模型一

一、数学模型

1. 决策变量

- `xij`: 0-1变量，表示是否选择边(i,j)
- `ui`: 连续变量，表示城市i在路径中的位置(1到n-1)

2. 目标函数

```
min ∑∑(cij * xij)  其中i,j∈V, i≠j
```

其中cij表示城市i到j的距离

3. 约束条件

1. **度数约束**

```
∑xij = 1  ∀i∈V  (出度)
∑xji = 1  ∀i∈V  (入度)
```

2. **MTZ子回路消除约束**

```
ui - uj + n*xij ≤ n-1  ∀i,j∈V\{1}, i≠j
```

3. **变量取值范围**

```
xij ∈ {0,1}  ∀i,j∈V, i≠j
1 ≤ ui ≤ n-1  ∀i∈V\{1}
```

二、模型特点分析

1. MTZ约束原理

- MTZ约束通过引入辅助变量ui建立城市访问顺序
- 当xij=1时，保证ui < uj
- 防止形成不包含起点的子回路

2. 优点

- 变量数量适中：$O(n^2)$个x变量和O(n)个u变量
- 约束数量合理：$O(n^2)$个约束
- 模型线性，易于求解
- 内存占用相对较小

3. 缺点

- LP松弛解较差
- 对称性强
- 在大规模问题上收敛慢

三、实现策略

1. 基本框架

1. 构建目标函数和基本变量
2. 添加度数约束
3. 添加MTZ约束
4. 设置求解参数

2. 改进措施

1. **增强约束**

```
// 加强MTZ约束
ui ≥ 1             ∀i∈V\{1}
ui ≤ n-1           ∀i∈V\{1}
ui - uj ≥ 1 - n*(1-xij)  ∀i,j∈V\{1}, i≠j
```

2. **打破对称性**

```
// 固定起点
u[1] = 0
```

3. **初始解注入**

- 使用蚁群算法获取好的初始解
- 通过warm start机制注入

四、求解策略

1. 参数设置

```
model->set(GRB_DoubleParam_TimeLimit, 600);    // 时间限制
model->set(GRB_DoubleParam_MIPGap, 0.01);     // 优化间隔
model->set(GRB_IntParam_Threads, 0);          // 线程数
```

2. 混合策略

1. 蚁群算法获取初始解
2. MTZ模型精确求解
3. 设置时间限制保证求解效率

代码：

```
#include "gurobi_c++.h"
#include <cassert>
#include <cmath>
#include <random>
#include <algorithm>
#include <vector>
```

```cpp
#include <iostream>
#include <fstream>
#include <sstream>
#include <chrono>
#include <omp.h>
using namespace std;

// 算法参数
const double ALPHA = 1.0;     // 信息素重要程度
const double BETA = 2.0;      // 启发式因子重要程度
const double RHO = 0.1;       // 信息素蒸发系数
const double Q = 100;         // 信息素增加强度
const int MAX_ITER = 100;     // 最大迭代次数
const int ANT_NUM = 50;       // 蚂蚁数量


string itos(int i) {stringstream s; s << i; return s.str(); }

// 蚁群算法类
class AntColony {
private:
    int n;
    vector<vector<double>> distance;
    vector<vector<double>> pheromone;
    vector<int> bestTour;
    double bestLength;
    mt19937 gen;

    double calculateDistance(const pair<double,double>& a, const
pair<double,double>& b) {
        return sqrt(pow(a.first - b.first, 2) + pow(a.second -
b.second, 2));
    }

    vector<int> constructSolution() {
        vector<bool> visited(n, false);
        vector<int> tour;
        int current = uniform_int_distribution<>(0, n-1)(gen);

        tour.push_back(current);
        visited[current] = true;
```

```
        while (tour.size() < n) {
            vector<double> prob;
            double total = 0;

            // 计算概率
            for (int next = 0; next < n; next++) {
                if (!visited[next]) {
                    double p = pow(pheromone[current][next], ALPHA)
*
                               pow(1.0/distance[current][next],
BETA);
                    prob.push_back(p);
                    total += p;
                } else {
                    prob.push_back(0);
                }
            }

            // 轮盘赌选择
            double r = uniform_real_distribution<>(0, total)(gen);
            double sum = 0;
            int next = -1;

            for (int i = 0; i < n && next == -1; i++) {
                if (!visited[i]) {
                    sum += prob[i];
                    if (sum >= r) {
                        next = i;
                    }
                }
            }

            if (next == -1) {
                for (int i = 0; i < n; i++) {
                    if (!visited[i]) {
                        next = i;
                        break;
                    }
                }
            }

            tour.push_back(next);
```

```
            visited[next] = true;
            current = next;
        }

        return tour;
    }

    double calculateTourLength(const vector<int>& tour) {
        double length = 0;
        for (size_t i = 0; i < tour.size(); i++) {
            int from = tour[i];
            int to = tour[(i + 1) % tour.size()];
            length += distance[from][to];
        }
        return length;
    }

public:
    AntColony(const vector<pair<double,double>>& coords) :
        gen(chrono::steady_clock::now().time_since_epoch().count())
{
        n = coords.size();
        distance.resize(n, vector<double>(n));
        pheromone.resize(n, vector<double>(n, 1.0));
        bestLength = numeric_limits<double>::max();

        for (int i = 0; i < n; i++) {
            for (int j = 0; j < n; j++) {
                distance[i][j] = calculateDistance(coords[i],
coords[j]);
            }
        }
    }

    vector<int> solve() {
        cout << "\n============ 蚁群算法优化开始 ============" <<
endl;

        for (int iter = 0; iter < MAX_ITER; iter++) {
            vector<vector<int>> antPaths(ANT_NUM);
            vector<double> pathLengths(ANT_NUM);
```

```
            #pragma omp parallel for
            for (int k = 0; k < ANT_NUM; k++) {
                antPaths[k] = constructSolution();
                pathLengths[k] = calculateTourLength(antPaths[k]);

                #pragma omp critical
                {
                    if (pathLengths[k] < bestLength) {
                        bestLength = pathLengths[k];
                        bestTour = antPaths[k];
                        cout << "迭代 " << iter << ": 新的最优解 = "
<< bestLength << endl;
                    }
                }
            }

            // 更新信息素
            for (int i = 0; i < n; i++) {
                for (int j = 0; j < n; j++) {
                    pheromone[i][j] *= (1.0 - RHO);
                }
            }

            for (int k = 0; k < ANT_NUM; k++) {
                double delta = Q / pathLengths[k];
                for (size_t i = 0; i < antPaths[k].size(); i++) {
                    int from = antPaths[k][i];
                    int to = antPaths[k][(i + 1) %
antPaths[k].size()];
                    pheromone[from][to] += delta;
                    pheromone[to][from] += delta;
                }
            }
        }

        cout << "蚁群算法最终最优解长度: " << bestLength << endl;
        return bestTour;
    }

    double getBestLength() const { return bestLength; }
};
```

```
    // MTZ约束求解器类
    class MTZSolver {
    private:
        GRBEnv* env;
        GRBModel* model;
        GRBVar** x;  // 边变量
        GRBVar* u;   // MTZ辅助变量
        int n;
        vector<pair<double,double>> coords;

    public:
        MTZSolver(const vector<pair<double,double>>& coordinates,
const vector<int>& initialTour) {
            coords = coordinates;
            n = coords.size();

            try {
                env = new GRBEnv();
                model = new GRBModel(*env);

                // 创建变量
                x = new GRBVar*[n];
                for (int i = 0; i < n; i++) {
                    x[i] = new GRBVar[n];
                }

                u = new GRBVar[n];

                // 添加变量和目标函数
                GRBLinExpr obj = 0;
                for (int i = 0; i < n; i++) {
                    for (int j = 0; j < n; j++) {
                        if (i != j) {
                            double dist = sqrt(pow(coords[i].first
- coords[j].first, 2) +
                                            pow(coords[i].second -
coords[j].second, 2));
                            x[i][j] = model->addVar(0.0, 1.0, dist,
GRB_BINARY);
                            obj += dist * x[i][j];
                        }
                    }
```

```
                    u[i] = model->addVar(0.0, n-1, 0.0,
GRB_CONTINUOUS);
                }

                model->setObjective(obj, GRB_MINIMIZE);

                // 添加约束
                // 1. 每个城市进出度为1
                for (int i = 0; i < n; i++) {
                    GRBLinExpr in = 0, out = 0;
                    for (int j = 0; j < n; j++) {
                        if (i != j) {
                            in += x[j][i];
                            out += x[i][j];
                        }
                    }
                    model->addConstr(in == 1);
                    model->addConstr(out == 1);
                }

                // 2. MTZ约束
                for (int i = 1; i < n; i++) {
                    for (int j = 1; j < n; j++) {
                        if (i != j) {
                            model->addConstr(u[i] - u[j] + n * x[i]
[j] <= n - 1);
                        }
                    }
                }

                // 设置求解参数
                model->set(GRB_DoubleParam_TimeLimit, 600);
                model->set(GRB_DoubleParam_MIPGap, 0.01);
                model->set(GRB_IntParam_Threads, 0);

                // 设置初始解
                for (size_t i = 0; i < initialTour.size() - 1; i++)
{
                    x[initialTour[i]]
[initialTour[i+1]].set(GRB_DoubleAttr_Start, 1.0);
                }
```

```
                x[initialTour.back()]
[initialTour.front()].set(GRB_DoubleAttr_Start, 1.0);

            } catch (GRBException& e) {
                cout << "Error code = " << e.getErrorCode() <<
endl;
                cout << e.getMessage() << endl;
            }
        }

        ~MTZSolver() {
            for (int i = 0; i < n; i++) {
                delete[] x[i];
            }
            delete[] x;
            delete[] u;
            delete model;
            delete env;
        }

        vector<int> solve() {
            vector<int> tour;
            try {
                cout << "\n============ MTZ优化开始 ============" <<
endl;
                model->optimize();

                if (model->get(GRB_IntAttr_Status) == GRB_OPTIMAL)
{
                    // 重建路径
                    vector<bool> visited(n, false);
                    int current = 0;
                    tour.push_back(current);
                    visited[current] = true;

                    while (tour.size() < n) {
                        for (int j = 0; j < n; j++) {
                            if (!visited[j] && x[current]
[j].get(GRB_DoubleAttr_X) > 0.5) {
                                tour.push_back(j);
                                visited[j] = true;
                                current = j;
```

```
                                        break;
                                    }
                                }
                            }
                        }
                    } catch (GRBException& e) {
                        cout << "Error code = " << e.getErrorCode() <<
endl;
                        cout << e.getMessage() << endl;
                    }
                    return tour;
                }

                double getObjectiveValue() {
                    return model->get(GRB_DoubleAttr_ObjVal);
                }
            };



int main(int argc, char* argv[]) {
    if (argc < 2) {
        cout << "用法: " << argv[0] << " <tsp文件路径>" << endl;
        return 1;
    }

    string tsp_file = argv[1];
    ifstream infile(tsp_file);
    if (!infile.is_open()) {
        cout << "错误: 无法打开文件 " << tsp_file << endl;
        return 1;
    }

    vector<pair<double, double>> coords;
    string line;
    while (getline(infile, line)) {
        if (line == "NODE_COORD_SECTION") break;
    }

    while (getline(infile, line)) {
        if (line == "EOF") break;
        stringstream ss(line);
```

```
        int index;
        double x, y;
        ss >> index >> x >> y;
        coords.emplace_back(x, y);
    }
    infile.close();

    int n = coords.size();
    cout << "问题规模: " << n << " 个城市" << endl;

    try {
        // 第一阶段: 蚁群算法
        cout << "\n============ 第一阶段: 蚁群算法 ============" << 
endl;
        AntColony aco(coords);
        vector<int> aco_tour = aco.solve();
        double aco_length = aco.getBestLength();

        // 选择更好的解作为初始解
        vector<int> initial_tour =  aco_tour ;
        double initial_length = aco_length;

        // 第三阶段: MTZ精确求解
        cout << "\n============ 第三阶段: MTZ精确求解 ============" << 
endl;
        MTZSolver mtz(coords, initial_tour);
        vector<int> final_tour = mtz.solve();
        double final_length = mtz.getObjectiveValue();

        // 输出总结果
        cout << "\n============ 优化结果总结 ============" << endl;
        cout << "蚁群算法解长度: " << aco_length << endl;
        cout << "选择的初始解长度: " << initial_length << endl;
        cout << "MTZ最优解长度: " << final_length << endl;
        cout << "最终改进比例: " << (initial_length - final_length) / 
initial_length * 100 << "%" << endl;

        cout << "\n最优路径: ";
        for (size_t i = 0; i < final_tour.size(); i++) {
            cout << final_tour[i] << " ";
            if ((i + 1) % 20 == 0) cout << endl;
        }
```

```
        cout << endl;

    } catch (GRBException& e) {
        cout << "Gurobi错误 " << e.getErrorCode() << ": " <<
e.getMessage() << endl;
    } catch (const exception& e) {
        cout << "标准错误: " << e.what() << endl;
    } catch (...) {
        cout << "未知错误" << endl;
    }

    return 0;
}
```

