LAP (Linear Assignment Problem)

Создано системой Doxygen 1.9.4

Глава 1

Алфавитный указатель групп

1.1 Группы

Полный список групп.

Глава 2

Алфавитный указатель пространств имен

2.1 Пространства имен

Полный список пространств имен.

Глава 3

Алфавитный указатель классов

3.1 Классы

Классы с их кратким описанием.

Глава 4

Список файлов

4.1 Файлы

Полный список файлов.

Глава 5

Группы

5.1 Spml

Пространства имен

- namespace SPML

  Специальная библиотека программных модулей (СБ ПМ)

5.1.1 Подробное описание

Глава 6

Пространства имен

6.1 Пространство имен SPML

Специальная библиотека программных модулей (СБ ПМ)

Пространства имен

- namespace Compare
  Сравнение чисел
- namespace LAP
  Решение задачи о назначениях
- namespace Sparse
  Решение задачи о назначениях

6.1.1 Подробное описание

Специальная библиотека программных модулей (СБ ПМ)

Специальная библиотека программных модулей (СБПМ)

6.2 Пространство имен SPML::Compare

Сравнение чисел

Функции

- bool AreEqualAbs (float first, float second, const float &eps=EPS_F)
  Сравнение двух действительных чисел (по абсолютной разнице)
- bool AreEqualAbs (double first, double second, const double &eps=EPS_D)
  Сравнение двух действительных чисел (по абсолютной разнице)
- bool AreEqualRel (float first, float second, const float &eps=EPS_REL)
  Сравнение двух действительных чисел (по относительной разнице)
- bool AreEqualRel (double first, double second, const double &eps=EPS_REL)
  Сравнение двух действительных чисел (по относительной разнице)
- bool IsZeroAbs (float value, const float &eps=EPS_F)
  Проверка действительного числа на равенство нулю (по абсолютной разнице)
- bool IsZeroAbs (double value, const double &eps=EPS_D)
  Проверка действительного числа на равенство нулю (по абсолютной разнице)

6.2.1 Подробное описание

Сравнение чисел

6.2.2 Функции

6.2.2.1 AreEqualAbs() [1/2]

```
bool SPML::Compare::AreEqualAbs (
                double first,
                double second,
                const double & eps = EPS_D )   [inline]
```

Сравнение двух действительных чисел (по абсолютной разнице)

Возвращает результат: abs(first - second) < eps

Аргументы

in	first	- первое число
in	second	- второе число
in	eps	- абсолютная точность сравнения

Возвращает

true - если разница меньше точности, иначе false

См. определение в файле compare.hpp строка 48

6.2.2.2 AreEqualAbs() [2/2]

```
bool SPML::Compare::AreEqualAbs (
                float first,
                float second,
                const float & eps = EPS_F )   [inline]
```

Сравнение двух действительных чисел (по абсолютной разнице)

Возвращает результат: abs(first - second) < eps

Аргументы

in	first	- первое число
in	second	- второе число
in	eps	- абсолютная точность сравнения

Возвращает

true - если разница меньше точности, иначе false

См. определение в файле compare.hpp строка 35

6.2.2.3 AreEqualRel() [1/2]

```
bool SPML::Compare::AreEqualRel (
                double first,
                double second,
                const double & eps = EPS_REL )   [inline]
```

Сравнение двух действительных чисел (по относительной разнице)

Возарвщает результат: (abs((first - second) / first) < eps) && (abs((first - second) / second) < eps)

Аргументы

in	first	- первое число
in	second	- второе число
in	eps	- относительная точность сравнения

Возвращает

true - если разница меньше точности, иначе false

См. определение в файле compare.hpp строка 76

6.2.2.4 AreEqualRel() [2/2]

```
bool SPML::Compare::AreEqualRel (
                float first,
                float second,
                const float & eps = EPS_REL )   [inline]
```

Сравнение двух действительных чисел (по относительной разнице)

Возарвщает результат: (abs((first - second) / first) < eps) && (abs((first - second) / second) < eps)

Аргументы

in	first	- первое число
in	second	- второе число
in	eps	- относительная точность сравнения

Возвращает

true - если разница меньше точности, иначе false

См. определение в файле compare.hpp строка 62

6.2.2.5 IsZeroAbs() [1/2]

```
bool SPML::Compare::IsZeroAbs (
            double value,
            const double & eps = EPS_D )   [inline]
```

Проверка действительного числа на равенство нулю (по абсолютной разнице)

Возвращает результат: abs(value) < eps

Аргументы

in	value	- проверяемое число
in	eps	- абсолютная точность сравнения

Возвращает

true - если разница меньше точности, иначе false

См. определение в файле compare.hpp строка 102

6.2.2.6 IsZeroAbs() [2/2]

```
bool SPML::Compare::IsZeroAbs (
            float value,
            const float & eps = EPS_F )   [inline]
```

Проверка действительного числа на равенство нулю (по абсолютной разнице)

Возвращает результат: abs(value) < eps

Аргументы

in	value	- проверяемое число
in	eps	- абсолютная точность сравнения

Возвращает

true - если разница меньше точности, иначе false

См. определение в файле compare.hpp строка 90

6.3 Пространство имен SPML::LAP

Решение задачи о назначениях

Классы

- struct LapConstraints
- class Murty

  Класс решения K-best задачи методом Murty.
- struct MurtyNode

  Узел решения
- struct MurtySolution

  Решение метода Murty.

