任务一

In [18]:
```
import numpy as np
import pandas as pd
import math
from scipy.optimize import linprog

# 定义时间段
hours = np.arange(1, 25)

# 定义天然气价格
C_天然气_升 = 3.28

# 定义电能价格
electric_price = [364 if 23 <= h or h <= 7 else 711 if h in [8, 9, 18, 19] else 1264 for h in hours]

#定义电网购电价格
C_电网_MW = 415.9

# 运行成本转换为元/MW
C_热电联产_MW = 0.0618 * 1000  # 热电联产的运行成本（元/MW）
C_光伏_MW = 0.55 * 1000       # 光伏的运行成本（元/MW）
C_电锅炉_MW = 0.0472 * 1000   # 电锅炉的运行成本（元/MW）

# 从文件读取的电负荷和热负荷数据
data_df = pd.read_excel('data.xlsx')

# 将电负荷和热负荷从 kW 转换为 MW
electric_load = data_df['电负荷'].values / 1000  # kW to MW
thermal_load = data_df['热负荷'].values / 1000   # kW to MW

# 初始化变量以存储最优解
optimal_x4 = 0
optimal_x5 = 1
min_cost = float('inf')
optimal_solution = None  # 用于存储最优解的具体值

# 枚举 x4 和 x5 的不同值
step = 0.001
for x4 in np.arange(0, 1, step):
    x5 = 1 - x4  # 通路占比约束

    # 目标函数
    c = []
    for t in range(24):
        C_电能_t = electric_price[t]
        c.extend([
            C_光伏_MW * 0.9645,  # X1 cost
            C_电能_t + C_电锅炉_MW * (1 - x4) * 0.95,  # X2 cost
            C_天然气_升 + C_热电联产_MW * ((1 - x5) * 0.05 * 0.3 * 36 / 3600 + x5 * 0.05 * 0.6 * 0.8 * 36 / 3600),  # X3 cost
            C_电网_MW # X6 cost
        ])

    # 约束条件
    A_eq = []
    b_eq = []
    A_ub = []
    b_ub = []

    for t in range(24):
        # 电负荷平衡
        row_electric = [0, 0, 0, 0] * t + [0.9645, x4, (1 - x5) * 0.05 * 0.3 * 36 / 3600, 1] + [0, 0, 0, 0] * (23 - t)
        A_eq.append(row_electric)
        b_eq.append(electric_load[t])

        # 热负荷平衡
        row_thermal = [0, 0, 0, 0] * t + [0, (1 - x4) * 0.95, x5 * 0.05 * 0.6 * 0.8 * 36 / 3600, 0] + [0, 0, 0, 0] * (23 - t)
        A_eq.append(row_thermal)
        b_eq.append(thermal_load[t])

        # 热电联产装机容量约束
        row_cogen = [0] * t * 4 + [0, 0, ((1 - x5) * 0.05 * 0.3 + x5 * 0.05 * 0.6 * 0.8) * 36 / 3600, 0] + [0] * (23 - t) * 4
        A_ub.append(row_cogen)
        b_ub.append(4.5)  # 4.5 MW

        # 光伏装机容量约束
        row_pv = [0] * t * 4 + [0.9645, 0, 0, 0] + [0] * (23 - t) * 4
        A_ub.append(row_pv)
        b_ub.append(0.8)  # 0.8 MW

        # 电锅炉装机容量约束
        row_boiler = [0] * t * 4 + [0, (1 - x4) * 0.95, 0, 0] + [0] * (23 - t) * 4
        A_ub.append(row_boiler)
        b_ub.append(2.4)  # 2.4 MW

        # 太阳能输入量约束
        if t < 6 or t > 17:
            row_solar = [0, 0, 0, 0] * t + [1, 0, 0, 0] + [0, 0, 0, 0] * (23 - t)
            A_eq.append(row_solar)
            b_eq.append(0)

    # 线性规划求解
    res = linprog(c, A_eq=A_eq, b_eq=b_eq, A_ub=A_ub, b_ub=b_ub, method='highs')

    # 检查枚举方案是否优于当前最低成本
    if res.success and res.fun < min_cost:
        min_cost = res.fun
        optimal_x4 = x4
        optimal_x5 = x5
        optimal_solution = res.x

# 输出结果
if optimal_solution is not None:
    print("Optimal solution found.")