算例

rat575

```
问题规模: 575 个城市

============ 第一阶段: 蚁群算法 ============

============ 蚁群算法优化开始 ============
迭代 0: 新的最优解 = 35059.9
迭代 0: 新的最优解 = 33792.4
迭代 0: 新的最优解 = 33324.2
迭代 0: 新的最优解 = 32719.7
迭代 0: 新的最优解 = 32650.6
迭代 0: 新的最优解 = 29519.6
迭代 5: 新的最优解 = 29498.7
迭代 9: 新的最优解 = 28964.1
迭代 9: 新的最优解 = 28607.5
迭代 9: 新的最优解 = 28583.3
迭代 10: 新的最优解 = 27682.2
迭代 12: 新的最优解 = 26919.3
迭代 13: 新的最优解 = 26551.5
迭代 15: 新的最优解 = 26151
迭代 15: 新的最优解 = 26037.4
迭代 16: 新的最优解 = 25255.5
迭代 17: 新的最优解 = 24783.4
```

```
迭代 17: 新的最优解 = 23498.1
迭代 19: 新的最优解 = 23048.8
迭代 20: 新的最优解 = 22101.6
迭代 21: 新的最优解 = 21191.3
迭代 23: 新的最优解 = 20464.4
迭代 23: 新的最优解 = 20285.9
迭代 23: 新的最优解 = 20254
迭代 24: 新的最优解 = 20233.7
迭代 24: 新的最优解 = 19173.9
迭代 26: 新的最优解 = 18428.2
迭代 27: 新的最优解 = 18074
迭代 27: 新的最优解 = 18020.3
迭代 28: 新的最优解 = 17162.2
迭代 28: 新的最优解 = 16939.6
迭代 29: 新的最优解 = 16815.3
迭代 29: 新的最优解 = 16757.9
迭代 30: 新的最优解 = 15634.9
迭代 30: 新的最优解 = 15571.3
迭代 31: 新的最优解 = 15534.7
迭代 32: 新的最优解 = 15249.5
迭代 33: 新的最优解 = 14929.4
迭代 33: 新的最优解 = 14786.4
迭代 33: 新的最优解 = 14772.8
迭代 33: 新的最优解 = 14747.5
迭代 33: 新的最优解 = 14229.5
迭代 35: 新的最优解 = 13855.3
迭代 36: 新的最优解 = 13502.6
迭代 36: 新的最优解 = 13365.3
迭代 37: 新的最优解 = 12794.4
迭代 37: 新的最优解 = 12540.8
迭代 39: 新的最优解 = 12258.8
迭代 39: 新的最优解 = 12164.1
迭代 40: 新的最优解 = 12158.6
迭代 40: 新的最优解 = 12055.6
迭代 41: 新的最优解 = 11972.2
迭代 42: 新的最优解 = 11794.9
迭代 43: 新的最优解 = 11664.6
迭代 43: 新的最优解 = 11537.3
迭代 43: 新的最优解 = 11469.4
迭代 44: 新的最优解 = 11081
迭代 45: 新的最优解 = 10668
迭代 51: 新的最优解 = 10569.8
```

```
迭代 52: 新的最优解 = 10445.6
迭代 52: 新的最优解 = 10382.8
迭代 52: 新的最优解 = 10148.6
迭代 53: 新的最优解 = 10075.7
迭代 54: 新的最优解 = 9878.89
迭代 54: 新的最优解 = 9724.32
迭代 60: 新的最优解 = 9574.41
迭代 62: 新的最优解 = 9525.7
迭代 66: 新的最优解 = 9310.9
迭代 69: 新的最优解 = 9218.98
迭代 73: 新的最优解 = 9165.81
迭代 82: 新的最优解 = 9136.87
迭代 84: 新的最优解 = 9034.91
迭代 93: 新的最优解 = 8933.69
蚁群算法最终最优解长度: 8933.69

============ 第三阶段: MTZ精确求解 ============
Set parameter Username
Set parameter LicenseID to value 2642819
Academic license - for non-commercial use only - expires 2026-03-27
Set parameter TimeLimit to value 600
Set parameter MIPGap to value 0.01
Set parameter Threads to value 0

============ MTZ优化开始 ============
Gurobi Optimizer version 12.0.1 build v12.0.1rc0 (linux64 - "Ubuntu
22.04.5 LTS")

CPU model: AMD Ryzen 9 5900HX with Radeon Graphics, instruction set
[SSE2|AVX|AVX2]
Thread count: 8 physical cores, 16 logical processors, using up to
16 threads

Non-default parameters:
TimeLimit  600
MIPGap  0.01

Optimize a model with 330052 rows, 330625 columns and 1646806
nonzeros
Model fingerprint: 0x3b8f1d9a
Variable types: 575 continuous, 330050 integer (330050 binary)
Coefficient statistics:
```

```
  Matrix range     [1e+00, 6e+02]
  Objective range  [2e+00, 5e+02]
  Bounds range     [1e+00, 6e+02]
  RHS range        [1e+00, 6e+02]

Warning: Completing partial solution with 329475 unfixed non-
continuous variables out of 330050
User MIP start produced solution with objective 8933.69 (0.31s)
Loaded user MIP start with objective 8933.69

Presolve removed 0 rows and 1 columns
Presolve time: 2.12s
Presolved: 330052 rows, 330624 columns, 1646806 nonzeros
Variable types: 574 continuous, 330050 integer (330050 binary)
Deterministic concurrent LP optimizer: primal simplex, dual
simplex, and barrier
Showing barrier log only...

Root barrier log...

Ordering time: 0.01s

Barrier statistics:
 AA' NZ     : 1.650e+05
 Factor NZ  : 1.656e+05 (roughly 70 MB of memory)
 Factor Ops : 6.354e+07 (less than 1 second per iteration)
 Threads    : 6

                  Objective                Residual
Iter       Primal          Dual         Primal    Dual     Compl
  Time
   0   2.84013400e+07 -1.48828401e+08  2.49e+02 2.27e-13  9.29e+02
    4s
   1   6.93293118e+05 -1.42825777e+07  7.90e+00 1.02e-12  5.22e+01
    5s

Barrier performed 1 iterations in 4.52 seconds (6.32 work units)
Barrier solve interrupted - model solved by another algorithm

Concurrent spin time: 0.05s

Solved with dual simplex
```

```
Use crossover to convert LP symmetric solution to basic solution...

Root simplex log...

Iteration    Objective       Primal Inf.    Dual Inf.      Time
    2588    6.0223838e+03   0.000000e+00   2.577151e+01      5s
    2710    6.0223838e+03   0.000000e+00   0.000000e+00      5s

Root relaxation: objective 6.022384e+03, 2710 iterations, 1.80
seconds (1.76 work units)

    Nodes    |    Current Node    |     Objective Bounds      |
Work
 Expl Unexpl |  Obj  Depth IntInf | Incumbent    BestBd   Gap |
It/Node Time


     0     0 6022.38380    0  995 8933.69185 6022.38380  32.6%
-    6s
H    0     0                    8746.4697772 6022.38380  31.1%
-    8s
H    0     0                    8596.4610168 6022.38380  29.9%
-   11s
     0     0 6208.95679    0 1074 8596.46102 6208.95679  27.8%
-   23s
H    0     0                    8444.4888409 6208.95679  26.5%
-   29s
H    0     0                    8430.9250500 6208.95679  26.4%
-   35s
H    0     0                    8430.4191060 6208.95679  26.4%
-   37s
H    0     0                    8427.0629228 6208.95679  26.3%
-   39s
H    0     0                    8427.0137927 6208.95679  26.3%
-   41s
     0     0 6208.95908    0 1064 8427.01379 6208.95908  26.3%
-   43s
     0     0 6391.36322    0 1090 8427.01379 6391.36322  24.2%
-   50s
     0     0 6392.62050    0 1110 8427.01379 6392.62050  24.1%
-   52s
```

```
      0      0 6392.62050    0 1106 8427.01379 6392.62050  24.1%   -   52s
      0      0 6619.07156    0 1138 8427.01379 6619.07156  21.5%   -   60s
      0      0 6623.43063    0 1145 8427.01379 6623.43063  21.4%   -   61s
      0      0 6623.43063    0 1144 8427.01379 6623.43063  21.4%   -   62s
      0      0 6666.97173    0 1134 8427.01379 6666.97173  20.9%   -   68s
      0      0 6667.45586    0 1135 8427.01379 6667.45586  20.9%   -   70s
      0      0 6667.45586    0 1132 8427.01379 6667.45586  20.9%   -   75s
      0      0 6667.45586    0 1132 8427.01379 6667.45586  20.9%   -   76s
      0      0 6667.45586    0 1051 8427.01379 6667.45586  20.9%   -   82s
      0      0 6667.45586    0 1063 8427.01379 6667.45586  20.9%   -   83s
      0      0 6668.17175    0 1058 8427.01379 6668.17175  20.9%   -   88s
      0      0 6670.87213    0 1061 8427.01379 6670.87213  20.8%   -   89s
      0      0 6670.87213    0 1061 8427.01379 6670.87213  20.8%   -   89s
      0      0 6670.87213    0 1064 8427.01379 6670.87213  20.8%   -   94s
      0      0 6670.87213    0 1064 8427.01379 6670.87213  20.8%   -   95s
      0      0 6671.06429    0 1064 8427.01379 6671.06429  20.8%   -  100s
      0      0 6671.06429    0 1064 8427.01379 6671.06429  20.8%   -  101s
      0      0 6671.06565    0 1070 8427.01379 6671.06565  20.8%   -  107s
      0      0 6671.06565    0 1070 8427.01379 6671.06565  20.8%   -  108s
      0      0 6672.82335    0 1051 8427.01379 6672.82335  20.8%   -  113s
      0      0 6673.04253    0 1056 8427.01379 6673.04253  20.8%   -  114s
```

```
     0     0 6673.04253    0 1054 8427.01379 6673.04253  20.8%
-  120s
     0     0 6673.04253    0 1054 8427.01379 6673.04253  20.8%
-  121s
     0     0 6673.04253    0  845 8427.01379 6673.04253  20.8%
-  128s
     0     0 6673.04253    0  853 8427.01379 6673.04253  20.8%
-  129s
     0     0 6673.04253    0  868 8427.01379 6673.04253  20.8%
-  134s
     0     0 6673.04253    0  868 8427.01379 6673.04253  20.8%
-  135s
     0     0 6673.04253    0  868 8427.01379 6673.04253  20.8%
-  140s
     0     0 6673.04253    0  775 8427.01379 6673.04253  20.8%
-  142s
     0     2 6707.81603    0  775 8427.01379 6707.81603  20.4%
-  163s
     7    16 6707.81603    3  639 8427.01379 6707.81603  20.4%
240  165s
    23    33 6707.81603    5  941 8427.01379 6707.81603  20.4%
183  170s
    41    54 6707.81603    6  942 8427.01379 6707.81603  20.4%
177  176s
    66    82 6707.81603    8  994 8427.01379 6707.81603  20.4%
170  182s
    81    95 6707.81603    9  927 8427.01379 6707.81603  20.4%
173  185s
   118   132 6707.81603   11  879 8427.01379 6707.81603  20.4%
156  192s
   145   162 6707.81603   13  785 8427.01379 6707.81603  20.4%
148  197s
   174   183 6707.81603   15  935 8427.01379 6707.81603  20.4%
145  202s
H  182   191                    8380.7129603 6707.81603  20.0%
143  209s
H  184   191                    8124.2823067 6707.81603  17.4%
144  209s
H  185   191                    7978.8638155 6707.81603  15.9%
145  209s
   190   204 6707.81603   16  932 7978.86382 6707.81603  15.9%
142  211s
```

```
    219   235 6707.81603   18  920 7978.86382 6707.81603  15.9%
138  217s
    234   251 6707.81603   21  945 7978.86382 6707.81603  15.9%
139  220s
H  251   259                     7978.3668712 6707.81603  15.9%
137  224s
H  254   259                     7976.5167405 6707.81603  15.9%
137  224s
H  257   259                     7956.8045641 6707.81603  15.7%
137  224s
    258   277 6707.81603   24  923 7956.80456 6707.81603  15.7%
137  227s
    276   296 6707.81603   25  720 7956.80456 6707.81603  15.7%
136  231s
    295   313 6707.81603   27  746 7956.80456 6707.81603  15.7%
135  235s
    335   357 6707.81603   29  794 7956.80456 6707.81603  15.7%
132  243s
    356   376 6707.81603   30  688 7956.80456 6707.81603  15.7%
131  247s
    375   392 6707.81603   31  785 7956.80456 6707.81603  15.7%
130  250s
    391   409 6707.81603   32  693 7956.80456 6707.81603  15.7%
132  255s
    408   426 6707.81603   33  712 7956.80456 6707.81603  15.7%
133  260s
H  426   434                     7921.4254701 6707.81603  15.3%
133  264s
    433   458 6707.81603   35  698 7921.42547 6707.81603  15.3%
134  269s
    457   476 6707.81603   36  730 7921.42547 6707.81603  15.3%
133  274s
    475   500 6707.81603   38  673 7921.42547 6707.81603  15.3%
134  279s
    499   535 6707.81603   40  644 7921.42547 6707.81603  15.3%
134  286s
    534   564 6707.81603   43  706 7921.42547 6707.81603  15.3%
132  292s
    563   591 6707.81603   46  616 7921.42547 6707.81603  15.3%
131  298s
    590   607 6707.81603   49  750 7921.42547 6707.81603  15.3%
131  304s
```

```
H  591   607                     7906.9380630 6707.81603  15.2%
130  304s
   606   638 6707.81603   50   707 7906.93806 6707.81603  15.2%
130  310s
   637   686 6707.81603   53   701 7906.93806 6707.81603  15.2%
129  318s
H  687   694                     7859.0084070 6707.81603  14.6%
125  325s
H  692   694                     7808.4311402 6707.81603  14.1%
125  325s
H  693   694                     7726.7532111 6707.81603  13.2%
124  325s
   695   735 6707.81603   59   714 7726.75321 6707.81603  13.2%
124  333s
   736   786 6707.81603   65   716 7726.75321 6707.81603  13.2%
122  341s
   787   832 6707.81603   72   874 7726.75321 6707.81603  13.2%
120  350s
   833   882 6707.81603   79   855 7726.75321 6707.81603  13.2%
118  358s
   883   943 6707.81603   87   818 7726.75321 6707.81603  13.2%
115  367s
   944   967 6707.81603   94   761 7726.75321 6707.81603  13.2%
111  373s
   968   975 6707.81603   97   760 7726.75321 6707.81603  13.2%
109  379s
   976   986 6707.81603   98   775 7726.75321 6707.81603  13.2%
108  385s
   987  1022 6707.81603  102   734 7726.75321 6707.81603  13.2%
107  392s
H 1023  1096                     7718.1835812 6707.81603  13.1%
105  402s
H 1088  1096                     7710.5646079 6707.81603  13.0%
102  402s
  1097  1185 6707.81603  120   857 7710.56461 6707.81603  13.0%
101  412s
  1188  1250 6707.81603  137   751 7710.56461 6707.81603  13.0%
95.9  423s
  1259  1306 6707.81603  149   755 7710.56461 6707.81603  13.0%
93.1  435s
H 1318  1338                     7680.6631398 6707.81603  12.7%
91.3  445s
```

```
H 1327  1338                        7645.8934153 6707.81603  12.3%
91.2  445s
  1350  1401 6716.91733  164  743 7645.89342 6707.81603  12.3%
90.3  457s
  1416  1515 6717.40382  177  781 7645.89342 6707.81603  12.3%
88.5  470s
H 1532  1522                        7616.8068835 6707.81603  11.9%
84.2  485s
H 1533  1522                        7602.0314501 6707.81603  11.8%
84.1  485s
H 1536  1522                        7403.1594058 6707.81603  9.39%
84.0  485s
  1540  1648 6718.34833  191  783 7403.15941 6707.81603  9.39%
83.9  498s
  1670  1794 6718.69408  203  761 7403.15941 6707.81603  9.39%
79.7  513s
  1818  1895 6719.06198  217  776 7403.15941 6707.81603  9.39%
75.5  527s
  1920  2020 6719.45225  229  623 7403.15941 6707.81603  9.39%
73.6  541s
  2045  2179 6719.71955  238  798 7403.15941 6707.81603  9.39%
71.0  556s
H 2204  2308                        7399.0066816 6707.81603  9.34%
67.7  572s
  2334  2473 6720.39947  260  775 7399.00668 6707.81603  9.34%
65.7  588s
H 2503  2480                        7371.6554641 6707.81603  9.01%
62.9  598s
H 2504  2480                        7311.2652357 6707.81603  8.25%
62.9  598s
  2511  2492 6720.82459  270  769 7311.26524 6707.81603  8.25%
62.9  600s

Cutting planes:
  Learned: 56
  Gomory: 38
  Cover: 2
  Implied bound: 160
  MIR: 164
  RLT: 115
  Relax-and-lift: 71
  BQP: 8
```

```
  PSD: 5

Explored 2523 nodes (172441 simplex iterations) in 600.06 seconds
(640.68 work units)
Thread count was 16 (of 16 available processors)

Solution count 10: 7311.27 7371.66 7399.01 ... 7718.18

Time limit reached
Best objective 7.311265235701e+03, best bound 6.707816025654e+03,
gap 8.2537%

============ 优化结果总结 ============
蚁群算法解长度: 8933.69
选择的初始解长度: 8933.69
MTZ最优解长度: 7311.27
最终改进比例: 18.1608%
```

