# 并行单纯形法

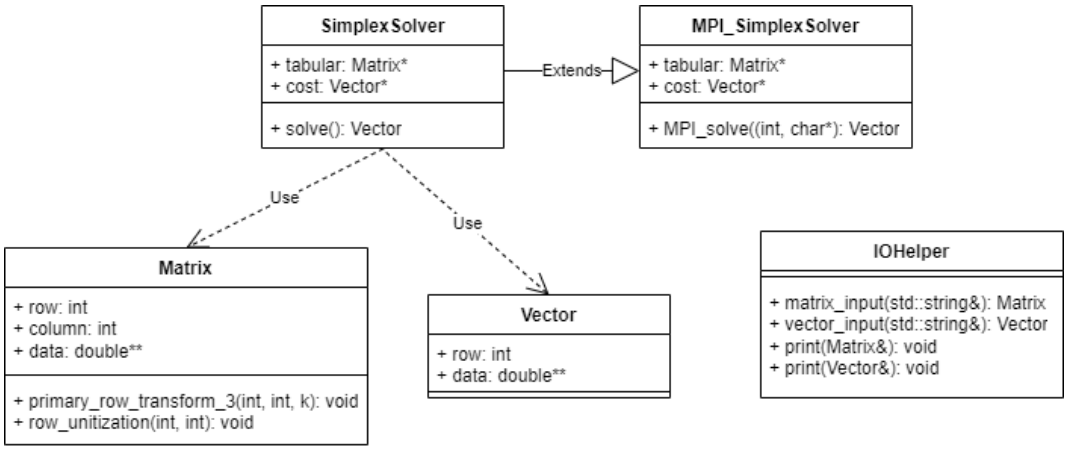
211840196 张博阳

## 1 实现思路

Richard Marciano, Teodor Rus在1988年就提出利用并行手段加快换基结束后迭代单纯形表的计算速度[1]. 在2017年, Péter Tar等提出了利用密集计算等方法对单纯形法进行并行加速, 其中对于凸多面体的顶点寻优问题, 从多个初始位置开始进行单纯形表迭代可以实现加速及更好的鲁棒性表现[1]. 本项目从这两个方向分别进行单纯形法的并行化实现. 由于计算资源的有限, 仅对两种方法分别进行实现, 而无对进程分组的两种方法结合实现并行.

## 2 代码框架及并行化手段

项目类图如下. `SimplexSolver`类为单线程下的单纯形法求解器类, 其中依赖`Matrix`和`Vector`两个工具类.