Перечисления

- enum TSearchParam { SP_Min , SP_Max }

  Критерий поиска - минимум/максимум для задачи о назначениях

Функции

- void hungarian_step_1 (unsigned int &step, arma::mat &cost, const unsigned int &N)
- void hungarian_step_2 (unsigned int &step, const arma::mat &cost, arma::umat &indM, arma::ivec &rcov, arma::ivec &ccov, const unsigned int &N)
- void hungarian_step_3 (unsigned int &step, const arma::umat &indM, arma::ivec &ccov, const unsigned int &N)
- void hungarian_find_noncovered_zero (int &row, int &col, const arma::mat &cost, const arma::ivec &rcov, const arma::ivec &ccov, const unsigned int &N)
- bool hungarian_star_in_row (int &row, const arma::umat &indM, const unsigned int &N)
- void hungarian_find_star_in_row (const int &row, int &col, const arma::umat &indM, const unsigned int &N)
- void hungarian_step_4 (unsigned int &step, const arma::mat &cost, arma::umat &indM, arma::ivec &rcov, arma::ivec &ccov, int &rpath_0, int &cpath_0, const unsigned int &N)
- void hungarian_find_star_in_col (const int &col, int &row, const arma::umat &indM, const unsigned int &N)
- void hungarian_find_prime_in_row (const int &row, int &col, const arma::umat &indM, const unsigned int &N)
- void hungarian_augment_path (const int &path_count, arma::umat &indM, const arma::imat &path)
- void hungarian_clear_covers (arma::ivec &rcov, arma::ivec &ccov)
- void hungarian_erase_primes (arma::umat &indM, const unsigned int &N)
- void hungarian_step_5 (unsigned int &step, arma::umat &indM, arma::ivec &rcov, arma::ivec &ccov, arma::imat &path, int &rpath_0, int &cpath_0, const unsigned int &N)
- void hungarian_find_smallest (double &minval, const arma::mat &cost, const arma::ivec &rcov, const arma::ivec &ccov, const unsigned int &N)
- void hungarian_step_6 (unsigned int &step, arma::mat &cost, const arma::ivec &rcov, const arma::ivec &ccov, const unsigned int &N)

- void Hungarian (const arma::mat &assigncost, int dim, TSearchParam sp, double infValue, double resolution, arma::ivec &rowsol, double &lapcost)

  Венгерский метод решения задачи о назначениях (Метод Мункреса)
- void JVCdense (const arma::mat &assigncost, int dim, TSearchParam sp, double infValue, double resolution, arma::ivec &rowsol, double &lapcost)

  Метод Джонкера-Волгенанта-Кастаньона (Jonker-Volgenant-Castanon) решения задачи о назначениях для плотных матриц
- void updateDual (int nc, arma::vec &d, arma::vec &v, arma::ivec &todo, int last, double min_)
- void updateAssignments (arma::ivec &lab, arma::ivec &y, arma::ivec &x, int j, int i0)
- int solveForOneL (std::vector< double > &cc_, const std::vector< int > &kk, const std::vector< int > &first, int l, int nc, arma::vec &d, arma::ivec &ok, arma::ivec &free, arma::vec &v, arma::ivec &lab, arma::ivec &todo, arma::ivec &y, arma::ivec &x, int td1, double resolution, double infValue, bool &fail)
- int JVCsparse (const std::vector< double > &cc, const std::vector< int > &kk, const std::vector< int > &first, TSearchParam sp, double infValue, double resolution, arma::ivec &rowsol, double &lapcost)

  Метод Джонкера-Волгенанта-Кастаньона (Jonker-Volgenant-Castanon) решения задачи о назначениях для разреженных матриц
- int JVCsparse (const Sparse::CMatrixCSR &csr, TSearchParam sp, double infValue, double resolution, arma::ivec &rowsol, double &lapcost)

  Метод Джонкера-Волгенанта-Кастаньона (Jonker-Volgenant-Castanon) решения задачи о назначениях для разреженных матриц
- bool lap_is_allowed (int i, int j)
- void Mack (const arma::mat &assigncost, int dim, TSearchParam sp, double infValue, double resolution, arma::ivec &rowsol, double &lapcost)

  Метод Мака решения задачи о назначениях
- void Murty_updateDual (int nc, arma::vec &d, arma::vec &v, arma::ivec &todo, int last, double min_)
- void Murty_updateAssignments (arma::ivec &lab, arma::ivec &y, arma::ivec &x, int j, int i0)
- int Murty_solveForOneL (std::vector< double > &cc_, const std::vector< int > &kk, const std::vector< int > &first, int l, int nc, arma::vec &d, arma::ivec &ok, arma::ivec &free, arma::vec &v, arma::ivec &lab, arma::ivec &todo, arma::ivec &y, arma::ivec &x, int td1, double resolution, double infValue, bool &fail)
- int Murty_JVCsparse (const std::vector< double > &cc, const std::vector< int > &kk, const std::vector< int > &first, TSearchParam sp, double infValue, double resolution, arma::ivec &rowsol, double &lapcost, arma::vec *u_init=nullptr, arma::vec *v_init=nullptr, arma::vec *u_out=nullptr, arma::vec *v_out=nullptr)

  Метод Джонкера-Волгенанта-Кастаньона (Jonker-Volgenant-Castanon) решения задачи о назначениях для разреженных матриц
- int Murty_JVCsparse (const Sparse::CMatrixCSR &csr, TSearchParam sp, double infValue, double resolution, arma::ivec &rowsol, double &lapcost, arma::vec *u_init=nullptr, arma::vec *v_init=nullptr, arma::vec *u_out=nullptr, arma::vec *v_out=nullptr)

  Метод Джонкера-Волгенанта-Кастаньона (Jonker-Volgenant-Castanon) решения задачи о назначениях для разреженных матриц
- void SequentalExtremum (const arma::mat &assigncost, TSearchParam sp, double infValue, double resolution, arma::ivec &rowsol, double &lapcost)

  Процедура последовательного поиска экстремума по строкам/столбцам матрицы ценностей
- void SequentalExtremum (const Sparse::CMatrixCOO &assigncost, TSearchParam sp, double infValue, double resolution, arma::ivec &rowsol, double &lapcost)

  Процедура последовательного поиска экстремума по строкам/столбцам матрицы ценностей

Переменные

- thread_local const LapConstraints * g_lap_constraints = nullptr

6.3.1 Подробное описание

Решение задачи о назначениях

6.3.2 Перечисления

6.3.2.1 TSearchParam

enum SPML::LAP::TSearchParam

Критерий поиска - минимум/максимум для задачи о назначениях

Элементы перечислений

SP_Min	
SP_Max	

См. определение в файле searchparam.hpp строка 22

6.3.3 Функции

6.3.3.1 Hungarian()

```
void SPML::LAP::Hungarian (
                const arma::mat & assigncost,
                int dim,
                TSearchParam sp,
                double infValue,
                double resolution,
                arma::ivec & rowsol,
                double & lapcost )
```

Венгерский метод решения задачи о назначениях (Метод Мункреса)

At last, we must create a function that enables us to jump around the different steps of the algorithm. The following code shows the main function of the algorithm. It defines also the important variables to be passed to the different steps.