u727

```
问题规模: 724 个城市

============ 第一阶段: 蚁群算法 ============

============ 蚁群算法优化开始 ============
迭代 0: 新的最优解 = 225421
迭代 0: 新的最优解 = 218969
迭代 0: 新的最优解 = 218616
迭代 0: 新的最优解 = 215072
迭代 0: 新的最优解 = 214870
迭代 0: 新的最优解 = 208563
迭代 5: 新的最优解 = 206502
迭代 6: 新的最优解 = 205066
迭代 6: 新的最优解 = 205005
迭代 10: 新的最优解 = 201259
迭代 12: 新的最优解 = 199894
迭代 15: 新的最优解 = 197800
迭代 18: 新的最优解 = 195728
迭代 19: 新的最优解 = 195023
```

```
迭代 20: 新的最优解 = 193819
迭代 21: 新的最优解 = 187930
迭代 22: 新的最优解 = 186557
迭代 25: 新的最优解 = 185798
迭代 25: 新的最优解 = 183746
迭代 26: 新的最优解 = 183718
迭代 27: 新的最优解 = 182220
迭代 27: 新的最优解 = 180634
迭代 27: 新的最优解 = 174948
迭代 29: 新的最优解 = 174748
迭代 30: 新的最优解 = 173320
迭代 30: 新的最优解 = 170598
迭代 31: 新的最优解 = 170430
迭代 32: 新的最优解 = 165084
迭代 32: 新的最优解 = 165042
迭代 33: 新的最优解 = 164082
迭代 33: 新的最优解 = 156648
迭代 35: 新的最优解 = 149591
迭代 36: 新的最优解 = 148991
迭代 36: 新的最优解 = 145641
迭代 38: 新的最优解 = 145263
迭代 39: 新的最优解 = 135940
迭代 40: 新的最优解 = 132483
迭代 42: 新的最优解 = 127940
迭代 42: 新的最优解 = 125528
迭代 43: 新的最优解 = 123387
迭代 43: 新的最优解 = 119991
迭代 44: 新的最优解 = 118880
迭代 45: 新的最优解 = 118819
迭代 45: 新的最优解 = 118797
迭代 45: 新的最优解 = 113931
迭代 45: 新的最优解 = 113649
迭代 46: 新的最优解 = 108533
迭代 46: 新的最优解 = 104991
迭代 48: 新的最优解 = 104260
迭代 49: 新的最优解 = 103003
迭代 49: 新的最优解 = 102481
迭代 49: 新的最优解 = 101883
迭代 50: 新的最优解 = 97025.2
迭代 50: 新的最优解 = 94172.5
迭代 51: 新的最优解 = 93936.5
迭代 51: 新的最优解 = 89830.6
```

```
迭代 53: 新的最优解 = 89257.2
迭代 53: 新的最优解 = 88289.7
迭代 54: 新的最优解 = 88225.3
迭代 55: 新的最优解 = 85947.6
迭代 55: 新的最优解 = 83842
迭代 56: 新的最优解 = 81632.1
迭代 56: 新的最优解 = 81412.8
迭代 58: 新的最优解 = 75174.2
迭代 61: 新的最优解 = 75019.6
迭代 61: 新的最优解 = 73828.3
迭代 62: 新的最优解 = 72828.4
迭代 62: 新的最优解 = 69597.1
迭代 65: 新的最优解 = 66660.6
迭代 66: 新的最优解 = 65490.4
迭代 72: 新的最优解 = 64990.4
迭代 73: 新的最优解 = 64341.3
迭代 74: 新的最优解 = 64129.3
迭代 77: 新的最优解 = 63981.6
迭代 77: 新的最优解 = 63484.6
迭代 78: 新的最优解 = 63346.9
迭代 78: 新的最优解 = 63176.3
迭代 78: 新的最优解 = 62344.3
迭代 78: 新的最优解 = 60076.1
迭代 84: 新的最优解 = 60050.2
迭代 85: 新的最优解 = 59751.7
迭代 86: 新的最优解 = 58697.3
迭代 96: 新的最优解 = 58021.6
蚁群算法最终最优解长度: 58021.6

============ 第三阶段: MTZ精确求解 ============
Set parameter Username
Set parameter LicenseID to value 2642819
Academic license - for non-commercial use only - expires 2026-03-27
Set parameter TimeLimit to value 600
Set parameter MIPGap to value 0.01
Set parameter Threads to value 0

============ MTZ优化开始 ============
Gurobi Optimizer version 12.0.1 build v12.0.1rc0 (linux64 - "Ubuntu
22.04.5 LTS")
```

```
CPU model: AMD Ryzen 9 5900HX with Radeon Graphics, instruction set
[SSE2|AVX|AVX2]
Thread count: 8 physical cores, 16 logical processors, using up to
16 threads

Non-default parameters:
TimeLimit  600
MIPGap  0.01

Optimize a model with 523454 rows, 524176 columns and 2612922
nonzeros
Model fingerprint: 0xaf2289a6
Variable types: 724 continuous, 523452 integer (523452 binary)
Coefficient statistics:
  Matrix range     [1e+00, 7e+02]
  Objective range  [3e+00, 3e+03]
  Bounds range     [1e+00, 7e+02]
  RHS range        [1e+00, 7e+02]

Warning: Completing partial solution with 522728 unfixed non-
continuous variables out of 523452
User MIP start produced solution with objective 58021.6 (0.50s)
Loaded user MIP start with objective 58021.6

Presolve removed 0 rows and 1 columns
Presolve time: 3.81s
Presolved: 523454 rows, 524175 columns, 2612922 nonzeros
Variable types: 723 continuous, 523452 integer (523452 binary)
Deterministic concurrent LP optimizer: primal simplex, dual
simplex, and barrier
Showing barrier log only...

Root barrier log...

Ordering time: 0.01s

Barrier statistics:
 AA' NZ     : 2.617e+05
 Factor NZ  : 2.624e+05 (roughly 100 MB of memory)
 Factor Ops : 1.268e+08 (less than 1 second per iteration)
 Threads    : 6
```

```
                   Objective                Residual
Iter       Primal          Dual         Primal    Dual     Compl
  Time
   0   2.71799669e+08 -1.34378345e+09  3.10e+02 9.09e-13  5.30e+03
    8s
   1   6.55718288e+06 -1.26786813e+08  9.93e+00 6.37e-12  2.94e+02
    8s

Barrier performed 1 iterations in 7.96 seconds (10.16 work units)
Barrier solve interrupted - model solved by another algorithm

Concurrent spin time: 0.09s

Solved with dual simplex

Root simplex log...

Iteration    Objective       Primal Inf.    Dual Inf.      Time
    1161    3.5831126e+04   0.000000e+00   0.000000e+00      8s

Use crossover to convert LP symmetric solution to basic solution...

Root crossover log...

       0 DPushes remaining with DInf 0.0000000e+00
9s

     556 PPushes remaining with PInf 0.0000000e+00
9s
       0 PPushes remaining with PInf 0.0000000e+00
9s

  Push phase complete: Pinf 0.0000000e+00, Dinf 1.9846333e+02
9s



Root simplex log...

Iteration    Objective       Primal Inf.    Dual Inf.      Time
    3154    3.5831126e+04   0.000000e+00   1.984633e+02      9s
    3256    3.5831126e+04   0.000000e+00   0.000000e+00      9s
```

```
Root relaxation: objective 3.583113e+04, 3256 iterations, 3.13
seconds (2.62 work units)
Total elapsed time = 10.29s (DegenMoves)

    Nodes    |    Current Node    |     Objective Bounds      |
Work
 Expl Unexpl |  Obj  Depth IntInf | Incumbent    BestBd   Gap |
It/Node Time

     0     0 35831.1265    0 1324 58021.5576 35831.1265  38.2%
-   13s
H    0     0                    57035.202340 35831.1265  37.2%
-   17s
H    0     0                    56589.035633 35831.1265  36.7%
-   21s
H    0     0                    56576.628765 35831.1265  36.7%
-   31s
     0     0 37795.7946    0 1365 56576.6288 37795.7946  33.2%
-   50s
H    0     0                    56538.233316 37822.2357  33.1%
-   56s
H    0     0                    55006.645031 37822.2357  31.2%
-   61s
H    0     0                    54977.733272 37822.2357  31.2%
-   78s
H    0     0                    53584.182052 37822.2357  29.4%
-   78s
H    0     0                    52864.339546 37822.2357  28.5%
-   78s
H    0     0                    52810.103130 37822.2357  28.4%
-   81s
H    0     0                    52539.325808 37822.2357  28.0%
-   85s
H    0     0                    52085.956984 37822.2357  27.4%
-   85s
H    0     0                    52076.369201 37822.2357  27.4%
-   91s
H    0     0                    52052.577423 37822.2357  27.3%
-   91s
     0     0 37822.2357    0 1369 52052.5774 37822.2357  27.3%
-   92s
```

```
     0     0 37822.2357    0 1369 52052.5774 37822.2357  27.3%
-   93s
     0     0 38329.4050    0 1332 52052.5774 38329.4050  26.4%
-  107s
     0     0 38336.4968    0 1342 52052.5774 38336.4968  26.4%
-  113s
     0     0 38336.4968    0 1342 52052.5774 38336.4968  26.4%
-  113s
     0     0 40050.4148    0 1411 52052.5774 40050.4148  23.1%
-  131s
     0     0 40059.3046    0 1409 52052.5774 40059.3046  23.0%
-  133s
     0     0 40059.3046    0 1409 52052.5774 40059.3046  23.0%
-  134s
     0     0 40428.2748    0 1390 52052.5774 40428.2748  22.3%
-  148s
     0     0 40432.1649    0 1382 52052.5774 40432.1649  22.3%
-  151s
     0     0 40432.1649    0 1382 52052.5774 40432.1649  22.3%
-  151s
     0     0 40440.7590    0 1397 52052.5774 40440.7590  22.3%
-  162s
     0     0 40440.7791    0 1399 52052.5774 40440.7791  22.3%
-  165s
     0     0 40440.7814    0 1398 52052.5774 40440.7814  22.3%
-  175s
     0     0 40440.7814    0 1398 52052.5774 40440.7814  22.3%
-  177s
     0     0 40459.3954    0 1366 52052.5774 40459.3954  22.3%
-  188s
     0     0 40459.3954    0 1366 52052.5774 40459.3954  22.3%
-  191s
     0     0 40459.4057    0 1395 52052.5774 40459.4057  22.3%
-  201s
     0     0 40459.4584    0 1396 52052.5774 40459.4584  22.3%
-  204s
     0     0 40466.8225    0 1397 52052.5774 40466.8225  22.3%
-  215s
     0     0 40466.8225    0 1397 52052.5774 40466.8225  22.3%
-  217s
     0     0 40471.8316    0 1404 52052.5774 40471.8316  22.2%
-  228s
```

```
     0     0 40481.0646    0 1404 52052.5774 40481.0646  22.2%
-  230s
     0     0 40481.0646    0 1404 52052.5774 40481.0646  22.2%
-  241s
     0     0 40481.0646    0 1404 52052.5774 40481.0646  22.2%
-  244s
     0     0 40481.0646    0 1247 52052.5774 40481.0646  22.2%
-  259s
     0     0 40481.0646    0 1254 52052.5774 40481.0646  22.2%
-  262s
     0     0 40481.0646    0 1254 52052.5774 40481.0646  22.2%
-  276s
     0     0 40481.0646    0 1253 52052.5774 40481.0646  22.2%
-  278s
     0     0 40481.0646    0 1250 52052.5774 40481.0646  22.2%
-  291s
     0     0 40481.0646    0 1244 52052.5774 40481.0646  22.2%
-  296s
     0     2 40786.5740    0 1244 52052.5774 40786.5740  21.6%
-  335s
     7    16 40786.5740    3 1396 52052.5774 40786.5740  21.6%
318  342s
    15    24 40786.5740    4 1366 52052.5774 40786.5740  21.6%
221  345s
    23    37 40786.5740    5 1312 52052.5774 40786.5740  21.6%
191  350s
    36    48 40786.5740    6 1318 52052.5774 40786.5740  21.6%
169  355s
    64    84 40786.5740    8 1270 52052.5774 40786.5740  21.6%
147  362s
    83   100 40786.5740    9 1257 52052.5774 40786.5740  21.6%
128  365s
   114   130 40786.5740   11 1261 52052.5774 40786.5740  21.6%
106  371s
   129   146 40786.5740   13 1274 52052.5774 40786.5740  21.6%
102  375s
   161   175 40786.5740   15 1273 52052.5774 40786.5740  21.6%
99.0  383s
   174   185 40786.5740   16 1271 52052.5774 40786.5740  21.6%
104  387s
   184   195 40786.5740   16 1287 52052.5774 40786.5740  21.6%
106  390s
```

```
   194   209 40786.5740   17 1279 52052.5774 40786.5740  21.6%
107  396s
   208   220 40786.5740   18 1177 52052.5774 40786.5740  21.6%
111  401s
   219   231 40786.5740   18 1226 52052.5774 40786.5740  21.6%
115  405s
   230   244 40786.5740   19 1271 52052.5774 40786.5740  21.6%
116  410s
   256   270 40786.5740   23 1197 52052.5774 40786.5740  21.6%
116  420s
   269   283 40786.5740   24 1180 52052.5774 40786.5740  21.6%
119  426s
   282   298 40786.5740   24 1157 52052.5774 40786.5740  21.6%
121  432s
H  297   306                      51328.658026 40786.5740  20.5%
121  438s
   305   324 40786.5740   26 1172 51328.6580 40786.5740  20.5%
122  444s
   323   342 40786.5740   27 1167 51328.6580 40786.5740  20.5%
121  450s
   341   361 40786.5740   28 1166 51328.6580 40786.5740  20.5%
121  455s
   360   383 40786.5740   30 1157 51328.6580 40786.5740  20.5%
119  462s
   382   406 40786.5740   31 1114 51328.6580 40786.5740  20.5%
117  470s
   405   427 40786.5740   33 1197 51328.6580 40786.5740  20.5%
115  477s
H  426   435                      51271.405195 40786.5740  20.4%
114  486s
H  430   435                      51140.882621 40786.5740  20.2%
115  486s
H  431   435                      51110.572871 40786.5740  20.2%
115  486s
   434   451 40786.5740   35 1121 51110.5729 40786.5740  20.2%
115  497s
   450   481 40786.5740   36 1204 51110.5729 40786.5740  20.2%
116  506s
   480   504 40786.5740   37 1211 51110.5729 40786.5740  20.2%
114  515s
   503   535 40786.5740   40 1187 51110.5729 40786.5740  20.2%
112  525s
```

```
   534   558 40786.5740   42 1193 51110.5729 40786.5740  20.2%
110  535s
H  557   566                   50462.142578 40786.5740  19.2%
110  548s
   565   609 40786.5740   44 1182 50462.1426 40786.5740  19.2%
112  559s
   610   638 40786.5740   48 1178 50462.1426 40786.5740  19.2%
107  570s
H  639   646                   50381.286608 40786.5740  19.0%
107  581s
   647   692 40786.5740   50 1150 50381.2866 40786.5740  19.0%
109  593s
   693   719 40786.5740   55 1168 50381.2866 40786.5740  19.0%
106  600s

Cutting planes:
  Learned: 74
  Gomory: 42
  Lift-and-project: 2
  Cover: 3
  Implied bound: 305
  MIR: 238
  RLT: 95
  Relax-and-lift: 93
  PSD: 31

Explored 720 nodes (91286 simplex iterations) in 600.15 seconds
(545.10 work units)
Thread count was 16 (of 16 available processors)

Solution count 10: 50381.3 50462.1 51110.6 ... 52539.3

Time limit reached
Best objective 5.038128660777e+04, best bound 4.078657399896e+04,
gap 19.0442%

============ 优化结果总结 ============
蚁群算法解长度: 58021.6
选择的初始解长度: 58021.6
MTZ最优解长度: 50381.3
最终改进比例: 13.168%
```