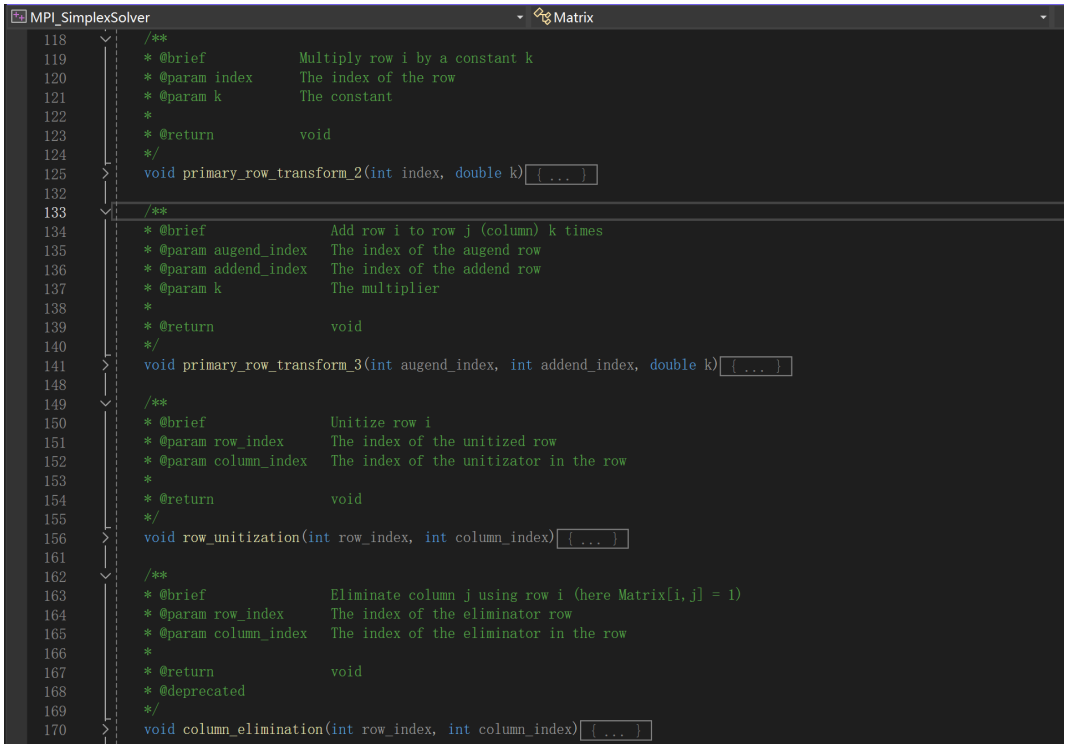

在`SimplexSolver`基础上扩展为`MPI_SimplexSolver`, 其使用MPI接口对核心函数`solve()`实现并行. `IOHelper`为项目进行I/O操作的辅助类.

`Matrix`类封装了二维数组和初等行变换等单纯形法需要的矩阵行为.

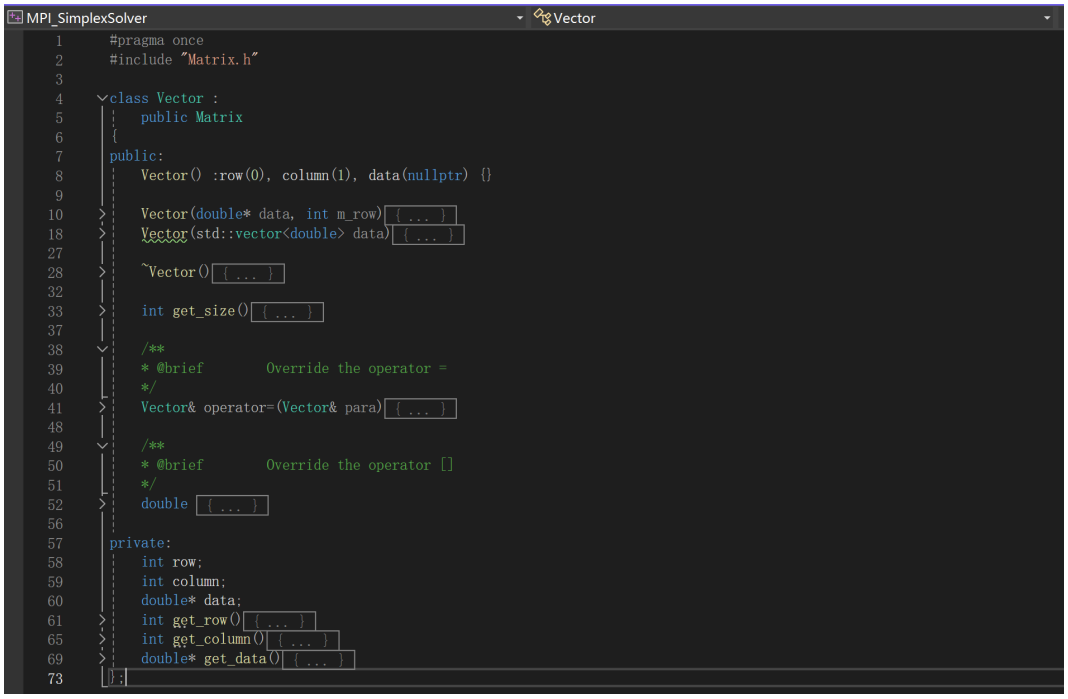
`Vector`类是介于一维数组与`Matrix`类间的中介类.

