满足总负荷需求的概率不低于置信水平 $1-\varepsilon$。在实际求解的过程中，需要将这种含有概率的随即约束转化为确定性约束。

【公式变形】

式（1-8）中只有$P_{\mathrm{pv},i,t}$是随机变量，于是将确定性项和随机项分开，转换为：

$$\mathbb{P}\left(\sum_{i=1}^{N}P_{\mathrm{pv},i,t}\geq\sum_{i=1}^{N}D_{i,t}-\sum_{i=1}^{N}P_{g,i,t}\right)\geq 1-\varepsilon$$

可以将确定项看作为阈值常数，式（1-9）表示，光伏出力的实际值大于阈值常数的概率不低于置信水平$1-\varepsilon$。

【引入概率分布假设】

假设t时刻的光伏总出力$\sum_{i=1}^{N}P_{\mathrm{pv},i,t}$服从正态分布，均值为$\mu_{\mathrm{pv},t}$，标准差为$\sigma_{\mathrm{pv},t}$，对不等式左侧进行标准化变换，有：

$$\mathbb{P}\left(\frac{\sum_{i=1}^{N}P_{\mathrm{pv},i,t}-\mu_{\mathrm{pv},t}}{\sigma_{\mathrm{pv},t}}\geq\frac{\left(\sum_{i=1}^{N}D_{i,t}-\sum_{i=1}^{N}P_{g,i,t}\right)-\mu_{\mathrm{pv},t}}{\sigma_{\mathrm{pv},t}}\right)\geq 1-\varepsilon$$

根据标准正态分布的对称性$\mathbb{P}(Z\geq z)=1-\Phi(z)$，要使概率$\mathbb{P}\geq 1-\varepsilon$，则不等式右边的项需要位于分布的左侧：

$$\frac{\left(\sum_{i=1}^{N}D_{i,t}-\sum_{i=1}^{N}P_{g,i,t}\right)-\mu_{\mathrm{pv},t}}{\sigma_{\mathrm{pv},t}}\leq\Phi^{-1}(\varepsilon)$$

【整理得到最终的线性约束】

整理不等式，得到最终的线性约束：

$$\sum_{i=1}^{N}P_{g,i,t}\geq\sum_{i=1}^{N}D_{i,t}-\left(\mu_{\mathrm{pv},t}-\Phi^{-1}(1-\varepsilon)\cdot\sigma_{\mathrm{pv},t}\right)$$

其中，$\mu_{\mathrm{pv},t}-\Phi^{-1}(1-\varepsilon)\cdot\sigma_{\mathrm{pv},t}$表示光伏的可靠出力，即为$\varepsilon\%$分位数。表示光伏有$(1-\varepsilon)\%$的概率出力大于$\left(\mu_{\mathrm{pv},t}-\Phi^{-1}(1-\varepsilon)\cdot\sigma_{\mathrm{pv},t}\right)$。

五、 代码编写

1. 选择前 10 组数据作为抽样数据，光伏的成本设置为 0，采用 SAA 方法求解

只列写与 ED 不同的部分，其余代码均与 ED 一致。

【定义决策变量】

```
%% Define variables
P_g = sdpvar(nb, nt,nsample);% 决策变量:10 个场景下发电机出力 (39,96,10)
P_pv = sdpvar(nb, nt,nsample);% 决策变量: 10 个场景下光伏出力 (39,96,10)
P_bus = P_g+P_pv-repmat(Pd, 1, 1, 10);% 10 个场景下节点注入功率 (39,96,10)
```

【目标函数】

```
%% Define Objective —— Stochastic Programming: SAA
%假定光伏出力成本为 0
Obj = 0;
for i = 1:nsample
    % 约束部分 ...
    % Obj
    Obj=Obj + (C1' * sum(P_g(:,:,i), 2)+sum(C0)*nt) * TS;
end
% 取平均值
Obj=Obj/nsample;
```

【约束方程】

```
%% Define Constraints: 10 场景
Constraints = [];

for i = 1:nsample
    % 出力约束
    Constraints = [Constraints, P_g(:,:,i) <= Pg_max, ...
                                P_g(:,:,i) >= Pg_min];
    % 爬坡约束
    Constraints = [Constraints, P_g(:,:,i) * U <= Ru, ...
                                P_g(:,:,i) * U >= Rd];
    % 线路潮流约束
    Constraints = [Constraints, H * P_bus(:,:,i) <= Pbranch_max, ...
                                H * P_bus(:,:,i) >= Pbranch_min];
    % 负荷平衡约束（直流潮流模型，30 节点接入光伏出力，ΣPbus=0）
    Constraints = [Constraints, sum(P_g(:,:,i), 1)+sum(P_pv(:,:,i),
1)==sum(Pd,1)];
    % 光伏出力约束
    Constraints = [Constraints, P_pv(:,:,i) <= Ppv_max(:,:,i)];
    Constraints = [Constraints, P_pv(:,:,i) >= 0];
```

```
    % Obj ...
end
```