u1060

```
问题规模: 1060 个城市

============ 第一阶段: 蚁群算法 ============

============ 蚁群算法优化开始 ============
迭代 0: 新的最优解 = 1.46181e+06
迭代 0: 新的最优解 = 1.4152e+06
迭代 0: 新的最优解 = 1.3881e+06
迭代 0: 新的最优解 = 1.3427e+06
迭代 0: 新的最优解 = 1.28639e+06
迭代 0: 新的最优解 = 1.23e+06
迭代 1: 新的最优解 = 1.18536e+06
迭代 17: 新的最优解 = 1.17849e+06
迭代 21: 新的最优解 = 1.14776e+06
迭代 32: 新的最优解 = 1.1332e+06
迭代 34: 新的最优解 = 1.13205e+06
迭代 37: 新的最优解 = 1.08756e+06
迭代 41: 新的最优解 = 1.0478e+06
迭代 45: 新的最优解 = 1.02819e+06
迭代 46: 新的最优解 = 1.02526e+06
迭代 46: 新的最优解 = 993831
迭代 48: 新的最优解 = 987781
迭代 49: 新的最优解 = 866826
迭代 54: 新的最优解 = 846088
迭代 54: 新的最优解 = 843396
迭代 54: 新的最优解 = 798596
迭代 56: 新的最优解 = 785957
迭代 57: 新的最优解 = 766072
迭代 57: 新的最优解 = 758018
迭代 57: 新的最优解 = 748316
迭代 59: 新的最优解 = 725420
迭代 60: 新的最优解 = 696462
迭代 61: 新的最优解 = 689344
迭代 62: 新的最优解 = 676412
迭代 63: 新的最优解 = 664174
迭代 63: 新的最优解 = 653109
迭代 63: 新的最优解 = 625091
迭代 65: 新的最优解 = 619409
迭代 65: 新的最优解 = 617694
迭代 66: 新的最优解 = 613488
```

```
迭代 66: 新的最优解 = 612966
迭代 66: 新的最优解 = 594694
迭代 67: 新的最优解 = 592380
迭代 67: 新的最优解 = 583055
迭代 67: 新的最优解 = 581679
迭代 67: 新的最优解 = 573546
迭代 68: 新的最优解 = 569501
迭代 68: 新的最优解 = 566935
迭代 68: 新的最优解 = 555889
迭代 68: 新的最优解 = 553469
迭代 69: 新的最优解 = 537128
迭代 70: 新的最优解 = 535803
迭代 71: 新的最优解 = 520275
迭代 71: 新的最优解 = 518413
迭代 71: 新的最优解 = 501847
迭代 72: 新的最优解 = 500135
迭代 72: 新的最优解 = 487680
迭代 72: 新的最优解 = 483807
迭代 75: 新的最优解 = 460460
迭代 76: 新的最优解 = 447026
迭代 78: 新的最优解 = 428226
迭代 79: 新的最优解 = 418355
迭代 80: 新的最优解 = 397452
迭代 83: 新的最优解 = 394994
迭代 83: 新的最优解 = 394459
迭代 86: 新的最优解 = 387208
迭代 86: 新的最优解 = 385918
迭代 87: 新的最优解 = 383062
迭代 90: 新的最优解 = 381651
迭代 90: 新的最优解 = 369762
迭代 92: 新的最优解 = 362831
迭代 95: 新的最优解 = 360549
迭代 97: 新的最优解 = 355690
迭代 97: 新的最优解 = 354813
迭代 97: 新的最优解 = 354797
迭代 98: 新的最优解 = 340344
蚁群算法最终最优解长度: 340344

============ 第三阶段: MTZ精确求解 ============
Set parameter Username
Set parameter LicenseID to value 2642819
Academic license - for non-commercial use only - expires 2026-03-27
```

```
Set parameter TimeLimit to value 600
Set parameter MIPGap to value 0.01
Set parameter Threads to value 0

============ MTZ优化开始 ============
Gurobi Optimizer version 12.0.1 build v12.0.1rc0 (linux64 - "Ubuntu
22.04.5 LTS")

CPU model: AMD Ryzen 9 5900HX with Radeon Graphics, instruction set
[SSE2|AVX|AVX2]
Thread count: 8 physical cores, 16 logical processors, using up to
16 threads

Non-default parameters:
TimeLimit  600
MIPGap  0.01

Optimize a model with 1122542 rows, 1123600 columns and 5606346
nonzeros
Model fingerprint: 0x225ff487
Variable types: 1060 continuous, 1122540 integer (1122540 binary)
Coefficient statistics:
  Matrix range     [1e+00, 1e+03]
  Objective range  [7e+01, 2e+04]
  Bounds range     [1e+00, 1e+03]
  RHS range        [1e+00, 1e+03]

Warning: Completing partial solution with 1121480 unfixed non-
continuous variables out of 1122540
User MIP start produced solution with objective 340344 (1.10s)
Loaded user MIP start with objective 340344
Processed MIP start in 1.13 seconds (1.32 work units)

Presolve removed 0 rows and 1 columns (presolve time = 5s)...
Presolve removed 0 rows and 1 columns
Presolve time: 9.25s
Presolved: 1122542 rows, 1123599 columns, 5606346 nonzeros
Variable types: 1059 continuous, 1122540 integer (1122540 binary)
Deterministic concurrent LP optimizer: primal simplex, dual
simplex, and barrier
Showing barrier log only...
```

```
Root barrier log...

Ordering time: 0.02s

Barrier statistics:
 AA' NZ     : 5.613e+05
 Factor NZ  : 5.623e+05 (roughly 230 MB of memory)
 Factor Ops : 3.976e+08 (less than 1 second per iteration)
 Threads    : 6

                  Objective                Residual
Iter       Primal          Dual         Primal    Dual     Compl
  Time
   0   3.18661027e+09 -2.01698506e+10  4.70e+02 7.28e-12  3.68e+04
   19s
   1   6.92712489e+07 -1.97903001e+09  1.43e+01 4.00e-11  2.09e+03
   19s
   2   4.17468466e+06 -1.83444630e+07  4.49e-14 3.64e-11  2.01e+01
   19s

Barrier performed 2 iterations in 19.16 seconds (22.44 work units)
Barrier solve interrupted - model solved by another algorithm

Concurrent spin time: 0.16s (can be avoided by choosing Method=3)

Solved with dual simplex

Root simplex log...

Iteration    Objective       Primal Inf.    Dual Inf.      Time
    4931    1.8333898e+05   0.000000e+00   0.000000e+00     19s

Use crossover to convert LP symmetric solution to basic solution...

Root crossover log...

       0 DPushes remaining with DInf 0.0000000e+00
21s

     771 PPushes remaining with PInf 0.0000000e+00
21s
```

```
       0 PPushes remaining with PInf 0.0000000e+00
21s

  Push phase complete: Pinf 0.0000000e+00, Dinf 6.1669184e+03
21s



Root simplex log...

Iteration    Objective       Primal Inf.    Dual Inf.      Time
    7813    1.8333898e+05   0.000000e+00   6.166918e+03     21s
    8038    1.8333898e+05   0.000000e+00   0.000000e+00     22s

Root relaxation: objective 1.833390e+05, 8038 iterations, 7.42
seconds (6.85 work units)
Total elapsed time = 25.81s (DegenMoves)

    Nodes    |    Current Node    |     Objective Bounds      |
Work
 Expl Unexpl |  Obj  Depth IntInf | Incumbent    BestBd   Gap |
It/Node Time


     0     0 183338.980    0 1780 340344.451 183338.980  46.1%
-   28s
H    0     0                    336344.25437 183338.980  45.5%
-   38s
H    0     0                    332934.60191 183338.980  44.9%
-   46s
H    0     0                    329747.08540 183338.980  44.4%
-   66s
     0     0 198118.093    0 2183 329747.085 198118.093  39.9%
-   89s
H    0     0                    327111.49033 198118.093  39.4%
-  101s
H    0     0                    326953.10611 198118.093  39.4%
-  105s
H    0     0                    326885.79167 198118.093  39.4%
-  105s
H    0     0                    326776.49680 198118.093  39.4%
-  105s
H    0     0                    326496.33451 198118.093  39.3%
-  110s
```

```
     0     0 198133.486    0 2162 326496.335 198133.486  39.3%   -  116s
     0     0 198133.486    0 2166 326496.335 198133.486  39.3%   -  117s
     0     0 201354.546    0 2198 326496.335 201354.546  38.3%   -  148s
     0     0 201354.546    0 2200 326496.335 201354.546  38.3%   -  152s
     0     0 208496.172    0 2081 326496.335 208496.172  36.1%   -  185s
     0     0 208496.691    0 2085 326496.335 208496.691  36.1%   -  189s
     0     0 208802.350    0 2026 326496.335 208802.350  36.0%   -  220s
     0     0 208802.350    0 2026 326496.335 208802.350  36.0%   -  224s
     0     0 208802.350    0 2026 326496.335 208802.350  36.0%   -  248s
     0     0 208821.753    0 2026 326496.335 208821.753  36.0%   -  253s
     0     0 208821.753    0 1870 326496.335 208821.753  36.0%   -  289s
     0     0 208821.753    0 1879 326496.335 208821.753  36.0%   -  308s
     0     0 208821.753    0 1878 326496.335 208821.753  36.0%   -  335s
H    0     0                    326201.01859 208821.753  36.0%   -  344s
H    0     0                    322830.23422 208821.753  35.3%   -  352s
H    0     0                    321558.61248 208821.753  35.1%   -  382s
H    0     0                    290927.60186 208821.753  28.2%   -  387s
H    0     0                    290906.22088 208821.753  28.2%   -  391s
H    0     0                    290578.70625 208821.753  28.1%   -  391s
H    0     0                    290478.76626 208821.753  28.1%   -  398s
     0     0 208821.753    0 1878 290478.766 208821.753  28.1%   -  407s
```

```
     0     0 208821.753    0 1878 290478.766 208821.753  28.1%
-  434s
     0     0 208821.753    0 1878 290478.766 208821.753  28.1%
-  439s
     0     0 208821.753    0 1858 290478.766 208821.753  28.1%
-  475s
     0     0 208821.753    0 1858 290478.766 208821.753  28.1%
-  483s
     0     2 210406.565    0 1858 290478.766 210406.565  27.6%
-  555s
     3     8 210406.565    2 1874 290478.766 210406.565  27.6%
619  560s
     7    16 210406.565    3 1880 290478.766 210406.565  27.6%
369  565s
    23    40 210406.565    5 1872 290478.766 210406.565  27.6%
152  572s
    39    53 210406.565    6 1869 290478.766 210406.565  27.6%
105  576s
    52    67 210406.565    7 1817 290478.766 210406.565  27.6%
85.3  581s
    66    82 210406.565    8 1818 290478.766 210406.565  27.6%
74.2  585s
    97   111 210406.565   10 1816 290478.766 210406.565  27.6%
56.1  594s
   110   123 210406.565   12 1822 290478.766 210406.565  27.6%
54.4  599s

Cutting planes:
  Learned: 104
  Gomory: 16
  Implied bound: 432
  Clique: 1
  MIR: 265
  RLT: 185
  Relax-and-lift: 72
  BQP: 1
  PSD: 31

Explored 122 nodes (30408 simplex iterations) in 601.05 seconds
(539.50 work units)
Thread count was 16 (of 16 available processors)
```

```
Solution count 10: 290479 290579 290906 ... 326886

Time limit reached
Best objective 2.904787662550e+05, best bound 2.104065645758e+05,
gap 27.5656%

============ 优化结果总结 ============
蚁群算法解长度: 340344
选择的初始解长度: 340344
MTZ最优解长度: 290479
最终改进比例: 14.6515%
```

模型二

一、数学模型

1. 决策变量

- `xij`: 0-1变量，表示是否选择边(i,j)

2. 目标函数

```
min ∑∑(cij * xij)  其中i,j∈V, i≠j
```

其中cij表示城市i到j的距离

3. 约束条件

1. **度数约束**:

```
∑xij = 2  ∀i∈V  (每个顶点的度数为2)
```

2. **子回路消除约束**（通过回调函数动态添加）：

```
∑∑xij ≤ |S|-1  ∀S⊂V, S≠∅
```

其中S是顶点集合的任意真子集

二、回调函数建模特点

1. 基本原理

- 不预先添加所有子回路消除约束
- 在求解过程中动态检测和添加违反的约束
- 使用延迟约束(Lazy Constraints)机制

2. 优点

- 内存效率高：仅添加必要的约束
- 求解速度快：约束数量大幅减少
- 可处理大规模问题

3. 缺点

- 需要实现回调逻辑
- 约束生成的计算开销
- 理论收敛性较差

三、实现策略

1. 主要组件

1. **回调类实现**

```
class subtourelim: public GRBCallback {
    protected:
        void callback() {
            if (where == GRB_CB_MIPSOL) {
                // 检测子回路
                // 添加违反的约束
            }
        }
};
```

2. **子回路检测**

```
void findsubtour(int n, double** sol, int* tourlenP, int* tour) {
    // 使用深度优先搜索找子回路
    // 返回最小子回路
}
```