Источник: https://github.com/RcppCore/rcpp-gallery/blob/gh-pages/src/2013-09-24-minimal-assignment.cpp

Аргументы

in	assigncost	- квадратная матрица ценности, размер [dim,dim]

Аргументы

in	dim	- порядок квадратной матрицы ценности и размерность результата res соответственно
in	sp	- критерий поиска (минимум/максимум)
in	infValue	- большое положительное число
in	resolution	- точность для сравнения двух вещественных чисел
out	rowsol	- результат задачи о назначениях, размерность [dim] (индекс макс/мин элемента в i-ой строке) "rowsol[i] = j" означает, что в i-ой строке выбран j-ый элемент
out	lapcost	- сумма назначенных элементов матрицы ценности assigncost

См. определение в файле hungarian.cpp строка 384

6.3.3.2 hungarian_augment_path()

```
void SPML::LAP::hungarian_augment_path (
                const int & path_count,
                arma::umat & indM,
                const arma::imat & path )
```

In addition we need a function to augment the path, one to clear the covers from rows and one to erase the primed zeros from the indicator matrix indM.

См. определение в файле hungarian.cpp строка 261

6.3.3.3 hungarian_clear_covers()

```
void SPML::LAP::hungarian_clear_covers (
                arma::ivec & rcov,
                arma::ivec & ccov )
```

См. определение в файле hungarian.cpp строка 273

6.3.3.4 hungarian_erase_primes()

```
void SPML::LAP::hungarian_erase_primes (
                arma::umat & indM,
                const unsigned int & N )
```

См. определение в файле hungarian.cpp строка 279

6.3.3.5 hungarian_find_noncovered_zero()

```
void SPML::LAP::hungarian_find_noncovered_zero (
                int & row,
                int & col,
                const arma::mat & cost,
                const arma::ivec & rcov,
                const arma::ivec & ccov,
                const unsigned int & N )
```

We cover a column by looking for 1s in the indicator matrix indM (See step 2 for assuring that these are indeed only starred zeros).

Step 4 finds noncovered zeros and primes them. If there are zeros in a row and none of them is starred, prime them. For this task we program a helper function to keep the code more readable and reusable. The helper function searches for noncovered zeros.

См. определение в файле hungarian.cpp строка 110

6.3.3.6 hungarian_find_prime_in_row()

```
void SPML::LAP::hungarian_find_prime_in_row (
                const int & row,
                int & col,
                const arma::umat & indM,
                const unsigned int & N )
```

Then we need a function to find a primed zero in a row. Note, that these tasks are easily performed by searching the indicator matrix indM.

См. определение в файле hungarian.cpp строка 247

6.3.3.7 hungarian_find_smallest()

```
void SPML::LAP::hungarian_find_smallest (
                double & minval,
                const arma::mat & cost,
                const arma::ivec & rcov,
                const arma::ivec & ccov,
                const unsigned int & N )
```

Recall, if step 4 was successfull in uncovering all columns and covering all rows with a primed zero, it then calls step 6. Step 6 takes the cover vectors rcov and ccov and looks in the uncovered region of the cost matrix for the smallest value. It then subtracts this value from each element in an uncovered column and adds it to each element in a covered row. After this transformation, the algorithm starts again at step 4. Our last helper function searches for the smallest value in the uncovered region of the cost matrix.

См. определение в файле hungarian.cpp строка 342

6.3.3.8 hungarian_find_star_in_col()

```
void SPML::LAP::hungarian_find_star_in_col (
                const int & col,
                int & row,
                const arma::umat & indM,
                const unsigned int & N )
```

Notice the rpath_0 and cpath_0 variables. These integer variables store the first vertex for an augmenting path in step 5. If zeros could be primed we go further to step 5.

Step 5 constructs a path beginning at an uncovered primed zero (this is actually graph theory - alternating and augmenting paths) and alternating between starred and primed zeros. This path is continued until a primed zero with no starred zero in its column is found. Then, all starred zeros in this path are unstarred and all primed zeros are starred. All primes in the indicator matrix are erased and all rows are uncovered. Then return to step 3 to cover again columns.

Step 5 needs several helper functions. First, we need a function to find starred zeros in columns.

См. определение в файле hungarian.cpp строка 232

6.3.3.9 hungarian_find_star_in_row()

```
void SPML::LAP::hungarian_find_star_in_row (
                const int & row,
                int & col,
                const arma::umat & indM,
                const unsigned int & N )
```

См. определение в файле hungarian.cpp строка 165

6.3.3.10 hungarian_star_in_row()

```
bool SPML::LAP::hungarian_star_in_row (
                int & row,
                const arma::umat & indM,
                const unsigned int & N )
```

We can detect noncovered zeros by checking if the cost matrix contains at row r and column c a zero and row and column are not covered yet, i.e. rcov(r) == 0, ccov(c) == 0. This loop breaks, if we have found our first uncovered zero or no uncovered zero at all.

In step 4, if no uncovered zero is found we go to step 6. If instead an uncovered zero has been found, we set the indicator matrix at its position to 2. We then have to search for a starred zero in the row with the uncovered zero, uncover the column with the starred zero and cover the row with the starred zero. To indicate a starred zero in a row and to find it we create again two helper functions.

См. определение в файле hungarian.cpp строка 153

6.3.3.11 hungarian_step_1()

```
void SPML::LAP::hungarian_step_1 (
                unsigned int & step,
                arma::mat & cost,
                const unsigned int & N )
```

См. определение в файле hungarian.cpp строка 18

6.3.3.12 hungarian_step_2()

```
void SPML::LAP::hungarian_step_2 (
                unsigned int & step,
                const arma::mat & cost,
                arma::umat & indM,
                arma::ivec & rcov,
                arma::ivec & ccov,
                const unsigned int & N )
```

Note, that we use references for all function arguments. As we have to switch between the steps of the algorithm continously, we always must be able to determine which step should be chosen next. Therefore we give a mutable unsigned integer step as an argument to each step function of the algorithm.

Inside the function we can easily access a whole row by Armadillo's row() method for matrices. In the second step, we then search for a zero in the modified cost matrix of step one.

См. определение в файле hungarian.cpp строка 39

6.3.3.13 hungarian_step_3()

```
void SPML::LAP::hungarian_step_3 (
                unsigned int & step,
                const arma::umat & indM,
                arma::ivec & ccov,
                const unsigned int & N )
```

Only the first zero in a row is taken. Then, the indicator matrix indM indicates this zero by setting the corresponding element at (r, c) to 1. A unique zero - the only or first one in a column and row - is called starred zero. In step 2 we find such a starred zero.