该代码通过 for 循环，将 10 个场景的光伏出力约束全部加入约束方程中，优化结果是在 10 个采样场景下**保证 100%安全**的调度结果。

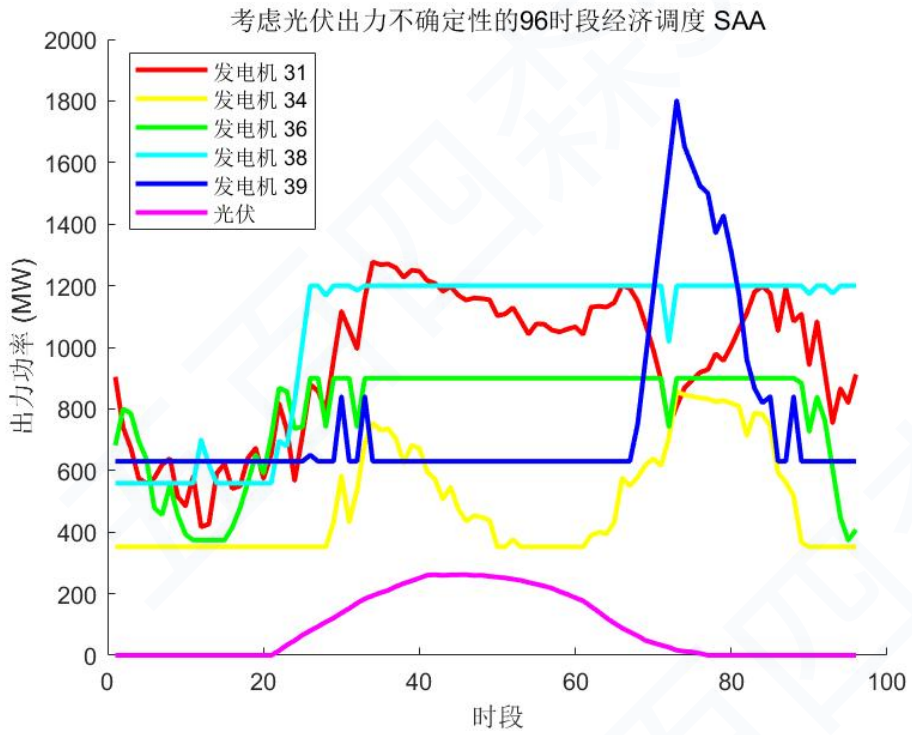
【优化求解】

```
%% Solve
options = sdpsettings('solver', 'gurobi', 'verbose', 1, 'debug', 1);
solution = optimize(Constraints, Obj, options);
```

【优化结果】

```
Solved in 2305 iterations and 0.57 seconds (0.96 work units)
Optimal objective  3.444814700e+05
求解时间：0.83032 秒
```

成本为 34449.15 $，各运行机组、光伏出力情况如图：



【对照无光伏 ED】

无光伏 ED 的优化结果成本为 34510.39 $，接入光伏后成本有下降，主要因为假设光伏出力无成本。

2. 采用 CCP 方法求解，$\varepsilon = 5\%$

【得到光伏出力数据 95%分位数】

假设光伏出力历史数据服从正态分布，估计光伏只有 5%的概率会超过其 95%的分位数，可以用于分析弃光风险的上界。

有两种得到 95%分位数的方法：

第一种：用 matlab 的 mean 和 std 函数得到光伏出力的均值$\mu_{\mathrm{pv},t}$和标准差$\sigma_{\mathrm{pv},t}$计算得到

```
%% 采用 norminv 计算
% 1. 计算均值 (Mean)
% dim = 3 表示沿着第 3 维(样本)求平均
mu_Ppv=mean(Ppvmax,3); %(1,96,1)

% 2. 计算标准差 (Standard Deviation)
% flag = 0 表示无偏估计(除以 N-1)
% dim = 3 表示沿着第 3 维计算
sigma_Ppv=std(Ppvmax,0,3);%(1,96,1)

% 95%分位数 计算光伏的"可靠出力" (Conservative PV Power)(1,96)
epsilon=0.05;
K_safe=norminv(1-epsilon,0,1); %安装或激活“Statistics and Machine Learning Toolbox”（统计与机器学习工具箱）
% K_safe = 1.6449;
PV_Conservative = mu_Ppv - K_safe * sigma_Ppv;
PV_Conservative = max(0, PV_Conservative); % 修正负数值
```

第二种：用 matlab 的 prctile 函数得到

```
%% 采用 prctile 计算
PV_Conservative=prctile(Ppvmax,95,3);
```

【约束方程】

基于式（1-12）修改负荷平衡约束，光伏出力约束也改为对应分位数

```
% 负荷平衡约束（直流潮流模型，30 节点接入光伏出力，ΣPbus=0）
Constraints = [Constraints, sum(P_g, 1)>=sum(Pd,1)-PV_Conservative];
% 光伏出力约束
Constraints = [Constraints, P_pv <= Ppvmax_ccp];
Constraints = [Constraints, P_pv >= 0];
```

【优化结果】

采用第一种分位数计算方法

```
Solved in 262 iterations and 0.06 seconds (0.08 work units)
Optimal objective  3.445014638e+05
求解时间：0.09904 秒
```

采用第二种分位数计算方法

```
Solved in 262 iterations and 0.07 seconds (0.08 work units)
Optimal objective  3.444636174e+05
求解时间：0.10677 秒
```

求解结果相近。