2. 求解流程

1. 构建基本模型（仅含度数约束）
2. 设置回调函数
3. 启动求解过程
4. 动态添加子回路消除约束

代码:

```
#include "gurobi_c++.h"
#include <cassert>
#include <cmath>
#include <random>
#include <algorithm>
#include <vector>
#include <iostream>
#include <fstream>
#include <sstream>
#include <chrono>
#include <omp.h>
using namespace std;
```

```cpp
// 算法参数
const double ALPHA = 1.0;     // 信息素重要程度
const double BETA = 2.0;      // 启发式因子重要程度
const double RHO = 0.1;       // 信息素蒸发系数
const double Q = 100;         // 信息素增加强度
const int MAX_ITER = 100;     // 最大迭代次数
const int ANT_NUM = 50;       // 蚂蚁数量


string itos(int i) {stringstream s; s << i; return s.str(); }

// 蚁群算法类
class AntColony {
private:
    int n;
    vector<vector<double>> distance;
    vector<vector<double>> pheromone;
    vector<int> bestTour;
    double bestLength;
    mt19937 gen;

    double calculateDistance(const pair<double,double>& a, const
pair<double,double>& b) {
        return sqrt(pow(a.first - b.first, 2) + pow(a.second -
b.second, 2));
    }

    vector<int> constructSolution() {
        vector<bool> visited(n, false);
        vector<int> tour;
        int current = uniform_int_distribution<>(0, n-1)(gen);

        tour.push_back(current);
        visited[current] = true;

        while (tour.size() < n) {
            vector<double> prob;
            double total = 0;

            // 计算概率
            for (int next = 0; next < n; next++) {
                if (!visited[next]) {
```

```
                    double p = pow(pheromone[current][next], ALPHA)
*
                             pow(1.0/distance[current][next],
BETA);
                    prob.push_back(p);
                    total += p;
                } else {
                    prob.push_back(0);
                }
            }

            // 轮盘赌选择
            double r = uniform_real_distribution<>(0, total)(gen);
            double sum = 0;
            int next = -1;

            for (int i = 0; i < n && next == -1; i++) {
                if (!visited[i]) {
                    sum += prob[i];
                    if (sum >= r) {
                        next = i;
                    }
                }
            }

            if (next == -1) {
                for (int i = 0; i < n; i++) {
                    if (!visited[i]) {
                        next = i;
                        break;
                    }
                }
            }

            tour.push_back(next);
            visited[next] = true;
            current = next;
        }

        return tour;
    }
```

```
    double calculateTourLength(const vector<int>& tour) {
        double length = 0;
        for (size_t i = 0; i < tour.size(); i++) {
            int from = tour[i];
            int to = tour[(i + 1) % tour.size()];
            length += distance[from][to];
        }
        return length;
    }

public:
    AntColony(const vector<pair<double,double>>& coords) :
        gen(chrono::steady_clock::now().time_since_epoch().count())
{
        n = coords.size();
        distance.resize(n, vector<double>(n));
        pheromone.resize(n, vector<double>(n, 1.0));
        bestLength = numeric_limits<double>::max();

        for (int i = 0; i < n; i++) {
            for (int j = 0; j < n; j++) {
                distance[i][j] = calculateDistance(coords[i],
coords[j]);
            }
        }
    }

    vector<int> solve() {
        cout << "\n============ 蚁群算法优化开始 ============" <<
endl;

        for (int iter = 0; iter < MAX_ITER; iter++) {
            vector<vector<int>> antPaths(ANT_NUM);
            vector<double> pathLengths(ANT_NUM);

            #pragma omp parallel for
            for (int k = 0; k < ANT_NUM; k++) {
                antPaths[k] = constructSolution();
                pathLengths[k] = calculateTourLength(antPaths[k]);

                #pragma omp critical
                {
```

```
                    if (pathLengths[k] < bestLength) {
                        bestLength = pathLengths[k];
                        bestTour = antPaths[k];
                        cout << "迭代 " << iter << ": 新的最优解 = "
<< bestLength << endl;
                    }
                }
            }

            // 更新信息素
            for (int i = 0; i < n; i++) {
                for (int j = 0; j < n; j++) {
                    pheromone[i][j] *= (1.0 - RHO);
                }
            }

            for (int k = 0; k < ANT_NUM; k++) {
                double delta = Q / pathLengths[k];
                for (size_t i = 0; i < antPaths[k].size(); i++) {
                    int from = antPaths[k][i];
                    int to = antPaths[k][(i + 1) %
antPaths[k].size()];
                    pheromone[from][to] += delta;
                    pheromone[to][from] += delta;
                }
            }
        }

        cout << "蚁群算法最终最优解长度: " << bestLength << endl;
        return bestTour;
    }

    double getBestLength() const { return bestLength; }
};

    // MTZ约束求解器类
    class MTZSolver {
    private:
        GRBEnv* env;
        GRBModel* model;
        GRBVar** x;  // 边变量
        GRBVar* u;   // MTZ辅助变量
```

```
        int n;
        vector<pair<double,double>> coords;

    public:
        MTZSolver(const vector<pair<double,double>>& coordinates,
const vector<int>& initialTour) {
            coords = coordinates;
            n = coords.size();

            try {
                env = new GRBEnv();
                model = new GRBModel(*env);

                // 创建变量
                x = new GRBVar*[n];
                for (int i = 0; i < n; i++) {
                    x[i] = new GRBVar[n];
                }

                u = new GRBVar[n];

                // 添加变量和目标函数
                GRBLinExpr obj = 0;
                for (int i = 0; i < n; i++) {
                    for (int j = 0; j < n; j++) {
                        if (i != j) {
                            double dist = sqrt(pow(coords[i].first
- coords[j].first, 2) +
                                                pow(coords[i].second -
coords[j].second, 2));
                            x[i][j] = model->addVar(0.0, 1.0, dist,
GRB_BINARY);
                            obj += dist * x[i][j];
                        }
                    }
                    u[i] = model->addVar(0.0, n-1, 0.0,
GRB_CONTINUOUS);
                }

                model->setObjective(obj, GRB_MINIMIZE);

                // 添加约束
```

```
                // 1. 每个城市进出度为1
                for (int i = 0; i < n; i++) {
                    GRBLinExpr in = 0, out = 0;
                    for (int j = 0; j < n; j++) {
                        if (i != j) {
                            in += x[j][i];
                            out += x[i][j];
                        }
                    }
                    model->addConstr(in == 1);
                    model->addConstr(out == 1);
                }

                // 2. MTZ约束
                for (int i = 1; i < n; i++) {
                    for (int j = 1; j < n; j++) {
                        if (i != j) {
                            model->addConstr(u[i] - u[j] + n * x[i]
[j] <= n - 1);
                        }
                    }
                }

                // 设置求解参数
                model->set(GRB_DoubleParam_TimeLimit, 600);
                model->set(GRB_DoubleParam_MIPGap, 0.01);
                model->set(GRB_IntParam_Threads, 0);

                // 设置初始解
                for (size_t i = 0; i < initialTour.size() - 1; i++)
{
                    x[initialTour[i]]
[initialTour[i+1]].set(GRB_DoubleAttr_Start, 1.0);
                }
                x[initialTour.back()]
[initialTour.front()].set(GRB_DoubleAttr_Start, 1.0);

            } catch (GRBException& e) {
                cout << "Error code = " << e.getErrorCode() <<
endl;
                cout << e.getMessage() << endl;
            }
```

```
        }

        ~MTZSolver() {
            for (int i = 0; i < n; i++) {
                delete[] x[i];
            }
            delete[] x;
            delete[] u;
            delete model;
            delete env;
        }

        vector<int> solve() {
            vector<int> tour;
            try {
                cout << "\n============ MTZ优化开始 ============" <<
endl;
                model->optimize();

                if (model->get(GRB_IntAttr_Status) == GRB_OPTIMAL)
{
                    // 重建路径
                    vector<bool> visited(n, false);
                    int current = 0;
                    tour.push_back(current);
                    visited[current] = true;

                    while (tour.size() < n) {
                        for (int j = 0; j < n; j++) {
                            if (!visited[j] && x[current]
[j].get(GRB_DoubleAttr_X) > 0.5) {
                                tour.push_back(j);
                                visited[j] = true;
                                current = j;
                                break;
                            }
                        }
                    }
                }
            } catch (GRBException& e) {
                cout << "Error code = " << e.getErrorCode() <<
endl;
```

```
                cout << e.getMessage() << endl;
            }
            return tour;
        }

        double getObjectiveValue() {
            return model->get(GRB_DoubleAttr_ObjVal);
        }
    };



int main(int argc, char* argv[]) {
    if (argc < 2) {
        cout << "用法: " << argv[0] << " <tsp文件路径>" << endl;
        return 1;
    }

    string tsp_file = argv[1];
    ifstream infile(tsp_file);
    if (!infile.is_open()) {
        cout << "错误: 无法打开文件 " << tsp_file << endl;
        return 1;
    }

    vector<pair<double, double>> coords;
    string line;
    while (getline(infile, line)) {
        if (line == "NODE_COORD_SECTION") break;
    }

    while (getline(infile, line)) {
        if (line == "EOF") break;
        stringstream ss(line);
        int index;
        double x, y;
        ss >> index >> x >> y;
        coords.emplace_back(x, y);
    }
    infile.close();

    int n = coords.size();
```

```
    cout << "问题规模: " << n << " 个城市" << endl;

    try {
        // 第一阶段: 蚁群算法
        cout << "\n============ 第一阶段: 蚁群算法 ============" << endl;
        AntColony aco(coords);
        vector<int> aco_tour = aco.solve();
        double aco_length = aco.getBestLength();

        // 选择更好的解作为初始解
        vector<int> initial_tour =  aco_tour ;
        double initial_length = aco_length;

        // 第三阶段: MTZ精确求解
        cout << "\n============ 第三阶段: MTZ精确求解 ============" << endl;
        MTZSolver mtz(coords, initial_tour);
        vector<int> final_tour = mtz.solve();
        double final_length = mtz.getObjectiveValue();

        // 输出总结果
        cout << "\n============ 优化结果总结 ============" << endl;
        cout << "蚁群算法解长度: " << aco_length << endl;
        cout << "选择的初始解长度: " << initial_length << endl;
        cout << "MTZ最优解长度: " << final_length << endl;
        cout << "最终改进比例: " << (initial_length - final_length) / initial_length * 100 << "%" << endl;

        cout << "\n最优路径: ";
        for (size_t i = 0; i < final_tour.size(); i++) {
            cout << final_tour[i] << " ";
            if ((i + 1) % 20 == 0) cout << endl;
        }
        cout << endl;

    } catch (GRBException& e) {
        cout << "Gurobi错误 " << e.getErrorCode() << ": " << e.getMessage() << endl;
    } catch (const exception& e) {
        cout << "标准错误: " << e.what() << endl;
    } catch (...) {
```

```
        cout << "未知错误" << endl;
    }

    return 0;
}
```