    print("Objective function value: {:.4f} 元".format(min_cost))
    print(f"Optimal X4={optimal_x4:.4f}")
    print(f"Optimal X5={optimal_x5:.4f}")
    for t in range(24):
        idx = t * 4
        print(f"Hour {t+1}: X1={optimal_solution[idx]:.4f} MWh, X2={optimal_solution[idx+1]:.4f} MWh, X3={optimal_solution[idx+2]:.4f} m³,X6={optimal_solution[idx+3]:.4f} MWh")

        X1_actual_output = optimal_solution[idx]  # X1 总功率
        X2_actual_output = (1 - optimal_x4) * optimal_solution[idx + 1]  # X2 总功率
        # 计算所需设备数量
        number_of_solar_panels = math.ceil(X1_actual_output / 0.00026)  # 光伏板数目
        number_of_boilers = math.ceil(X2_actual_output / 0.4)  # 电锅炉数目
        print(f"Number of Solar Panels: {number_of_solar_panels}, Number of Boilers: {number_of_boilers}")
else:
    print("No optimal solution found. Reason:", res.message)
```

```
Optimal solution found.
Objective function value: 250065.9319 元
Optimal X4=0.0000
Optimal X5=1.0000
Hour 1: X1=0.0000 MWh, X2=2.2962 MWh, X3=0.0000 m³,X6=2.2152 MWh
Number of Solar Panels: 0, Number of Boilers: 6
Hour 2: X1=0.0000 MWh, X2=2.4265 MWh, X3=0.0000 m³,X6=2.2377 MWh
Number of Solar Panels: 0, Number of Boilers: 7
Hour 3: X1=0.0000 MWh, X2=2.5263 MWh, X3=2277.3092 m³,X6=2.2264 MWh
Number of Solar Panels: 0, Number of Boilers: 7
Hour 4: X1=0.0000 MWh, X2=2.5263 MWh, X3=4058.8367 m³,X6=2.7215 MWh
Number of Solar Panels: 0, Number of Boilers: 7
Hour 5: X1=0.0000 MWh, X2=2.5263 MWh, X3=4574.5433 m³,X6=3.4079 MWh
Number of Solar Panels: 0, Number of Boilers: 7
Hour 6: X1=0.0000 MWh, X2=2.5263 MWh, X3=4855.8375 m³,X6=5.6357 MWh
Number of Solar Panels: 0, Number of Boilers: 7
Hour 7: X1=0.0000 MWh, X2=2.5263 MWh, X3=5652.8363 m³,X6=6.0183 MWh
Number of Solar Panels: 0, Number of Boilers: 7
Hour 8: X1=0.0000 MWh, X2=2.5263 MWh, X3=6402.9529 m³,X6=4.1505 MWh
Number of Solar Panels: 0, Number of Boilers: 7
Hour 9: X1=0.0000 MWh, X2=2.5263 MWh, X3=7059.3058 m³,X6=7.8748 MWh
Number of Solar Panels: 0, Number of Boilers: 7
Hour 10: X1=0.0000 MWh, X2=2.5263 MWh, X3=2839.8967 m³,X6=8.4599 MWh
Number of Solar Panels: 0, Number of Boilers: 7
Hour 11: X1=0.0000 MWh, X2=2.5263 MWh, X3=1152.1333 m³,X6=8.7075 MWh
Number of Solar Panels: 0, Number of Boilers: 7
Hour 12: X1=0.0000 MWh, X2=2.5263 MWh, X3=402.0167 m³,X6=8.8312 MWh
Number of Solar Panels: 0, Number of Boilers: 7
Hour 13: X1=0.0000 MWh, X2=2.3318 MWh, X3=0.0000 m³,X6=8.5162 MWh
Number of Solar Panels: 0, Number of Boilers: 6
Hour 14: X1=0.0000 MWh, X2=2.0475 MWh, X3=0.0000 m³,X6=8.5274 MWh