Note, that we use here Armadillo's element access via the method at(), which makes no bound checks and improves performance.

Note Bene: This code is thoroughly debugged - never do this for fresh written code!

In step 3 we cover each column with a starred zero. If already N columns are covered all starred zeros describe a complete assignment - so, go to step 7 and finish. Otherwise go to step 4.

См. определение в файле hungarian.cpp строка 77

6.3.3.14 hungarian_step_4()

```
void SPML::LAP::hungarian_step_4 (
                unsigned int & step,
                const arma::mat & cost,
                arma::umat & indM,
                arma::ivec & rcov,
                arma::ivec & ccov,
                int & rpath_0,
                int & cpath_0,
                const unsigned int & N )
```

We know that starred zeros are indicated by the indicator matrix containing an element equal to 1. Now, step 4.

См. определение в файле hungarian.cpp строка 180

6.3.3.15 hungarian_step_5()

```
void SPML::LAP::hungarian_step_5 (
                unsigned int & step,
                arma::umat & indM,
                arma::ivec & rcov,
                arma::ivec & ccov,
                arma::imat & path,
                int & rpath_0,
                int & cpath_0,
                const unsigned int & N )
```

The function to augment the path gets an integer matrix path of dimension $2 * N$ x 2. In it all vertices between rows and columns are stored row-wise. Now, we can set the complete step 5:

См. определение в файле hungarian.cpp строка 296

6.3.3.16 hungarian_step_6()

```
void SPML::LAP::hungarian_step_6 (
                unsigned int & step,
                arma::mat & cost,
                const arma::ivec & rcov,
                const arma::ivec & ccov,
                const unsigned int & N )
```

Step 6 looks as follows:

См. определение в файле hungarian.cpp строка 359

6.3.3.17 JVCdense()

```
void SPML::LAP::JVCdense (
        const arma::mat & assigncost,
        int dim,
        TSearchParam sp,
        double infValue,
        double resolution,
        arma::ivec & rowsol,
        double & lapcost )
```

Метод Джонкера-Волгенанта-Кастаньона (Jonker-Volgenant-Castanon) решения задачи о назначениях для плотных матриц

Источники: 1) "A Shortest Augmenting Path Algorithm for Dense and Sparse Linear Assignment Problems," Computing 38, 325-340, 1987 by R. Jonker and A. Volgenant, University of Amsterdam. 2) https://github.com/yongyanghz/LAPJV-algorithm-c 3) https://www.mathworks.com/matlabcentral/fileexchange/26836 lapjv-jonker-volgenant-algorithm-for-linear-assignment-problem-v3-0

Прим.

- Оригинальный код R. Jonker and A. Volgenant [1] для целых чисел адаптирован под вещественные

- Метод подразделен на 4 процедуры в соответствии с модификацией Castanon:
  - COLUMN REDUCTION
  - REDUCTION TRANSFER
  - AUGMENTING ROW REDUCTION - аукцион
  - AUGMENT SOLUTION FOR EACH FREE ROW на основе алгоритма Dijkstra
- Правки из [3] касающиеся точности сравнения вещественных чисел

Аргументы

in	assigncost	- квадратная матрица ценности, размер [dim,dim]
in	dim	- порядок квадратной матрицы ценности и размерность результата res соответственно
in	sp	- критерий поиска (минимум/максимум)
in	infValue	- большое положительное число
in	resolution	- точность для сравнения двух вещественных чисел
out	rowsol	- результат задачи о назначениях, размерность [dim] (индекс макс/мин элемента в i-ой строке) "rowsol[i] = j" означает, что в i-ой строке выбран j-ый элемент
out	lapcost	- сумма назначенных элементов матрицы ценности assigncost

См. определение в файле jvc_dense.cpp строка 18

6.3.3.18 JVCsparse() [1/2]

```
int SPML::LAP::JVCsparse (
        const Sparse::CMatrixCSR & csr,
```

TSearchParam sp,
double infValue,
double resolution,
arma::ivec & rowsol,
double & lapcost)

Метод Джонкера-Волгенанта-Кастаньона (Jonker-Volgenant-Castanon) решения задачи о назначениях для разреженных матриц

Аргументы

in	csr	- матрица в CSR формате (Compressed Sparse Row Yale format)
in	sp	- критерий поиска (минимум/максимум)
in	infValue	- большое положительное число
in	resolution	- точность для сравнения двух вещественных чисел
out	rowsol	- результат задачи о назначениях, размерность [dim] (индекс макс/мин элемента в i-ой строке) "rowsol[i] = j" означает, что в i-ой строке выбран j-ый элемент
out	lapcost	- сумма назначенных элементов матрицы ценности assigncost

Возвращает

0 - OK, 1 - fail

См. определение в файле jvc_sparse.cpp строка 388

6.3.3.19 JVCsparse() [2/2]

int SPML::LAP::JVCsparse (
const std::vector< double > & cc,
const std::vector< int > & kk,
const std::vector< int > & first,
TSearchParam sp,
double infValue,
double resolution,
arma::ivec & rowsol,
double & lapcost)

Метод Джонкера-Волгенанта-Кастаньона (Jonker-Volgenant-Castanon) решения задачи о назначениях для разреженных матриц

Аргументы

in	cc	- вектор ненулевых элементов матрицы
in	kk	- вектор индексов колонок ненулевых элементов, размер равен количеству ненулевых элементов
in	first	- вектор начальных смещений в векторе cc
in	sp	- критерий поиска (минимум/максимум)
in	infValue	- большое положительное число
in	resolution	- точность для сравнения двух вещественных чисел
out	rowsol	- результат задачи о назначениях, размерность [dim] (индекс макс/мин элемента в i-ой строке) "rowsol[i] = j" означает, что в i-ой строке выбран j-ый элемент
out	lapcost	- сумма назначенных элементов матрицы ценности assigncost

Возвращает

0 - OK, 1 - fail

См. определение в файле jvc_sparse.cpp строка 147

6.3.3.20 lap_is_allowed()

```
bool SPML::LAP::lap_is_allowed (
                int i,
                int j )   [inline]
```