`SimplexSolver`类持有一个`Matrix`和`Vector`作为单纯形表和判别数列表, 核心功能函数为单纯形法求解器`solve()`.

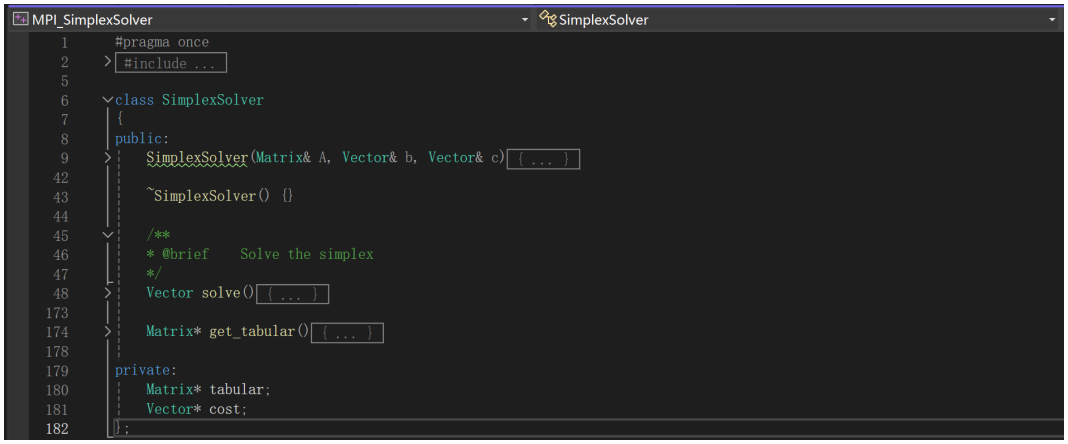
`MPI_SimplexSolver`类继承`SimplexSolver`类, 核心功能函数为`MPI_solve_parallel_trans()`和`MPI_solve_dir_search()`分别实现了迭代并行和多路径搜索并行计算.

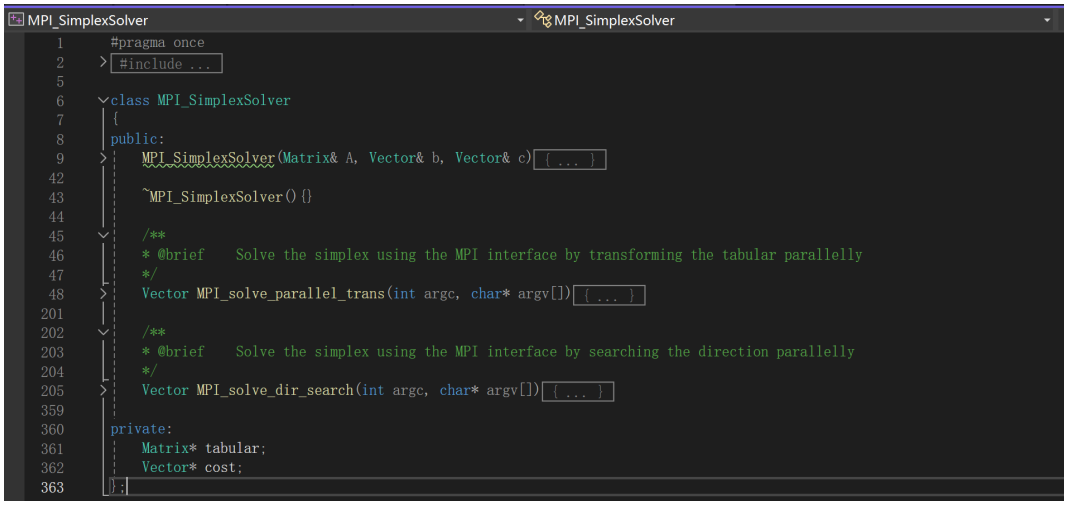



```
118     /**
119     * @brief            Multiply row i by a constant k
120     * @param index      The index of the row
121     * @param k          The constant
122     *
123     * @return           void
124     */
125     void primary_row_transform_2(int index, double k){ ... }
132
133     /**
134     * @brief                Add row i to row j (column) k times
135     * @param augend_index   The index of the augend row
136     * @param addend_index   The index of the addend row
137     * @param k              The multiplier
138     *
139     * @return               void
140     */
141     void primary_row_transform_3(int augend_index, int addend_index, double k){ ... }
148
149     /**
150     * @brief                Unitize row i
151     * @param row_index      The index of the unitized row
152     * @param column_index   The index of the unitizator in the row
153     *
154     * @return               void
155     */
156     void row_unitization(int row_index, int column_index){ ... }
161
162     /**
163     * @brief                Eliminate column j using row i (here Matrix[i,j] = 1)
164     * @param row_index      The index of the eliminator row
165     * @param column_index   The index of the eliminator in the row
166     *
167     * @return               void
168     * @deprecated
169     */
170     void column_elimination(int row_index, int column_index){ ... }
```




```
1   #pragma once
2   #include "Matrix.h"
3
4   class Vector :
5       public Matrix
6   {
7   public:
8       Vector() :row(0), column(1), data(nullptr) {}
9
10      Vector(double* data, int m_row){ ... }
18      Vector(std::vector<double> data){ ... }
27
28      ~Vector(){ ... }
32
33      int get_size(){ ... }
37
38      /**
39      * @brief        Override the operator =
40      */
41      Vector& operator=(Vector& para){ ... }
48
49      /**
50      * @brief        Override the operator []
51      */
52      double  { ... }
56
57  private:
58      int row;
59      int column;
60      double* data;
61      int get_row(){ ... }
65      int get_column(){ ... }
69      double* get_data(){ ... }
73  };
```