Number of Solar Panels: 0, Number of Boilers: 6
Hour 15: X1=0.0000 MWh, X2=2.2489 MWh, X3=0.0000 m³,X6=8.7525 MWh
Number of Solar Panels: 0, Number of Boilers: 6
Hour 16: X1=0.0000 MWh, X2=2.3081 MWh, X3=0.0000 m³,X6=8.0323 MWh
Number of Solar Panels: 0, Number of Boilers: 6
Hour 17: X1=0.0000 MWh, X2=2.3792 MWh, X3=0.0000 m³,X6=5.7370 MWh
Number of Solar Panels: 0, Number of Boilers: 6
Hour 18: X1=0.0000 MWh, X2=2.4265 MWh, X3=0.0000 m³,X6=5.9170 MWh
Number of Solar Panels: 0, Number of Boilers: 7
Hour 19: X1=0.0000 MWh, X2=2.5094 MWh, X3=0.0000 m³,X6=6.7159 MWh
Number of Solar Panels: 0, Number of Boilers: 7
Hour 20: X1=0.0000 MWh, X2=2.5263 MWh, X3=120.7225 m³,X6=4.4205 MWh
Number of Solar Panels: 0, Number of Boilers: 7
Hour 21: X1=0.0000 MWh, X2=2.5263 MWh, X3=308.2508 m³,X6=3.9480 MWh
Number of Solar Panels: 0, Number of Boilers: 7
Hour 22: X1=0.0000 MWh, X2=2.5263 MWh, X3=777.0742 m³,X6=2.3615 MWh
Number of Solar Panels: 0, Number of Boilers: 7
Hour 23: X1=0.0000 MWh, X2=2.5263 MWh, X3=1386.5450 m³,X6=2.2602 MWh
Number of Solar Panels: 0, Number of Boilers: 7
Hour 24: X1=0.0000 MWh, X2=2.5263 MWh, X3=2042.8975 m³,X6=2.2264 MWh
Number of Solar Panels: 0, Number of Boilers: 7
```

拓展三- 集成太阳能发电量预测模型

In [10]:

```
import numpy as np
import pandas as pd
import math
from scipy.optimize import linprog

# 定义时间段
hours = np.arange(1, 25)

# 定义天然气价格
C_天然气_升 = 3.28

# 定义电能价格
electric_price = [364 if 23 <= h or h <= 7 else 711 if h in [8, 9, 18, 19] else 1264 for h in hours]

# 定义电网购电价格
C_电网_MW = 415.9

# 运行成本转换为元/MW
C_热电联产_MW = 0.0618 * 1000  # 热电联产的运行成本（元/MW）
C_光伏_MW = 0.55 * 1000       # 光伏的运行成本（元/MW）
C_电锅炉_MW = 0.0472 * 1000   # 电锅炉的运行成本（元/MW）

# 从文件读取的电负荷和热负荷数据
data_df = pd.read_excel('data.xlsx')

# 将电负荷和热负荷从 kW 转换为 MW
electric_load = data_df['电负荷'].values / 1000  # kW to MW
thermal_load = data_df['热负荷'].values / 1000   # kW to MW

# 从预测数据文件中读取太阳能发电量预测
forecast_data_df = pd.read_csv('combined_hourly_forecast_june_18_19.csv')

# 选择特定日期的数据
selected_date = input('Forecasted date (yyyy-mm-dd) :')  # 预测日期
forecast_data_df['ds'] = pd.to_datetime(forecast_data_df['ds'])
selected_forecast_data = forecast_data_df[
    (forecast_data_df['ds'].dt.date == pd.to_datetime(selected_date).date()) |
    (forecast_data_df['ds'].dt.date == (pd.to_datetime(selected_date) + pd.Timedelta(days=1)).date())
]

# 获取选定日期的太阳能发电量预测
solar_output_forecast = selected_forecast_data['yhat'].values / 1000  # kW to MW

# 初始化变量以存储最优解
optimal_x4 = 0
optimal_x5 = 1
min_cost = float('inf')
optimal_solution = None  # 用于存储最优解的具体值

# 枚举 x4 和 x5 的不同值
step = 0.001