См. определение в файле lap_constraints.hpp строка 42

6.3.3.21 Mack()

```
void SPML::LAP::Mack (
                const arma::mat & assigncost,
                int dim,
                TSearchParam sp,
                double infValue,
                double resolution,
                arma::ivec & rowsol,
                double & lapcost )
```

Метод Мака решения задачи о назначениях

Источник: Банди Б. Основы линейного программирования: Пер. с англ. - М.:Радио м связь, 1989, стр 113-123

Аргументы

in	assigncost	- квадратная матрица ценности, размер [dim,dim]
in	dim	- порядок квадратной матрицы ценности и размерность результата res соответственно
in	sp	- критерий поиска (минимум/максимум)
in	infValue	- большое положительное число
in	resolution	- точность для сравнения двух вещественных чисел
out	rowsol	- результат задачи о назначениях, размерность [dim] (индекс макс/мин элемента в i-ой строке) "rowsol[i] = j" означает, что в i-ой строке выбран j-ый элемент
out	lapcost	- сумма назначенных элементов матрицы ценности assigncost

См. определение в файле mack.cpp строка 18

6.3.3.22 Murty_JVCsparse() [1/2]

```
int SPML::LAP::Murty_JVCsparse (
        const Sparse::CMatrixCSR & csr,
        TSearchParam sp,
        double infValue,
        double resolution,
        arma::ivec & rowsol,
        double & lapcost,
        arma::vec * u_init = nullptr,
        arma::vec * v_init = nullptr,
        arma::vec * u_out = nullptr,
        arma::vec * v_out = nullptr )
```

Метод Джонкера-Волгенанта-Кастаньона (Jonker-Volgenant-Castanon) решения задачи о назначениях для разреженных матриц

Аргументы

in	csr	- матрица в CSR формате (Compressed Sparse Row Yale format)
in	sp	- критерий поиска (минимум/максимум)
in	infValue	- большое положительное число
in	resolution	- точность для сравнения двух вещественных чисел
out	rowsol	- результат задачи о назначениях, размерность [dim] (индекс макс/мин элемента в i-ой строке) "rowsol[i] = j" означает, что в i-ой строке выбран j-ый элемент
out	lapcost	- сумма назначенных элементов матрицы ценности assigncost
	u_init	- начальная двойственная переменная строки
	v_init	- начальная двойственная переменная столбца
	u_out	- конечная двойственная переменная строки
	v_out	- конечная двойственная переменная столбца

Возвращает

0 - OK, 1 - fail

См. определение в файле murty_jvc_sparse.cpp строка 430

6.3.3.23 Murty_JVCsparse() [2/2]

```
int SPML::LAP::Murty_JVCsparse (
        const std::vector< double > & cc,
        const std::vector< int > & kk,
        const std::vector< int > & first,
        TSearchParam sp,
        double infValue,
        double resolution,
        arma::ivec & rowsol,
        double & lapcost,
        arma::vec * u_init = nullptr,
```

```
arma::vec * v_init = nullptr,
arma::vec * u_out = nullptr,
arma::vec * v_out = nullptr )
```

Метод Джонкера-Волгенанта-Кастаньона (Jonker-Volgenant-Castanon) решения задачи о назначениях для разреженных матриц

Аргументы

in	cc	- вектор ненулевых элементов матрицы
in	kk	- вектор индексов колонок ненулевых элементов, размер равен количеству ненулевых элементов
in	first	- вектор начальных смещений в векторе cc
in	sp	- критерий поиска (минимум/максимум)
in	infValue	- большое положительное число
in	resolution	- точность для сравнения двух вещественных чисел
out	rowsol	- результат задачи о назначениях, размерность [dim] (индекс макс/мин элемента в i-ой строке) "rowsol[i] = j" означает, что в i-ой строке выбран j-ый элемент
out	lapcost	- сумма назначенных элементов матрицы ценности assigncost
	u_init	- начальная двойственная переменная строки
	v_init	- начальная двойственная переменная столбца
	u_out	- конечная двойственная переменная строки
	v_out	- конечная двойственная переменная столбца

Возвращает

0 - OK, 1 - fail

См. определение в файле murty_jvc_sparse.cpp строка 158

6.3.3.24 Murty_solveForOneL()

```
int SPML::LAP::Murty_solveForOneL (
        std::vector< double > & cc_,
        const std::vector< int > & kk,
        const std::vector< int > & first,
        int l,
        int nc,
        arma::vec & d,
        arma::ivec & ok,
        arma::ivec & free,
        arma::vec & v,
        arma::ivec & lab,
        arma::ivec & todo,
        arma::ivec & y,
        arma::ivec & x,
        int td1,
        double resolution,
        double infValue,
        bool & fail )
```

См. определение в файле murty_jvc_sparse.cpp строка 49

6.3.3.25 Murty_updateAssignments()

```
void SPML::LAP::Murty_updateAssignments (
                arma::ivec & lab,
                arma::ivec & y,
                arma::ivec & x,
                int j,
                int i0 )
```

См. определение в файле murty_jvc_sparse.cpp строка 32

6.3.3.26 Murty_updateDual()

```
void SPML::LAP::Murty_updateDual (
                int nc,
                arma::vec & d,
                arma::vec & v,
                arma::ivec & todo,
                int last,
                double min_ )
```

См. определение в файле murty_jvc_sparse.cpp строка 23

6.3.3.27 SequentalExtremum() [1/2]

```
void SPML::LAP::SequentalExtremum (
                const arma::mat & assigncost,
                TSearchParam sp,
                double infValue,
                double resolution,
                arma::ivec & rowsol,
                double & lapcost )
```

Процедура последовательного поиска экстремума по строкам/столбцам матрицы ценностей

Неоптимальная процедура, сумма назначений будет неоптимальна, поскольку строки/столбцы с найденным экстремумом исключаются из анализа.