MPI_SimplexSolver
SimplexSolver
#pragma once
#include ...
class SimplexSolver
{
public:
SimplexSolver(Matrix& A, Vector& b, Vector& c) { ... }
~SimplexSolver() {}
/**
* @brief Solve the simplex
*/
Vector solve() { ... }
Matrix* get_tabular() { ... }
private:
Matrix* tabular;
Vector* cost;
};


MPI_SimplexSolver
MPI_SimplexSolver
#pragma once
#include ...
class MPI_SimplexSolver
{
public:
MPI_SimplexSolver(Matrix& A, Vector& b, Vector& c) { ... }
~MPI_SimplexSolver(){}
/**
* @brief Solve the simplex using the MPI interface by transforming the tabular parallelly
*/
Vector MPI_solve_parallel_trans(int argc, char* argv[]) { ... }
/**
* @brief Solve the simplex using the MPI interface by searching the direction parallelly
*/
Vector MPI_solve_dir_search(int argc, char* argv[]) { ... }
private:
Matrix* tabular;
Vector* cost;
};

## 3 数值表现

项目用单线程、MPI_solve_parallel_trans()和MPI_solve_dir_search()分别求解线性规划问题, 其中MPI_solve_dir_s
采用变量字典序多路搜索.

$$
\begin{aligned}
\min f &= -2x_1 - 3x_2 \\
\text{s.t.} -x_1 + 2x_2 + x_3 &= 12 \\
2x_1 + 3x_2 + x_4 &= 16 \\
x_1 - x_2 + x_5 &= 15 \\
x_i \geq 0, i &= 1, 2, 3, 4, 5
\end{aligned}
$$

重复求解上述问题100000次，运行时间如下. MPI_solve_parallel_trans()并行加速效果较好, MPI_solve_dir_search()因

| 单线程 | MPI_solve_parallel_trans() | MPI_solve_dir_search() |
|---|---|---|
| 0.40092s | 0.29572s | 0.38109s |

路径选取为简单字典序选取, 仅有少量加速.

## References

[1] Richard Marciano, Teodor Rus, Parallel Implementations of the Simplex Algorithm, Proceedings., 2nd Symposium on the Frontiers of Massively Parallel Computation, 10-12, October 1988.

[2] Péter Tar, Bálint Stágel, István Maros, Parallel search paths for the simplex algorithm, Central European Journal of Operations Research, (2017) 25:967–984.