算例

rat575

```
问题规模: 575 个城市

============ 第一阶段: 蚁群算法 ============

============ 蚁群算法优化开始 ============
迭代 0: 新的最优解 = 33267.3
迭代 0: 新的最优解 = 32350.2
迭代 0: 新的最优解 = 31836.1
迭代 1: 新的最优解 = 31389.8
迭代 1: 新的最优解 = 30487.8
迭代 2: 新的最优解 = 30432.7
迭代 5: 新的最优解 = 28871.4
迭代 5: 新的最优解 = 27796.1
迭代 10: 新的最优解 = 27679.5
迭代 12: 新的最优解 = 26986.2
迭代 13: 新的最优解 = 26418.8
迭代 14: 新的最优解 = 26186
迭代 15: 新的最优解 = 25891.2
迭代 15: 新的最优解 = 25883.1
迭代 15: 新的最优解 = 25842.8
迭代 15: 新的最优解 = 25643.8
迭代 16: 新的最优解 = 25158.7
迭代 17: 新的最优解 = 23325.8
迭代 19: 新的最优解 = 23191.4
迭代 20: 新的最优解 = 22824.2
迭代 20: 新的最优解 = 22614.1
迭代 21: 新的最优解 = 22579.6
迭代 21: 新的最优解 = 22026.3
迭代 22: 新的最优解 = 22025.4
迭代 22: 新的最优解 = 21495
```

```
迭代 23: 新的最优解 = 19537.9
迭代 25: 新的最优解 = 18831
迭代 25: 新的最优解 = 17895.7
迭代 27: 新的最优解 = 17845.1
迭代 27: 新的最优解 = 17572.9
迭代 28: 新的最优解 = 16042.3
迭代 30: 新的最优解 = 16034.5
迭代 31: 新的最优解 = 15797.6
迭代 31: 新的最优解 = 15399.7
迭代 31: 新的最优解 = 15280.8
迭代 31: 新的最优解 = 14828.3
迭代 32: 新的最优解 = 14734
迭代 33: 新的最优解 = 14251.3
迭代 34: 新的最优解 = 14163.3
迭代 34: 新的最优解 = 13785
迭代 35: 新的最优解 = 13718.5
迭代 36: 新的最优解 = 13183.8
迭代 36: 新的最优解 = 13059.5
迭代 38: 新的最优解 = 12668.4
迭代 38: 新的最优解 = 12660
迭代 39: 新的最优解 = 12352.4
迭代 40: 新的最优解 = 11863.7
迭代 41: 新的最优解 = 11854.5
迭代 42: 新的最优解 = 11839.7
迭代 42: 新的最优解 = 11625.7
迭代 43: 新的最优解 = 11301
迭代 44: 新的最优解 = 11275.8
迭代 45: 新的最优解 = 11229.3
迭代 46: 新的最优解 = 11125.2
迭代 47: 新的最优解 = 10822.5
迭代 48: 新的最优解 = 10384.9
迭代 50: 新的最优解 = 10027
迭代 55: 新的最优解 = 10011.8
迭代 59: 新的最优解 = 9687.15
迭代 61: 新的最优解 = 9526.7
迭代 69: 新的最优解 = 9454.75
迭代 70: 新的最优解 = 9409.26
迭代 72: 新的最优解 = 9405.94
迭代 79: 新的最优解 = 9232.14
迭代 80: 新的最优解 = 9175.23
迭代 84: 新的最优解 = 9116.94
迭代 88: 新的最优解 = 9080.43
```

```
最终最优解长度: 9080.43最优路径: 130 131 132 154 155 177 153 152 151
150 127 129 128 104 103 101 102 125 124 147
148 149 172 171 170 169 168 167 144 166 165 164 163 162 139 138 115
116 94 118
119 120 143 121 145 146 123 100 122 98 99 78 79 35 58 57 56 55 54
53
52 51 28 27 4 3 2 1 0 24 23 46 69 92 93 70 71 48 49 50
73 72 95 96 97 74 75 76 77 80 81 82 83 84 133 110 111 112 90 113
137 114 91 68 67 66 65 64 63 61 62 85 86 109 108 107 106 105 37 36
38 15 39 40 41 18 42 19 20 21 22 45 44 43 17 16 14 13 11 12
34 33 10 32 31 30 29 7 6 5 26 25 47 140 117 141 142 187 186 185
208 207 209 232 231 230 253 254 276 277 278 279 256 255 233 234 235
236 237 238
239 263 262 285 286 287 310 311 333 334 357 356 355 354 332 331 308
309 284 283
305 282 260 259 258 257 281 280 303 302 301 300 299 323 322 345 346
347 348 349
325 326 369 392 393 394 371 372 373 374 396 415 416 417 395 418 419
440 441 442
466 443 444 467 445 446 447 470 471 472 449 450 451 452 475 476 453
454 455 478
434 435 436 413 390 412 388 411 410 387 409 386 408 407 406 405 428
427 426 448
425 424 401 400 399 376 398 421 420 397 375 351 352 328 327 304 329
306 307 330
353 377 378 379 380 358 382 359 360 384 385 362 363 364 340 339 361
337 338 316
315 314 313 290 291 292 293 294 317 319 318 342 343 344 321 320 297
298 275 274
273 250 249 271 270 269 246 247 248 226 225 224 222 221 199 198 197
220 196 195
218 219 242 243 244 266 267 268 245 223 200 176 175 174 173 194 216
215 214 213
212 211 210 184 161 189 188 190 191 192 193 126 59 60 89 88 87 159
136 135
134 157 158 181 180 203 204 205 206 228 229 252 251 272 296 295 341
365 367 366
389 459 458 457 481 482 504 480 479 477 499 500 522 523 524 525 503
502 501 547
546 545 544 567 568 569 570 571 572 573 574 550 551 549 548 526 527
528 505 429
```

```
404 403 402 381 383 430 431 432 433 456 520 519 543 542 541 564 563
562 539 540
517 518 495 494 493 492 515 513 491 490 468 469 489 465 487 488 511
510 509 508
486 485 463 464 462 461 460 483 484 506 507 529 530 531 532 555 554
553 552 558
559 535 534 533 556 557 512 536 537 538 561 560 514 516 566 565 473
496 497 498
474 521 336 335 312 289 288 241 264 265 240 217 261 324 370 414 437
438 439 391
368 350 423 422 160 182 183 227 201 202 178 179 156 8 9

============ 第二阶段: Gurobi精确求解 ============
Set parameter Username
Set parameter LicenseID to value 2642819
Academic license - for non-commercial use only - expires 2026-03-27
Set parameter TimeLimit to value 600
Set parameter LazyConstraints to value 1
Set parameter Threads to value 0
Set parameter MIPGap to value 0.01
正在设置初始解...
开始Gurobi优化...
Gurobi Optimizer version 12.0.1 build v12.0.1rc0 (linux64 - "Ubuntu
22.04.5 LTS")

CPU model: AMD Ryzen 9 5900HX with Radeon Graphics, instruction set
[SSE2|AVX|AVX2]
Thread count: 8 physical cores, 16 logical processors, using up to
16 threads

Non-default parameters:
TimeLimit  600
MIPGap  0.01
LazyConstraints  1

Optimize a model with 575 rows, 165600 columns and 330625 nonzeros
Model fingerprint: 0xf1a76f9e
Variable types: 0 continuous, 165600 integer (165600 binary)
Coefficient statistics:
  Matrix range     [1e+00, 1e+00]
  Objective range  [2e+00, 5e+02]
  Bounds range     [1e+00, 1e+00]
```

```
  RHS range        [2e+00, 2e+00]

Warning: Completing partial solution with 164450 unfixed non-
continuous variables out of 165600
User MIP start produced solution with objective 9080.43 (0.07s)
Loaded user MIP start with objective 9080.43

Presolve removed 0 rows and 575 columns
Presolve time: 0.18s
Presolved: 575 rows, 165025 columns, 330050 nonzeros
Variable types: 0 continuous, 165025 integer (165025 binary)

Starting sifting (using dual simplex for sub-problems)...

    Iter     Pivots    Primal Obj      Dual Obj        Time
       0          0     infinity      0.0000000e+00      0s

Sifting complete



Root relaxation: objective 6.669777e+03, 916 iterations, 0.04
seconds (0.04 work units)

    Nodes    |    Current Node    |     Objective Bounds      |
Work
 Expl Unexpl |  Obj  Depth IntInf | Incumbent    BestBd   Gap |
It/Node Time

     0     0 6669.77664    0   78 9080.43020 6669.77664  26.5%
-    0s
     0     0 6697.04207    0   54 9080.43020 6697.04207  26.2%
-    3s
     0     0 6697.04207    0   54 9080.43020 6697.04207  26.2%
-    5s
     0     0 6713.72252    0   74 9080.43020 6713.72252  26.1%
-    6s
     0     0 6721.06370    0   99 9080.43020 6721.06370  26.0%
-    6s
     0     0 6722.19463    0   99 9080.43020 6722.19463  26.0%
-    7s
     0     0 6722.19463    0   99 9080.43020 6722.19463  26.0%
-   10s
```

```
     0     2 6729.95587    0   99 9080.43020 6729.95587  25.9%
-   10s
   836   882 6757.95217   73   20 9080.43020 6729.95587  25.9%
3.5   15s
  1666  1740 6784.89222  145   14 9080.43020 6729.95587  25.9%
4.1   20s
H 1821  1875                     9057.5808517 6729.95587  25.7%
4.1   21s
  2416  2510 6810.29951  208   12 9057.58085 6729.95587  25.7%
4.2   25s
  3238  3234 6841.58003  283   12 9057.58085 6729.95587  25.7%
4.3   30s
  3902  4064 6914.81612  342   12 9057.58085 6729.95587  25.7%
4.3   35s
  4675  4852 6976.16902  411    6 9057.58085 6729.95587  25.7%
4.4   40s
  5431  5409 7038.16799  474    6 9057.58085 6729.95587  25.7%
4.6   45s
  6178  6409 7091.75983  543    6 9057.58085 6729.95587  25.7%
4.7   50s
  7075  7033 7141.51524  619    6 9057.58085 6729.95587  25.7%
4.7   56s
  7431  7375 6794.98895  196   60 9057.58085 6729.95587  25.7%
4.7   60s
  7438  7380 6916.62910  378   89 9057.58085 6761.22860  25.4%
4.7   65s
  7458  7393 6782.28040   90  184 9057.58085 6770.89295  25.2%
4.7   70s
H 7553  7096                     8993.2222453 6771.95575  24.7%
4.9   73s
  7637  7160 6778.27564   24   69 8993.22225 6771.95575  24.7%
4.9   75s
H 7812  6912                     8974.2551581 6771.95575  24.5%
4.9   77s
  8045  7102 6793.26516   49   61 8974.25516 6771.95575  24.5%
5.0   80s
  8535  7419 6788.18670   37    6 8974.25516 6771.95575  24.5%
5.0   85s
  8826  7638 6805.63761   53   16 8974.25516 6771.95575  24.5%
5.1   90s
  9473  8094 6836.10665   89   44 8974.25516 6771.95575  24.5%
5.3   95s
```

```
  9630  8187 6848.35193  105    6 8974.25516 6771.95575  24.5%
5.3  108s
  9754  8310 6854.63489  107    6 8974.25516 6771.95575  24.5%
5.3  110s
H10088  8072                    7440.4780837 6771.95575  8.98%
5.4  114s
 10120  8069 6860.65995  125   36 7440.47808 6771.95575  8.98%
5.4  115s
 10494  8512 6880.96087  147    6 7440.47808 6771.95575  8.98%
5.5  120s
 11195  8770 6895.83794  175    6 7440.47808 6771.95575  8.98%
5.7  125s
 11681  9303 6905.68365  208   14 7440.47808 6771.95575  8.98%
5.7  131s
 12164  9649 6929.48331  228    6 7440.47808 6771.95575  8.98%
5.8  135s
 12709  9995 6938.13477  256    6 7440.47808 6771.95575  8.98%
5.9  140s
 13191 10273 6950.90132  292   14 7440.47808 6771.95575  8.98%
6.0  146s
 13734 10422 6963.25499  322    6 7440.47808 6771.95575  8.98%
6.1  150s
 14112 11015 6964.46609  335   10 7440.47808 6771.95575  8.98%
6.2  156s
 14838 11643 6974.70666  372    6 7440.47808 6771.95575  8.98%
6.3  163s
 15371 12016 6992.56677  384    6 7440.47808 6771.95575  8.98%
6.3  167s
 15926 11971 7004.78243  400   14 7440.47808 6771.95575  8.98%
6.4  170s
 16063 12438 7007.86255  402   22 7440.47808 6771.95575  8.98%
6.4  175s
 16802 12850 7039.91712  431    8 7440.47808 6771.95575  8.98%
6.6  183s
 17269 13024 7038.44553  428   10 7440.47808 6771.95575  8.98%
6.6  186s
 17626 13609 7043.37875  437   20 7440.47808 6771.95575  8.98%
6.7  195s
 18338 14120 7058.31404  460    8 7440.47808 6771.95575  8.98%
6.8  201s
 19169 14659 7082.63185  511    8 7440.47808 6771.95575  8.98%
6.8  207s
```

```
 20032 15287 7099.97395  562    6 7440.47808 6771.95575  8.98%
6.9  213s
 20990 15910 7117.20860  613    6 7440.47808 6771.95575  8.98%
6.9  220s
 22063 16557 7140.55549  671    6 7440.47808 6772.32925  8.98%
6.9  226s
 23100 16820 7162.49327  736   21 7440.47808 6772.32925  8.98%
6.9  231s
 23702 16646 7172.25125  761   10 7440.47808 6772.32925  8.98%
6.9  235s
 23720 16942 7176.34790  756    6 7440.47808 6772.32925  8.98%
6.9  240s
 24024 17232 7188.62388  776    8 7440.47808 6772.32925  8.98%
6.9  245s
 24425 18241 7193.20204  798   15 7440.47808 6772.32925  8.98%
6.9  252s
 25596 19053 7222.95532  876   10 7440.47808 6772.32925  8.98%
6.8  260s
 26842 19485 7278.11744  945    8 7440.47808 6772.32925  8.98%
6.8  267s
*26863 17341             952    7287.3037190 6772.32925  7.07%
6.8  267s
 27582 18222     cutoff  969      7287.30372 6772.32925  7.07%
6.8  274s
 28787 18786 6881.89473  130    8 7287.30372 6772.32925  7.07%
6.8  281s
 29332 19517 6921.28014  161   10 7287.30372 6772.32925  7.07%
6.8  288s
 30136 20708 7027.94379  277   10 7287.30372 6772.32925  7.07%
6.8  297s
 31427 21103 7284.41135  454   26 7287.30372 6773.13741  7.06%
6.7  303s
 31797 22327 6795.80534   42   89 7287.30372 6773.70809  7.05%
6.7  311s
 33033 23630 6899.41795  225   15 7287.30372 6773.70809  7.05%
6.7  320s
 34548 24513 7028.16094  414   14 7287.30372 6773.70809  7.05%
6.7  331s
 35266 25877 7053.11951  457    6 7287.30372 6773.70809  7.05%
6.7  340s
 36659 27183 7164.55786  623   14 7287.30372 6773.70809  7.05%
6.7  348s
```

```
 37995 28335 7266.62036  811   12 7287.30372 6773.70809  7.05%
6.6  356s
 39290 29489 6829.25612  130   38 7287.30372 6773.73457  7.05%
6.6  363s
 40518 29645 6882.24529  275    6 7287.30372 6773.73457  7.05%
6.6  368s
 40657 30544 6885.04233  281   10 7287.30372 6773.73457  7.05%
6.5  374s
 41580 31267 6961.96805  368   18 7287.30372 6773.73457  7.05%
6.5  381s
H41778 27957                    7216.4344548 6773.73457  6.13%
6.5  381s
 42341 29134 7042.76657  511   21 7216.43445 6773.73457  6.13%
6.5  389s
 43703 30294 7179.52398  682   39 7216.43445 6773.73457  6.13%
6.4  396s
H44157 30159                    7214.0195090 6773.73457  6.10%
6.4  396s
H44821 29903                    7208.9043281 6773.73457  6.04%
6.4  396s
 44962 31076 6809.08256   82   80 7208.90433 6773.73457  6.04%
6.3  403s
 46226 32043 6868.03791  241   40 7208.90433 6773.73457  6.04%
6.3  410s
 47228 32118 6912.55299  379   13 7208.90433 6773.73457  6.04%
6.3  415s
 47279 32874 6912.66189  380   13 7208.90433 6773.73457  6.04%
6.3  421s
 48049 33219 6953.34227  474   15 7208.90433 6773.73457  6.04%
6.2  427s
 48388 34358 7010.28367  607   29 7208.90433 6773.73457  6.04%
6.2  435s
 49604 35463 7105.22766  772   12 7208.90433 6773.73457  6.04%
6.2  442s
 50799 36569 6790.04998   21   87 7208.90433 6773.73457  6.04%
6.2  449s
 51927 37644 6880.10976  148   46 7208.90433 6773.73457  6.04%
6.2  456s
 53017 38180 6960.11422  280   42 7208.90433 6773.73457  6.04%
6.2  462s
 53549 39396 6986.08052  362   12 7208.90433 6773.73457  6.04%
6.2  470s
```

```
 54785 40544 7047.21151  523   27 7208.90433 6773.73457  6.04%
6.1  477s
 55974 41065 7098.72173  677   32 7208.90433 6773.73457  6.04%
6.1  483s
*56248 39206             748    7183.9807036 6773.73457  5.71%
6.1  483s
*56249 39114             748    7182.5999357 6773.73457  5.69%
6.1  483s
 56542 39890 7132.34645  766    8 7182.59994 6773.73457  5.69%
6.1  490s
*56660 38906             884    7170.3483444 6773.73457  5.53%
6.1  490s
 57555 39637 6790.14087   40   74 7170.34834 6776.09711  5.50%
6.1  496s
 58342 40603 6824.24514  116   66 7170.34834 6776.09711  5.50%
6.1  503s
 59367 41687 6914.43973  234    8 7170.34834 6776.09711  5.50%
6.1  510s
*59814 25286             284    6987.7553722 6776.09711  3.03%
6.1  510s
 60528 25717     cutoff  389      6987.75537 6776.09711  3.03%
6.1  517s
 61486 26642 6861.70568   69   16 6987.75537 6776.09711  3.03%
6.1  524s
 62435 27550 6910.58312  129   12 6987.75537 6776.09711  3.03%
6.2  531s
 63435 27603 6786.34910   17   99 6987.75537 6776.09711  3.03%
6.2  556s
H63436 26223                    6969.4507017 6776.09711  2.77%
6.2  557s
H63436 24912                    6946.7401149 6776.09711  2.46%
6.2  557s
H63436 23666                    6874.6125173 6776.09711  1.43%
6.2  557s
H63441 22487                    6856.4714962 6776.09711  1.17%
6.3  558s
H63441 21363                    6848.4891579 6776.09711  1.06%
6.3  558s
H63441 20294                    6835.8823949 6776.09711  0.87%
6.3  558s

Cutting planes:
```

```
  Lazy constraints: 4

Explored 63441 nodes (398749 simplex iterations) in 558.92 seconds
(535.63 work units)
Thread count was 16 (of 16 available processors)

Solution count 10: 6835.88 6848.49 6856.47 ... 7183.98

Optimal solution found (tolerance 1.00e-02)
Best objective 6.835882394850e+03, best bound 6.776097108189e+03,
gap 0.8746%

User-callback calls 202111, time in user-callback 1.45 sec

============ 优化结果 ============
蚁群算法初始解长度: 9080.43
Gurobi最优解长度: 6835.88
改进比例: 24.7185%
求解时间: 558.985 秒

最优路径: 0 1 2 3 4 26 25 47 48 71 72 73 50 49 27 28 51 52 53 54
55 79 78 77 76 75 74 97 96 95 118 119 120 143 142 141 163 164 165
166
144 167 168 169 170 171 172 149 148 126 147 146 145 121 123 122 98
99 100 124
125 102 101 103 80 81 82 83 84 105 104 128 129 127 150 151 152 153
177 155
154 132 131 130 107 106 108 109 85 86 87 88 89 65 64 63 62 61 60 59
37 36 35 58 57 56 32 31 30 29 7 6 5 8 9 10 33 34 12 11
13 14 38 39 15 16 17 40 41 18 42 19 20 21 22 45 44 43 66 67
68 91 114 137 113 90 112 111 110 133 134 135 157 158 136 160 159
182 183 206
205 228 204 203 180 181 156 179 178 202 201 200 176 175 174 173 195
196 220 197
198 199 221 223 222 244 243 266 267 268 245 246 247 248 224 225 226
227 229 252
251 250 249 271 270 269 291 292 290 313 314 315 316 317 293 294 295
296 272 273
274 275 298 297 320 321 344 343 319 318 342 341 364 365 367 366 389
390 412 388
411 410 387 409 386 408 407 406 405 428 427 426 448 425 424 423 422
444 443 442
```

```
466 465 489 490 468 469 467 445 446 447 470 471 472 449 450 451 474
473 496 497
498 520 521 522 499 477 502 501 500 523 524 525 503 504 480 479 478
456 455 454
453 476 475 452 429 430 431 432 433 434 435 413 436 459 458 457 481
482 505 528
527 526 548 549 550 551 574 573 572 571 570 569 568 546 547 545 567
544 566 565
564 563 562 539 516 540 517 541 542 543 519 518 495 494 493 492 515
491 513 514
537 538 561 560 559 558 557 555 554 553 552 529 530 531 532 533 556
534 535 536
512 511 488 487 510 486 509 508 507 506 484 483 460 461 462 485 463
464 439 438
437 414 415 416 440 441 419 420 421 398 397 396 395 418 417 371 394
393 392 391
368 369 370 372 373 374 375 376 399 400 401 402 403 404 381 383 384
385 362 363
340 339 338 337 361 360 359 382 358 380 379 378 377 353 354 332 355
356 357 336
335 312 334 333 311 310 309 308 331 330 307 329 306 304 327 328 352
351 350 349
326 325 348 347 346 345 322 323 324 299 300 301 302 303 280 281 256
279 278 277
276 254 253 255 233 234 257 258 259 260 282 305 283 284 285 261 262
263 286 287
288 289 265 264 241 242 219 218 217 240 239 238 237 236 235 212 213
214 215 216
194 193 192 191 190 189 188 187 186 211 210 209 232 231 230 207 208
185 184 161
162 139 138 140 117 94 116 115 93 92 70 69 46 23 24
```