Аргументы

in	assigncost	- матрица ценности
in	sp	- критерий поиска (минимум/максимум)
in	infValue	- большое положительное число
in	resolution	- точность для сравнения двух вещественных чисел
out	rowsol	- результат задачи о назначениях, размерность [dim] (индекс макс/мин элемента в i-ой строке) "rowsol[i] = j" означает, что в i-ой строке выбран j-ый элемент
out	lapcost	- сумма назначенных элементов матрицы ценности assigncost

См. определение в файле seqextr.cpp строка 18

6.3.3.28 SequentalExtremum() [2/2]

```
void SPML::LAP::SequentalExtremum (
            const Sparse::CMatrixCOO & assigncost,
            TSearchParam sp,
            double infValue,
            double resolution,
            arma::ivec & rowsol,
            double & lapcost )
```

Процедура последовательного поиска экстремума по строкам/столбцам матрицы ценностей

Неоптимальная процедура, сумма назначений будет неоптимальна, поскольку строки/столбцы с найденным экстремумом исключаются из анализа.

Аргументы

in	assigncost	- матрица ценности в сжатом COO формате
in	sp	- критерий поиска (минимум/максимум)
in	infValue	- большое положительное число
in	resolution	- точность для сравнения двух вещественных чисел
out	rowsol	- результат задачи о назначениях, размерность [dim] (индекс макс/мин элемента в i-ой строке) "rowsol[i] = j" означает, что в i-ой строке выбран j-ый элемент
out	lapcost	- сумма назначенных элементов матрицы ценности assigncost

См. определение в файле seqextr.cpp строка 84

6.3.3.29 solveForOneL()

```
int SPML::LAP::solveForOneL (
            std::vector< double > & cc_,
            const std::vector< int > & kk,
            const std::vector< int > & first,
            int l,
            int nc,
            arma::vec & d,
            arma::ivec & ok,
            arma::ivec & free,
            arma::vec & v,
            arma::ivec & lab,
            arma::ivec & todo,
            arma::ivec & y,
            arma::ivec & x,
            int td1,
            double resolution,
```

```
                double infValue,
                bool & fail )
```

См. определение в файле jvc_sparse.cpp строка 44

6.3.3.30 updateAssignments()

```
void SPML::LAP::updateAssignments (
                arma::ivec & lab,
                arma::ivec & y,
                arma::ivec & x,
                int j,
                int i0 )
```

См. определение в файле jvc_sparse.cpp строка 27

6.3.3.31 updateDual()

```
void SPML::LAP::updateDual (
                int nc,
                arma::vec & d,
                arma::vec & v,
                arma::ivec & todo,
                int last,
                double min_ )
```

См. определение в файле jvc_sparse.cpp строка 18

6.3.4 Переменные

6.3.4.1 g_lap_constraints

```
thread_local const LapConstraints * SPML::LAP::g_lap_constraints = nullptr
```

См. определение в файле murty_jvc_sparse.cpp строка 20

6.4 Пространство имен SPML::Sparse

Решение задачи о назначениях

Классы

- struct CKeyCOO

  Ключ элемента Aij матрицы A в COO формате (Coordinate list)
- struct CMatrixCOO

  Структура хранения матрицы в координатном COO формате (Coordinate list)
- struct CMatrixCSC

  Структура хранения матрицы в CSC формате (по столбцам) (Compressed Sparse Column Yale format)
- struct CMatrixCSR

  Структура хранения матрицы в CSR формате (построчно) (Compressed Sparse Row Yale format)

Функции

- void MatrixDenseToCOO (const arma::mat &A, std::vector< double > &coo_val, std::vector< int > &coo_row, std::vector< int > &coo_col)

  Преобразование плотной матрицы в COO формат (Coordinated list)
- void MatrixDenseToCOO (const arma::mat &A, CMatrixCOO &COO)

  Преобразование плотной матрицы в COO формат (Coordinated list)
- void MatrixCOOtoDense (const std::vector< double > &coo_val, const std::vector< int > &coo↩_row, const std::vector< int > &coo_col, arma::mat &A)

  Преобразование матрицы из COO формата (Coordinated list) в плотную матрицу
- void MatrixCOOtoDense (const CMatrixCOO &COO, arma::mat &A)

  Преобразование матрицы из COO формата (Coordinated list) в плотную матрицу
- void MatrixDenseToCSR (const arma::mat &A, std::vector< double > &csr_val, std::vector< int > &csr_kk, std::vector< int > &csr_first)

  Преобразование плотной матрицы в CSR формат (Compressed Sparse Row Yale format)
- void MatrixDenseToCSR (const arma::mat &A, CMatrixCSR &CSR)

  Преобразование плотной матрицы в CSR формат (Compressed Sparse Row Yale format)
- void MatrixCSRtoDense (const std::vector< double > &csr_val, const std::vector< int > &csr_kk, const std::vector< int > &csr_first, arma::mat &A)

  Преобразование матрицы из CSR формата (Compressed Sparse Row Yale format) в плотную матрицу
- void MatrixCSRtoDense (const CMatrixCSR &CSR, arma::mat &A)

  Преобразование матрицы из CSR формата (Compressed Sparse Row Yale format) в плотную матрицу
- void MatrixDenseToCSC (const arma::mat &A, std::vector< double > &csc_val, std::vector< int > &csc_kk, std::vector< int > &csc_first)

  Преобразование плотной матрицы в CSC формат (Compressed Sparse Column Yale format)
- void MatrixDenseToCSC (const arma::mat &A, CMatrixCSC &CSC)

  Преобразование плотной матрицы в CSC формат (Compressed Sparse Column Yale format)
- void MatrixCSCtoDense (const std::vector< double > &csc_val, const std::vector< int > &csc_kk, const std::vector< int > &csc_first, arma::mat &A)

  Преобразование матрицы из CSC формата (Compressed Sparse Column Yale format) в плотную матрицу
- void MatrixCSCtoDense (const CMatrixCSC &CSC, arma::mat &A)

  Преобразование матрицы из CSC формата (Compressed Sparse Column Yale format) в плотную матрицу
- void MatrixCOOtoCSR (const std::vector< double > &coo_val, const std::vector< int > &coo↩_row, const std::vector< int > &coo_col, std::vector< double > &csr_val, std::vector< int > &csr_kk, std::vector< int > &csr_first, bool sorted=false)

  Преобразование матрицы в COO формате (Coordinate list) в CSR формат (Compressed Sparse Row Yale format)
- void MatrixCOOtoCSR (const CMatrixCOO &COO, CMatrixCSR &CSR, bool sorted=false)

Преобразование матрицы в COO формате (Coordinate list) в CSR формат (Compressed Sparse Row Yale format)

- void MatrixCOOtoCSC (const std::vector< double > &coo_val, const std::vector< int > &coo_row, const std::vector< int > &coo_col, std::vector< double > &csc_val, std::vector< int > &csc_kk, std::vector< int > &csc_first, bool sorted=false)