for x4 in np.arange(0, 1, step):
    x5 = 1 - x4  # 通路占比约束

    # 目标函数
    c = []
    for t in range(24):
        C_电能_t = electric_price[t]
        c.extend([
            C_电能_t + C_电锅炉_MW * (1 - x4) * 0.95,  # X2 cost
            C_天然气_升 + C_热电联产_MW * ((1 - x5) * 0.05 * 0.3 * 36 / 3600 + x5 * 0.05 * 0.6 * 0.8 * 36 / 3600),  # X3 cost
            C_电网_MW # X6 cost
        ])

    # 约束条件
    A_eq = []
    b_eq = []
    A_ub = []
    b_ub = []

    for t in range(24):
        # 电负荷平衡
```

```
        adjusted_electric_load = max(electric_load[t] - solar_output_forecast[t], 0)
        row_electric = [0, 0, 0] * t + [x4, (1 - x5) * 0.05 * 0.3 * 36 / 3600, 1] + [0, 0, 0] * (23 - t)
        A_eq.append(row_electric)
        b_eq.append(adjusted_electric_load)

        # 热负荷平衡
        row_thermal = [0, 0, 0] * t + [(1 - x4) * 0.95, x5 * 0.05 * 0.6 * 0.8 * 36 / 3600, 0] + [0, 0, 0] * (23 - t)
        A_eq.append(row_thermal)
        b_eq.append(thermal_load[t])

        # 热电联产装机容量约束
        row_cogen = [0] * t * 3 + [0, ((1 - x5) * 0.05 * 0.3 + x5 * 0.05 * 0.6 * 0.8) * 36 / 3600, 0] + [0] * (23 - t) * 3
        A_ub.append(row_cogen)
        b_ub.append(4.5)  # 4.5 MW

        # 电锅炉装机容量约束
        row_boiler = [0] * t * 3 + [(1 - x4) * 0.95, 0, 0] + [0] * (23 - t) * 3
        A_ub.append(row_boiler)
        b_ub.append(2.4)  # 2.4 MW

    # 线性规划求解
    res = linprog(c, A_eq=A_eq, b_eq=b_eq, A_ub=A_ub, b_ub=b_ub, method='highs')

    # 检查枚举方案是否优于当前最低成本
    if res.success and res.fun < min_cost:
        min_cost = res.fun
        optimal_x4 = x4
        optimal_x5 = x5
        optimal_solution = res.x

# 计算X1以外的成本
if optimal_solution is not None:
    print("Optimal solution found.")
    print("Objective function value: {:.4f} 元".format(min_cost))
    print(f"Optimal X4={optimal_x4:.4f}")
    print(f"Optimal X5={optimal_x5:.4f}")
    for t in range(24):
        idx = t * 3
        X1_output = solar_output_forecast[t]
        X2_actual_output = (1 - optimal_x4) * optimal_solution[idx]
        wasted_solar = max(X1_output - electric_load[t], 0)  # 计算浪费的太阳能
        print(f"Hour {t+1}: X1={X1_output:.4f} MWh, X2={optimal_solution[idx]:.4f} MWh, X3={optimal_solution[idx+1]:.4f} m³, X6={optimal_solution[idx+2]:.4f} MWh")
        # 计算所需设备数量
        number_of_boilers = math.ceil(X2_actual_output / 0.4)
        print(f"Number of Boilers: {number_of_boilers}, Waste of solar energy: {wasted_solar:.4f} MWh")
else:
    print("No optimal solution found. Reason:", res.message)
```

```
Forecasted date (yyyy-mm-dd) :2020-6-19
Optimal solution found.