u727

```
问题规模: 724 个城市

============ 第一阶段: 蚁群算法 ============

============ 蚁群算法优化开始 ============
迭代 0: 新的最优解 = 228191
迭代 0: 新的最优解 = 221683
```

```
迭代 0: 新的最优解 = 219540
迭代 0: 新的最优解 = 211861
迭代 0: 新的最优解 = 207304
迭代 6: 新的最优解 = 206205
迭代 9: 新的最优解 = 200793
迭代 12: 新的最优解 = 200489
迭代 15: 新的最优解 = 200330
迭代 16: 新的最优解 = 195931
迭代 17: 新的最优解 = 194038
迭代 19: 新的最优解 = 188316
迭代 20: 新的最优解 = 185466
迭代 25: 新的最优解 = 179503
迭代 29: 新的最优解 = 175657
迭代 29: 新的最优解 = 169771
迭代 31: 新的最优解 = 166253
迭代 32: 新的最优解 = 160229
迭代 34: 新的最优解 = 159679
迭代 34: 新的最优解 = 157860
迭代 35: 新的最优解 = 155685
迭代 36: 新的最优解 = 155151
迭代 36: 新的最优解 = 149970
迭代 36: 新的最优解 = 148873
迭代 36: 新的最优解 = 148347
迭代 37: 新的最优解 = 147162
迭代 38: 新的最优解 = 146449
迭代 38: 新的最优解 = 144152
迭代 39: 新的最优解 = 142595
迭代 40: 新的最优解 = 137475
迭代 40: 新的最优解 = 131205
迭代 40: 新的最优解 = 130277
迭代 42: 新的最优解 = 129364
迭代 42: 新的最优解 = 122984
迭代 43: 新的最优解 = 122877
迭代 43: 新的最优解 = 121802
迭代 44: 新的最优解 = 119756
迭代 45: 新的最优解 = 118749
迭代 45: 新的最优解 = 117482
迭代 46: 新的最优解 = 115380
迭代 46: 新的最优解 = 114754
迭代 46: 新的最优解 = 108490
迭代 46: 新的最优解 = 108064
迭代 47: 新的最优解 = 107968
```

```
迭代 48: 新的最优解 = 103540
迭代 48: 新的最优解 = 103038
迭代 49: 新的最优解 = 99962.1
迭代 49: 新的最优解 = 98842
迭代 50: 新的最优解 = 95108
迭代 51: 新的最优解 = 93401.8
迭代 52: 新的最优解 = 92959.9
迭代 52: 新的最优解 = 92680.6
迭代 53: 新的最优解 = 86077.6
迭代 55: 新的最优解 = 82358
迭代 56: 新的最优解 = 82296.9
迭代 57: 新的最优解 = 80338.2
迭代 57: 新的最优解 = 77390.7
迭代 59: 新的最优解 = 76430.1
迭代 61: 新的最优解 = 74835.8
迭代 61: 新的最优解 = 71902.3
迭代 63: 新的最优解 = 70007
迭代 64: 新的最优解 = 67994.3
迭代 67: 新的最优解 = 67243.7
迭代 69: 新的最优解 = 66671
迭代 69: 新的最优解 = 66526.9
迭代 70: 新的最优解 = 66390.2
迭代 71: 新的最优解 = 64219.2
迭代 75: 新的最优解 = 63430.4
迭代 75: 新的最优解 = 63389.4
迭代 76: 新的最优解 = 62895.4
迭代 77: 新的最优解 = 62597
迭代 77: 新的最优解 = 60361.2
迭代 83: 新的最优解 = 59911.7
迭代 85: 新的最优解 = 59824.6
迭代 88: 新的最优解 = 59294.7
迭代 92: 新的最优解 = 57359.2
最终最优解长度: 57359.2

============ 第二阶段: Gurobi精确求解 ============
Set parameter Username
Set parameter LicenseID to value 2642819
Academic license - for non-commercial use only - expires 2026-03-27
Set parameter TimeLimit to value 600
Set parameter LazyConstraints to value 1
Set parameter Threads to value 0
Set parameter MIPGap to value 0.01
```

```
正在设置初始解...
开始Gurobi优化...
Gurobi Optimizer version 12.0.1 build v12.0.1rc0 (linux64 - "Ubuntu
22.04.5 LTS")

CPU model: AMD Ryzen 9 5900HX with Radeon Graphics, instruction set
[SSE2|AVX|AVX2]
Thread count: 8 physical cores, 16 logical processors, using up to
16 threads

Non-default parameters:
TimeLimit  600
MIPGap  0.01
LazyConstraints  1

Optimize a model with 724 rows, 262450 columns and 524176 nonzeros
Model fingerprint: 0xe95f7e11
Variable types: 0 continuous, 262450 integer (262450 binary)
Coefficient statistics:
  Matrix range     [1e+00, 1e+00]
  Objective range  [3e+00, 3e+03]
  Bounds range     [1e+00, 1e+00]
  RHS range        [2e+00, 2e+00]

Warning: Completing partial solution with 261002 unfixed non-
continuous variables out of 262450
User MIP start produced solution with objective 57359.2 (0.12s)
Loaded user MIP start with objective 57359.2

Presolve removed 0 rows and 724 columns
Presolve time: 0.30s
Presolved: 724 rows, 261726 columns, 523452 nonzeros
Variable types: 0 continuous, 261726 integer (261726 binary)

Starting sifting (using dual simplex for sub-problems)...

    Iter     Pivots    Primal Obj      Dual Obj        Time
       0          0     infinity      0.0000000e+00      1s

Sifting complete
```

```
Root relaxation: objective 4.050704e+04, 1078 iterations, 0.07
seconds (0.06 work units)

    Nodes    |    Current Node    |     Objective Bounds      |
Work
 Expl Unexpl |  Obj  Depth IntInf | Incumbent    BestBd   Gap |
It/Node Time


     0     0 40507.0406    0   78 57359.1647 40507.0406  29.4%
-    0s
     0     0 40507.0406    0   78 57359.1647 40507.0406  29.4%
-    5s
     0     0 40763.1471    0   78 57359.1647 40763.1471  28.9%
-    5s
     0     0 40763.1471    0   78 57359.1647 40763.1471  28.9%
-   11s
     0     0 40820.0590    0   92 57359.1647 40820.0590  28.8%
-   11s
     0     0 40857.0386    0  103 57359.1647 40857.0386  28.8%
-   12s
     0     0 40857.5670    0  103 57359.1647 40857.5670  28.8%
-   12s
     0     0 40857.5670    0  103 57359.1647 40857.5670  28.8%
-   15s
     0     0 40857.5670    0  103 57359.1647 40857.5670  28.8%
-   20s
     0     2 41015.0016    0  116 57359.1647 41015.0016  28.5%
-   21s
   355   380 41020.1382   33   85 57359.1647 41015.0016  28.5%
3.2   25s
   938   990 41175.0679   78   50 57359.1647 41015.0016  28.5%
3.4   30s
  1404  1474 41284.2214  117   38 57359.1647 41015.0016  28.5%
3.6   35s
  1892  1977 41432.4206  165   14 57359.1647 41015.0016  28.5%
3.7   40s
H 2186  2200                    57326.390709 41015.0016  28.5%
3.6   42s
  2380  2490 41552.3720  204   12 57326.3907 41015.0016  28.5%
3.8   45s
  2748  2872 41607.0637  233   10 57326.3907 41015.0016  28.5%
3.7   50s
```

```
  3321  3460 41828.1693  286    6 57326.3907 41015.0016  28.5%
3.7   55s
  3786  3971 41908.2782  316    6 57326.3907 41015.0016  28.5%
3.7   61s
  4340  4353 42044.0699  366    6 57326.3907 41015.0016  28.5%
3.8   65s
  4725  4734 42161.0981  396    8 57326.3907 41015.0016  28.5%
3.8   70s
  5170  5414 42182.6868  422    8 57326.3907 41015.0016  28.5%
3.9   76s
  5684  5925 42308.2980  453    8 57326.3907 41015.0016  28.5%
3.9   81s
  6203  6206 42512.9647  502   12 57326.3907 41015.0016  28.5%
3.9   85s
  6501  6775 42674.5884  537   14 57326.3907 41015.0016  28.5%
3.9   92s
  6886  7098 42864.8524  576   28 57326.3907 41015.0016  28.5%
4.0   97s
  7105  7441 42979.0245  594   16 57326.3907 41015.0016  28.5%
4.0  100s
  7452  7444 42168.7388  407   70 57326.3907 41082.5389  28.3%
4.0  110s
  7456  7447 41589.3497  227   89 57326.3907 41185.0822  28.2%
4.0  117s
  7464  7452 41310.1142  140  138 57326.3907 41269.9445  28.0%
4.0  120s
  7509  7498 41305.3465   17   77 57326.3907 41291.2408  28.0%
4.2  125s
H 7540  7138                    57185.935493 41291.2408  27.8%
4.2  126s
  7679  7238 41355.5773   28   90 57185.9355 41291.2408  27.8%
4.2  130s
  7927  7418 41383.3335   46   38 57185.9355 41291.2408  27.8%
4.2  135s
  8194  7615 41439.9235   65   34 57185.9355 41291.2408  27.8%
4.3  140s
  8443  7739 41500.5183   86   38 57185.9355 41291.2408  27.8%
4.3  145s
  8665  7958 41617.2688  100   14 57185.9355 41291.2408  27.8%
4.4  150s
  8886  8102 41678.1502  113   16 57185.9355 41291.2408  27.8%
4.4  155s
```

```
  9262  8350 41788.1524  133   10 57185.9355 41291.2408  27.8%
4.4  160s
  9509  8466 41829.7715  148    6 57185.9355 41293.5020  27.8%
4.5  165s
  9879  8794 41922.9775  172    6 57185.9355 41293.5020  27.8%
4.5  171s
 10190  8993 42209.1464  191   14 57185.9355 41293.5020  27.8%
4.5  176s
 10517  9240 42290.5029  211    6 57185.9355 41293.5020  27.8%
4.6  180s
 10917  9511 42444.7400  237    8 57185.9355 41293.5020  27.8%
4.7  185s
 11313  9795 42481.5420  264    8 57185.9355 41293.5020  27.8%
4.7  190s
 11743 10097 42571.1232  292    6 57185.9355 41293.5020  27.8%
4.8  196s
 12251 10472 42706.2694  321   10 57185.9355 41293.5020  27.8%
4.8  202s
 12529 10389 42834.3849  337    6 57185.9355 41293.5020  27.8%
4.9  323s
 12545 10399 42836.8301  338    6 57185.9355 41293.5020  27.8%
4.9  326s
 12687 10493 42886.5452  348    8 57185.9355 41293.5020  27.8%
4.9  331s
 12831 10589 42938.6062  357    8 57185.9355 41293.5020  27.8%
4.9  336s
 12956 10750 42978.2376  370   12 57185.9355 41293.5020  27.8%
5.0  341s
 13128 10897 43052.6711  383   33 57185.9355 41293.5020  27.8%
5.0  346s
 13266 11056 43044.2331  385   48 57185.9355 41293.5020  27.8%
5.1  352s
 13459 11002 43124.3235  397   28 57185.9355 41293.5020  27.8%
5.1  356s
 13635 11115 43191.4770  408   20 57185.9355 41293.5020  27.8%
5.1  362s
 13643 11286 43160.1488  409   30 57185.9355 41293.5020  27.8%
5.1  365s
 13850 11403 43223.1744  422   14 57185.9355 41293.5020  27.8%
5.2  372s
 14040 11525 43287.0876  433   20 57185.9355 41293.5020  27.8%
5.2  376s
```

```
 14213 11785 43424.7047  445   28 57185.9355 41293.5020  27.8%
5.2  381s
 14655 11864 43561.5807  475    6 57185.9355 41293.5020  27.8%
5.2  387s
 14766 12768 43588.6522  482    6 57185.9355 41293.5020  27.8%
5.3  401s
 15726 13388 43778.8283  542    6 57185.9355 41293.5020  27.8%
5.4  417s
 16686 14007 44164.9403  602   16 57185.9355 41293.5020  27.8%
5.5  428s
 17646 14617 44333.8300  660    6 57185.9355 41293.5020  27.8%
5.6  441s
 18606 15267 44555.4099  720   14 57185.9355 41294.3225  27.8%
5.6  454s
 19566 15901 44711.9379  780    6 57185.9355 41294.3225  27.8%
5.7  464s
 20526 16505 44872.6220  840    8 57185.9355 41294.3225  27.8%
5.8  475s
 21486 16463 45444.3567  897   42 57185.9355 41294.3225  27.8%
5.9  495s
 21762 16881 45735.1733  914    6 57185.9355 41294.3225  27.8%
5.9  522s
 22287 17654 45850.9373  937    6 57185.9355 41294.3225  27.8%
5.9  531s
 23247 18296 46291.7203  997   12 57185.9355 41294.3225  27.8%
6.0  542s
 24207 18907 46617.7544 1057   14 57185.9355 41294.3225  27.8%
6.0  553s
 25167 19560 41756.5941   44   40 57185.9355 41294.3225  27.8%
6.0  563s
 26127 20194 41950.9950  105   12 57185.9355 41294.3225  27.8%
6.1  573s
 27087 20816 42065.4963  143    6 57185.9355 41294.3225  27.8%
6.1  585s
 28047 21452 42196.9784  195    8 57185.9355 41294.3225  27.8%
6.2  596s
 29007 21577 42287.2688  236    8 57185.9355 41294.3225  27.8%
6.2  600s

Cutting planes:
  Gomory: 13
  Lift-and-project: 36
```

```
  Zero half: 58
  Lazy constraints: 191

Explored 29353 nodes (182403 simplex iterations) in 600.03 seconds
(720.74 work units)
Thread count was 16 (of 16 available processors)

Solution count 3: 57185.9 57326.4 57359.2

Time limit reached
Best objective 5.718593549330e+04, best bound 4.129432253200e+04,
gap 27.7894%

User-callback calls 219322, time in user-callback 1.33 sec

未找到最优解
求解状态: 9
```