  Преобразование матрицы в COO формате (Coordinate list) в CSC формат (Compressed Sparse Column Yale format)
- void MatrixCOOtoCSC (const CMatrixCOO &COO, CMatrixCSC &CSC, bool sorted=false)

  Преобразование матрицы в COO формате (Coordinate list) в CSC формат (Compressed Sparse Column Yale format)

6.4.1 Подробное описание

Решение задачи о назначениях

Разреженные матрицы

6.4.2 Функции

6.4.2.1 MatrixCOOtoCSC() [1/2]

```
void SPML::Sparse::MatrixCOOtoCSC (
            const CMatrixCOO & COO,
            CMatrixCSC & CSC,
            bool sorted = false )
```

Преобразование матрицы в COO формате (Coordinate list) в CSC формат (Compressed Sparse Column Yale format)

Аргументы

in	COO	- матрица в COO формате (Coordinate list)
out	CSC	- матрица в CSR формате (Compressed Sparse Column Yale format)
in	sorted	- признак отсортированности COO входа по столбцам (даёт ускорение в ~2 раза)

См. определение в файле sparse.cpp строка 328

6.4.2.2 MatrixCOOtoCSC() [2/2]

```
void SPML::Sparse::MatrixCOOtoCSC (
            const std::vector< double > & coo_val,
            const std::vector< int > & coo_row,
            const std::vector< int > & coo_col,
            std::vector< double > & csc_val,
```

```
std::vector< int > & csc_kk,
std::vector< int > & csc_first,
bool sorted = false )
```

Преобразование матрицы в COO формате (Coordinate list) в CSC формат (Compressed Sparse Column Yale format)

Аргументы

in	coo_val	- вектор ненулевых элементов матрицы A, размер равен количеству ненулевых элементов nnz
in	coo_row	- индексы строк ненулевых элементов, размер nnz
in	coo_col	- индексы столбцов ненулевых элементов, размер nnz
in	csc_val	- вектор ненулевых элементов матрицы A, размер равен количеству ненулевых элементов nnz
in	csc_kk	- вектор индексов строк ненулевых элементов, размер равен количеству ненулевых элементов nnz
in	csc_first	- вектор начальных смещений в векторе CSR, размер m+1
in	sorted	- признак отсортированности COO входа по столбцам (даёт ускорение в ∼2 раза)

См. определение в файле sparse.cpp строка 256

6.4.2.3 MatrixCOOtoCSR() [1/2]

```
void SPML::Sparse::MatrixCOOtoCSR (
        const CMatrixCOO & COO,
        CMatrixCSR & CSR,
        bool sorted = false )
```

Преобразование матрицы в COO формате (Coordinate list) в CSR формат (Compressed Sparse Row Yale format)

Аргументы

in	COO	- матрица в COO формате (Coordinate list)
out	CSR	- матрица в CSR формате (Compressed Sparse Row Yale format)
in	sorted	- признак отсортированности COO входа по строкам (даёт ускорение в ∼2 раза)

См. определение в файле sparse.cpp строка 251

6.4.2.4 MatrixCOOtoCSR() [2/2]

```
void SPML::Sparse::MatrixCOOtoCSR (
        const std::vector< double > & coo_val,
```

```
const std::vector< int > & coo_row,
const std::vector< int > & coo_col,
std::vector< double > & csr_val,
std::vector< int > & csr_kk,
std::vector< int > & csr_first,
bool sorted = false )
```

Преобразование матрицы в COO формате (Coordinate list) в CSR формат (Compressed Sparse Row Yale format)

Аргументы

in	coo_val	- вектор ненулевых элементов матрицы A, размер равен количеству ненулевых элементов nnz
in	coo_row	- индексы строк ненулевых элементов, размер nnz
in	coo_col	- индексы столбцов ненулевых элементов, размер nnz
in	csr_val	- вектор ненулевых элементов матрицы A, размер равен количеству ненулевых элементов nnz
in	csr_kk	- вектор индексов колонок ненулевых элементов, размер равен количеству ненулевых элементов nnz
in	csr_first	- вектор начальных смещений в векторе CSR, размер n+1
in	sorted	- признак отсортированности COO входа по строкам (даёт ускорение в ~2 раза)

См. определение в файле sparse.cpp строка 179

6.4.2.5 MatrixCOOtoDense() [1/2]

```
void SPML::Sparse::MatrixCOOtoDense (
        const CMatrixCOO & COO,
        arma::mat & A )
```

Преобразование матрицы из COO формата (Coordinated list) в плотную матрицу

Аргументы

in	COO	- структура хранения матрицы в COO формате (Coordinate list)
out	A	- плотная матрица, размер [n,m] (n - число строк, m - число столбцов)

См. определение в файле sparse.cpp строка 60

6.4.2.6 MatrixCOOtoDense() [2/2]

```
void SPML::Sparse::MatrixCOOtoDense (
        const std::vector< double > & coo_val,
        const std::vector< int > & coo_row,
```

const std::vector< int > & coo_col,
arma::mat & A)

Преобразование матрицы из COO формата (Coordinated list) в плотную матрицу

Аргументы

in	coo_val	- вектор ненулевых элементов матрицы A, размер равен количеству ненулевых элементов nnz
in	coo_row	- индексы строк ненулевых элементов, размер nnz
in	coo_col	- индексы столбцов ненулевых элементов, размер nnz
out	A	- плотная матрица, размер [n,m] (n - число строк, m - число столбцов)

См. определение в файле sparse.cpp строка 48

6.4.2.7 MatrixCSCtoDense() [1/2]

```
void SPML::Sparse::MatrixCSCtoDense (
                const CMatrixCSC & CSC,
                arma::mat & A )
```

Преобразование матрицы из CSC формата (Compressed Sparse Column Yale format) в плотную матрицу

Аргументы

in	CSC	- структура хранения матрицы в CSC формате (Compressed Sparse Column Yale format)
out	A	- плотная матрица, размер [n,m] (n - число строк, m - число столбцов)

См. определение в файле sparse.cpp строка 172

6.4.2.8 MatrixCSCtoDense() [2/2]

```
void SPML::Sparse::MatrixCSCtoDense (
                const std::vector< double > & csc_val,
                const std::vector< int > & csc_kk,
                const std::vector< int > & csc_first,
                arma::mat & A )
```