Objective function value: 212542.5100 元
Optimal X4=0.0000
Optimal X5=1.0000
Hour 1: X1=0.0000 MWh, X2=2.2962 MWh, X3=0.0000 m³, X6=2.2152 MWh
Number of Boilers: 6, Waste of solar energy: 0.0000 MWh
Hour 2: X1=0.0000 MWh, X2=2.4265 MWh, X3=0.0000 m³, X6=2.2377 MWh
Number of Boilers: 7, Waste of solar energy: 0.0000 MWh
Hour 3: X1=0.0000 MWh, X2=2.5263 MWh, X3=2277.3092 m³, X6=2.2264 MWh
Number of Boilers: 7, Waste of solar energy: 0.0000 MWh
Hour 4: X1=0.0000 MWh, X2=2.5263 MWh, X3=4058.8367 m³, X6=2.7215 MWh
Number of Boilers: 7, Waste of solar energy: 0.0000 MWh
Hour 5: X1=0.0000 MWh, X2=2.5263 MWh, X3=4574.5433 m³, X6=3.4079 MWh
Number of Boilers: 7, Waste of solar energy: 0.0000 MWh
Hour 6: X1=0.0000 MWh, X2=2.5263 MWh, X3=4855.8375 m³, X6=5.6357 MWh
Number of Boilers: 7, Waste of solar energy: 0.0000 MWh
Hour 7: X1=0.0000 MWh, X2=2.5263 MWh, X3=5652.8363 m³, X6=6.0183 MWh
Number of Boilers: 7, Waste of solar energy: 0.0000 MWh
Hour 8: X1=6.8442 MWh, X2=2.5263 MWh, X3=6402.9529 m³, X6=0.0000 MWh
Number of Boilers: 7, Waste of solar energy: 2.6937 MWh
Hour 9: X1=46.3293 MWh, X2=2.5263 MWh, X3=7059.3058 m³, X6=0.0000 MWh
Number of Boilers: 7, Waste of solar energy: 38.4545 MWh
Hour 10: X1=98.6995 MWh, X2=2.5263 MWh, X3=2839.8967 m³, X6=0.0000 MWh
Number of Boilers: 7, Waste of solar energy: 90.2396 MWh
Hour 11: X1=148.8467 MWh, X2=2.5263 MWh, X3=1152.1333 m³, X6=0.0000 MWh
Number of Boilers: 7, Waste of solar energy: 140.1392 MWh
Hour 12: X1=184.4411 MWh, X2=2.5263 MWh, X3=402.0167 m³, X6=0.0000 MWh
Number of Boilers: 7, Waste of solar energy: 175.6099 MWh
Hour 13: X1=201.7916 MWh, X2=2.3318 MWh, X3=0.0000 m³, X6=0.0000 MWh
Number of Boilers: 6, Waste of solar energy: 193.2754 MWh
Hour 14: X1=203.5960 MWh, X2=2.0475 MWh, X3=0.0000 m³, X6=0.0000 MWh
Number of Boilers: 6, Waste of solar energy: 195.0685 MWh
Hour 15: X1=192.3909 MWh, X2=2.2489 MWh, X3=0.0000 m³, X6=0.0000 MWh
Number of Boilers: 6, Waste of solar energy: 183.6385 MWh
Hour 16: X1=167.2688 MWh, X2=2.3081 MWh, X3=0.0000 m³, X6=0.0000 MWh
Number of Boilers: 6, Waste of solar energy: 159.2365 MWh
Hour 17: X1=127.2934 MWh, X2=2.3792 MWh, X3=0.0000 m³, X6=0.0000 MWh
Number of Boilers: 6, Waste of solar energy: 121.5564 MWh
Hour 18: X1=77.4585 MWh, X2=2.4265 MWh, X3=0.0000 m³, X6=0.0000 MWh
Number of Boilers: 7, Waste of solar energy: 71.5415 MWh
Hour 19: X1=29.8706 MWh, X2=2.5094 MWh, X3=0.0000 m³, X6=0.0000 MWh
Number of Boilers: 7, Waste of solar energy: 23.1547 MWh
Hour 20: X1=0.0000 MWh, X2=2.5263 MWh, X3=120.7225 m³, X6=4.4205 MWh
Number of Boilers: 7, Waste of solar energy: 0.0000 MWh
Hour 21: X1=0.0000 MWh, X2=2.5263 MWh, X3=308.2508 m³, X6=3.9480 MWh
Number of Boilers: 7, Waste of solar energy: 0.0000 MWh
Hour 22: X1=0.0000 MWh, X2=2.5263 MWh, X3=777.0742 m³, X6=2.3615 MWh
Number of Boilers: 7, Waste of solar energy: 0.0000 MWh
Hour 23: X1=0.0000 MWh, X2=2.5263 MWh, X3=1386.5450 m³, X6=2.2602 MWh
Number of Boilers: 7, Waste of solar energy: 0.0000 MWh
Hour 24: X1=0.0000 MWh, X2=2.5263 MWh, X3=2042.8975 m³, X6=2.2264 MWh
Number of Boilers: 7, Waste of solar energy: 0.0000 MWh
```