u1060

```
问题规模: 1060 个城市

============ 第一阶段: 蚁群算法 ============

============ 蚁群算法优化开始 ============
迭代 0: 新的最优解 = 1.28531e+06
迭代 0: 新的最优解 = 1.25017e+06
迭代 0: 新的最优解 = 1.22558e+06
迭代 5: 新的最优解 = 1.1951e+06
迭代 6: 新的最优解 = 1.17603e+06
迭代 27: 新的最优解 = 1.16638e+06
迭代 28: 新的最优解 = 1.14151e+06
迭代 34: 新的最优解 = 1.12887e+06
迭代 37: 新的最优解 = 1.12552e+06
迭代 38: 新的最优解 = 1.11264e+06
迭代 39: 新的最优解 = 1.10978e+06
迭代 39: 新的最优解 = 1.10678e+06
迭代 39: 新的最优解 = 1.08949e+06
迭代 41: 新的最优解 = 1.08935e+06
迭代 41: 新的最优解 = 1.08294e+06
迭代 42: 新的最优解 = 1.06543e+06
```

```
迭代 42: 新的最优解 = 1.04509e+06
迭代 42: 新的最优解 = 1.0445e+06
迭代 42: 新的最优解 = 1.02843e+06
迭代 45: 新的最优解 = 1.00734e+06
迭代 46: 新的最优解 = 1.0006e+06
迭代 46: 新的最优解 = 963301
迭代 48: 新的最优解 = 947746
迭代 51: 新的最优解 = 907448
迭代 52: 新的最优解 = 854353
迭代 53: 新的最优解 = 853465
迭代 53: 新的最优解 = 849742
迭代 54: 新的最优解 = 822544
迭代 55: 新的最优解 = 819626
迭代 55: 新的最优解 = 800681
迭代 56: 新的最优解 = 778290
迭代 57: 新的最优解 = 764623
迭代 58: 新的最优解 = 730548
迭代 59: 新的最优解 = 719796
迭代 59: 新的最优解 = 715129
迭代 61: 新的最优解 = 702328
迭代 62: 新的最优解 = 679496
迭代 62: 新的最优解 = 663537
迭代 63: 新的最优解 = 650982
迭代 64: 新的最优解 = 637803
迭代 64: 新的最优解 = 624677
迭代 65: 新的最优解 = 608675
迭代 65: 新的最优解 = 607364
迭代 65: 新的最优解 = 603961
迭代 66: 新的最优解 = 582144
迭代 67: 新的最优解 = 569319
迭代 68: 新的最优解 = 558405
迭代 69: 新的最优解 = 536119
迭代 70: 新的最优解 = 535037
迭代 71: 新的最优解 = 517441
迭代 71: 新的最优解 = 513643
迭代 72: 新的最优解 = 512155
迭代 72: 新的最优解 = 502723
迭代 73: 新的最优解 = 493870
迭代 73: 新的最优解 = 490388
迭代 74: 新的最优解 = 456236
迭代 74: 新的最优解 = 449591
迭代 75: 新的最优解 = 446999
```

```
迭代 79: 新的最优解 = 444563
迭代 79: 新的最优解 = 427471
迭代 80: 新的最优解 = 396870
迭代 86: 新的最优解 = 396739
迭代 86: 新的最优解 = 379324
迭代 87: 新的最优解 = 379266
迭代 89: 新的最优解 = 374382
迭代 91: 新的最优解 = 369931
迭代 92: 新的最优解 = 363425
迭代 92: 新的最优解 = 353786
迭代 98: 新的最优解 = 353369
迭代 99: 新的最优解 = 350974
迭代 99: 新的最优解 = 339162
最终最优解长度: 339162

============ 第二阶段: Gurobi精确求解 ============
Set parameter Username
Set parameter LicenseID to value 2642819
Academic license - for non-commercial use only - expires 2026-03-27
Set parameter TimeLimit to value 600
Set parameter LazyConstraints to value 1
Set parameter Threads to value 0
Set parameter MIPGap to value 0.01
正在设置初始解...
开始Gurobi优化...
Gurobi Optimizer version 12.0.1 build v12.0.1rc0 (linux64 - "Ubuntu
22.04.5 LTS")

CPU model: AMD Ryzen 9 5900HX with Radeon Graphics, instruction set
[SSE2|AVX|AVX2]
Thread count: 8 physical cores, 16 logical processors, using up to
16 threads

Non-default parameters:
TimeLimit  600
MIPGap  0.01
LazyConstraints  1

Optimize a model with 1060 rows, 562330 columns and 1123600
nonzeros
Model fingerprint: 0xf96ce99a
Variable types: 0 continuous, 562330 integer (562330 binary)
```

```
Coefficient statistics:
  Matrix range     [1e+00, 1e+00]
  Objective range  [7e+01, 2e+04]
  Bounds range     [1e+00, 1e+00]
  RHS range        [2e+00, 2e+00]

Warning: Completing partial solution with 560210 unfixed non-
continuous variables out of 562330
User MIP start produced solution with objective 339162 (0.26s)
Loaded user MIP start with objective 339162

Presolve removed 0 rows and 1060 columns
Presolve time: 0.79s
Presolved: 1060 rows, 561270 columns, 1122540 nonzeros
Variable types: 0 continuous, 561270 integer (561270 binary)
Root relaxation presolved: 1060 rows, 561270 columns, 1122540
nonzeros

Deterministic concurrent LP optimizer: primal simplex, dual
simplex, and barrier
Showing barrier log only...

Root barrier log...

Ordering time: 0.02s

Barrier statistics:
 AA' NZ     : 5.613e+05
 Factor NZ  : 5.623e+05 (roughly 230 MB of memory)
 Factor Ops : 3.976e+08 (less than 1 second per iteration)
 Threads    : 6

                  Objective                Residual
Iter       Primal          Dual         Primal    Dual     Compl
  Time
   0   2.53817875e+09 -8.95839614e+09  7.49e+02 2.73e-12  1.93e+04
    2s
   1   9.90961605e+07 -1.58863431e+09  3.95e+01 1.55e-11  1.74e+03
    2s
   2   4.18740293e+06 -1.27100169e+07  1.31e-13 1.46e-11  1.51e+01
    2s
```

```
   3   3.22242694e+06 -2.85424688e+06  1.33e-14 9.09e-12  5.41e+00
    2s

Barrier performed 3 iterations in 2.23 seconds (2.27 work units)
Barrier solve interrupted - model solved by another algorithm

Concurrent spin time: 0.20s (can be avoided by choosing Method=3)

Solved with dual simplex

Root relaxation: objective 2.095214e+05, 6056 iterations, 1.04
seconds (0.66 work units)

    Nodes    |    Current Node    |     Objective Bounds      |
Work
 Expl Unexpl |  Obj  Depth IntInf | Incumbent    BestBd   Gap |
It/Node Time

     0     0 209521.386    0  148 339161.942 209521.386  38.2%
-    2s
     0     0 209521.386    0  148 339161.942 209521.386  38.2%
-    5s
     0     0 209521.386    0  148 339161.942 209521.386  38.2%
-   11s
     0     0 211474.102    0  126 339161.942 211474.102  37.6%
-   15s
     0     0 211474.102    0  126 339161.942 211474.102  37.6%
-   20s
     0     0 211474.102    0  126 339161.942 211474.102  37.6%
-   25s
     0     0 211979.170    0  133 339161.942 211979.170  37.5%
-   28s
     0     0 211979.170    0  133 339161.942 211979.170  37.5%
-   30s
     0     0 212603.551    0  171 339161.942 212603.551  37.3%
-   34s
     0     0 212608.030    0  160 339161.942 212608.030  37.3%
-   35s
     0     0 212623.552    0  160 339161.942 212623.552  37.3%
-   41s
     0     0 212623.552    0  160 339161.942 212623.552  37.3%
-   45s
```

```
     0     0 212623.552    0  160 339161.942 212623.552  37.3%
-    50s
     0     2 213176.896    0  182 339161.942 213176.896  37.1%
-    54s
     1     4 213176.896    1  166 339161.942 213176.896  37.1%
12.0   55s
   166   188 213284.546   17   77 339161.942 213176.896  37.1%
3.7   60s
   464   496 213507.965   44   58 339161.942 213176.896  37.1%
3.1   65s
   750   779 213627.240   72   14 339161.942 213176.896  37.1%
3.1   70s
   986   985 213875.373   89   18 339161.942 213176.896  37.1%
3.4   75s
  1191  1239 214221.693  106   14 339161.942 213176.896  37.1%
3.5   80s
  1507  1567 214794.287  134   22 339161.942 213176.896  37.1%
3.6   86s
  1666  1733 214987.145  149   12 339161.942 213176.896  37.1%
3.6   90s
  1927  2003 215167.769  171   12 339161.942 213176.896  37.1%
3.5   95s
  2214  2305 215450.973  195   14 339161.942 213176.896  37.1%
3.6  100s
  2413  2506 215962.785  216   16 339161.942 213176.896  37.1%
3.6  106s
  2630  2736 216165.319  230   30 339161.942 213176.896  37.1%
3.6  111s
  2892  2912 216205.971  250   14 339161.942 213176.896  37.1%
3.6  115s
  2938  3063 216362.576  252   12 339161.942 213176.896  37.1%
3.6  120s
  3231  3360 216605.333  274   12 339161.942 213176.896  37.1%
3.7  126s
  3540  3679 216780.551  300   12 339161.942 213176.896  37.1%
3.7  132s
  3704  3844 216756.106  312   12 339161.942 213176.896  37.1%
3.7  136s
  4049  4031 217335.007  341   16 339161.942 213176.896  37.1%
3.8  141s
  4057  4183 217291.479  341   12 339161.942 213176.896  37.1%
3.8  145s
```

```
  4217  4374 217332.100  355   12 339161.942 213176.896  37.1%
3.8  151s
  4400  4580 217692.759  370   12 339161.942 213176.896  37.1%
3.8  155s
  4827  4809 217522.610  407   16 339161.942 213176.896  37.1%
3.8  162s
  4835  5035 217790.399  407   14 339161.942 213176.896  37.1%
3.8  166s
  5067  5279 217600.245  427   14 339161.942 213176.896  37.1%
3.8  171s
  5316  5525 218502.907  444   12 339161.942 213176.896  37.1%
3.8  177s
  5560  5785 218507.884  461   14 339161.942 213176.896  37.1%
3.8  182s
  5823  6058 218793.593  485   12 339161.942 213176.896  37.1%
3.9  187s
  6094  6066 218799.731  506   28 339161.942 213176.896  37.1%
3.9  190s
  6102  6364 218970.722  506   12 339161.942 213176.896  37.1%
3.9  195s
  6400  6690 219336.497  534   12 339161.942 213176.896  37.1%
3.9  201s
  6726  7007 219509.144  555   14 339161.942 213176.896  37.1%
3.9  207s
  7043  7266 219680.300  579   18 339161.942 213176.896  37.1%
3.9  212s
  7303  7267 215199.624  196  160 339161.942 213176.896  37.1%
3.9  215s
  7305  7268 218285.830  581  148 339161.942 213898.986  36.9%
3.9  223s
  7306  7269 215019.831  142  121 339161.942 215019.831  36.6%
3.9  240s
  7310  7272 219806.343  619   85 339161.942 215449.686  36.5%
3.9  248s
  7314  7274 215573.256   87  122 339161.942 215573.256  36.4%
3.9  250s
  7323  7281 217565.583  301  136 339161.942 215655.347  36.4%
4.9  261s
  7324  7282 216432.638  278  122 339161.942 215655.347  36.4%
4.9  280s
  7329  7285 216602.561  256  158 339161.942 215896.143  36.3%
4.9  290s
```

```
  7334  7289 216168.613  190  145 339161.942 216168.613  36.3%
4.9  299s
  7335  7292 216168.613   27  140 339161.942 216168.613  36.3%
5.8  300s
  7411  7357 216168.613   34   57 339161.942 216168.613  36.3%
5.8  305s
  7496  7412 216283.630   40   38 339161.942 216168.613  36.3%
5.8  310s
  7603  7490 216457.558   47   36 339161.942 216168.613  36.3%
5.8  315s
  7728  7576 216431.225   56   38 339161.942 216168.613  36.3%
5.7  320s
  7829  7649 216463.342   62   36 339161.942 216168.613  36.3%
5.7  325s
  7960  7744 216521.665   70   34 339161.942 216168.613  36.3%
5.7  330s
  8080  7827 216524.358   77   32 339161.942 216168.613  36.3%
5.7  335s
H 8150  7471                        336377.83177 216168.613  35.7%
5.7  336s
  8160  7516 216534.915   81   36 336377.832 216168.613  35.7%
5.7  340s
  8256  7596 216700.313   86   36 336377.832 216168.613  35.7%
5.7  345s
  8437  7729 216715.305   97   36 336377.832 216168.613  35.7%
5.7  351s
  8566  7819 216765.837  103   38 336377.832 216168.613  35.7%
5.7  356s
  8706  7905 216974.147  110   36 336377.832 216168.613  35.7%
5.6  361s
  8860  7950 216957.243  119   28 336377.832 216168.613  35.7%
5.6  365s
  8876  8014 217287.151  120   54 336377.832 216168.613  35.7%
5.7  370s
  9042  8167 217232.468  132   58 336377.832 216168.613  35.7%
5.7  377s
  9146  8249 217435.646  139   54 336377.832 216168.613  35.7%
5.6  380s
  9387  8420 217469.911  154   52 336377.832 216168.613  35.7%
5.6  387s
  9514  8522 217541.480  162   56 336377.832 216168.613  35.7%
5.6  391s
```

```
  9666  8640 217669.704  174   44 336377.832 216168.613  35.7%
5.6  398s
  9827  8747 217943.697  185   44 336377.832 216168.613  35.7%
5.6  402s
  9987  8848 218180.757  195   44 336377.832 216168.613  35.7%
5.6  406s
 10150  8948 218306.164  205   38 336377.832 216168.613  35.7%
5.6  413s
 10296  9057 218501.652  216   28 336377.832 216168.613  35.7%
5.6  418s
 10454  9193 218455.736  225   28 336377.832 216168.613  35.7%
5.6  422s
 10642  9317 218519.218  236   30 336377.832 216168.613  35.7%
5.6  427s
 10829  9452 218610.586  245   28 336377.832 216168.613  35.7%
5.6  433s
 11026  9394 218710.575  254   28 336377.832 216168.613  35.7%
5.6  435s
 11034  9582 218993.860  255   28 336377.832 216168.613  35.7%
5.6  441s
 11225  9695 219223.447  268   42 336377.832 216168.613  35.7%
5.6  446s
 11401  9832 219248.755  279   28 336377.832 216168.613  35.7%
5.6  452s
 11597  9990 219417.149  290   28 336377.832 216168.613  35.7%
5.6  458s
 11820 10164 219723.728  303   32 336377.832 216168.613  35.7%
5.6  465s
 12069 10089 219802.564  318   30 336377.832 216168.613  35.7%
5.6  600s

Cutting planes:
  Gomory: 7
  Lift-and-project: 22
  Zero half: 68
  Lazy constraints: 81

Explored 12077 nodes (74363 simplex iterations) in 600.06 seconds
(583.58 work units)
Thread count was 16 (of 16 available processors)

Solution count 2: 336378 339162
```

```
Time limit reached
Best objective 3.363778317746e+05, best bound 2.161686130903e+05,
gap 35.7364%

User-callback calls 111578, time in user-callback 1.92 sec

未找到最优解
求解状态: 9
```