Преобразование матрицы из CSC формата (Compressed Sparse Column Yale format) в плотную матрицу

Аргументы

in	csc_val	- вектор ненулевых элементов матрицы A, размер равен количеству ненулевых элементов nnz
in	csc_kk	- вектор индексов строк ненулевых элементов, размер равен количеству ненулевых элементов nnz
in	csc_first	- вектор начальных смещений в векторе CSR, размер m+1
out	A	- плотная матрица, размер [n,m] (n - число строк, m - число столбцов)

См. определение в файле sparse.cpp строка 156

6.4.2.9 MatrixCSRtoDense() [1/2]

```
void SPML::Sparse::MatrixCSRtoDense (
                const CMatrixCSR & CSR,
                arma::mat & A )
```

Преобразование матрицы из CSR формата (Compressed Sparse Row Yale format) в плотную матрицу

Аргументы

in	CSR	- структура хранения матрицы в CSR формате (Compressed Sparse Row Yale format)
out	A	- плотная матрица, размер [n,m] (n - число строк, m - число столбцов)

См. определение в файле sparse.cpp строка 116

6.4.2.10 MatrixCSRtoDense() [2/2]

```
void SPML::Sparse::MatrixCSRtoDense (
                const std::vector< double > & csr_val,
                const std::vector< int > & csr_kk,
                const std::vector< int > & csr_first,
                arma::mat & A )
```

Преобразование матрицы из CSR формата (Compressed Sparse Row Yale format) в плотную матрицу

Аргументы

in	csr_val	- вектор ненулевых элементов матрицы A, размер равен количеству ненулевых элементов nnz
in	csr_kk	- вектор индексов колонок ненулевых элементов, размер равен количеству ненулевых элементов nnz
in	csr_first	- вектор начальных смещений в векторе CSR, размер n+1
out	A	- плотная матрица, размер [n,m] (n - число строк, m - число столбцов)

См. определение в файле sparse.cpp строка 100

6.4.2.11 MatrixDenseToCOO() [1/2]

```
void SPML::Sparse::MatrixDenseToCOO (
                const arma::mat & A,
                CMatrixCOO & COO )
```

Преобразование плотной матрицы в COO формат (Coordinated list)

Аргументы

in	A	- исходная матрица, размер [n,m] (n - число строк, m - число столбцов)
out	COO	- структура хранения матрицы в COO формате (Coordinate list)

См. определение в файле sparse.cpp строка 43

6.4.2.12 MatrixDenseToCOO() [2/2]

```
void SPML::Sparse::MatrixDenseToCOO (
        const arma::mat & A,
        std::vector< double > & coo_val,
        std::vector< int > & coo_row,
        std::vector< int > & coo_col )
```

Преобразование плотной матрицы в COO формат (Coordinated list)

Аргументы

in	A	- исходная матрица, размер [n,m] (n - число строк, m - число столбцов)
out	coo_val	- вектор ненулевых элементов матрицы A, размер nnz
out	coo_row	- индексы строк ненулевых элементов, размер nnz
out	coo_col	- индексы столбцов ненулевых элементов, размер nnz

См. определение в файле sparse.cpp строка 20

6.4.2.13 MatrixDenseToCSC() [1/2]

```
void SPML::Sparse::MatrixDenseToCSC (
        const arma::mat & A,
        CMatrixCSC & CSC )
```

Преобразование плотной матрицы в CSC формат (Compressed Sparse Column Yale format)

Аргументы

in	A	- исходная матрица, размер [n,m] (n - число строк, m - число столбцов)
out	CSC	- структура хранения матрицы в CSC формате (Compressed Sparse Column Yale format)

См. определение в файле sparse.cpp строка 151

6.4.2.14 MatrixDenseToCSC() [2/2]

void SPML::Sparse::MatrixDenseToCSC (
const arma::mat & A,
std::vector< double > & csc_val,
std::vector< int > & csc_kk,
std::vector< int > & csc_first)

Преобразование плотной матрицы в CSC формат (Compressed Sparse Column Yale format)

Данный способ хранения эффективен, если кол-во ненулевых элементов NNZ<(m∗(n-1)-1)/2

Аргументы

in	A	- исходная матрица, размер [n,m] (n - число строк, m - число столбцов)
out	csc_val	- вектор ненулевых элементов матрицы A, размер равен количеству ненулевых элементов nnz
out	csc_kk	- вектор индексов строк ненулевых элементов, размер равен количеству ненулевых элементов nnz
out	csc_first	- вектор начальных смещений в векторе CSR, размер m+1

См. определение в файле sparse.cpp строка 123

6.4.2.15 MatrixDenseToCSR() [1/2]

void SPML::Sparse::MatrixDenseToCSR (
const arma::mat & A,
CMatrixCSR & CSR)

Преобразование плотной матрицы в CSR формат (Compressed Sparse Row Yale format)

Аргументы

in	A	- исходная матрица, размер [n,m] (n - число строк, m - число столбцов)
out	CSR	- структура хранения матрицы в CSR формате (Compressed Sparse Row Yale format)

См. определение в файле sparse.cpp строка 95

6.4.2.16 MatrixDenseToCSR() [2/2]

void SPML::Sparse::MatrixDenseToCSR (
const arma::mat & A,
std::vector< double > & csr_val,
std::vector< int > & csr_kk,
std::vector< int > & csr_first)

Преобразование плотной матрицы в CSR формат (Compressed Sparse Row Yale format)

Данный способ хранения эффективен, если кол-во ненулевых элементов NNZ<(m∗(n-1)-1)/2

Аргументы

in	A	- исходная матрица, размер [n,m] (n - число строк, m - число столбцов)
out	csr_val	- вектор ненулевых элементов матрицы A, размер равен количеству ненулевых элементов nnz
out	csr_kk	- вектор индексов колонок ненулевых элементов, размер равен количеству ненулевых элементов nnz
out	csr_first	- вектор начальных смещений в векторе CSR, размер n+1

См. определение в файле sparse.cpp строка 67

Глава 7

Классы

7.1 Структура SPML::Sparse::CKeyCOO

Ключ элемента Aij матрицы A в COO формате (Coordinate list)

#include <sparse.hpp>

Открытые члены

- int i () const
  i - индекс строки
- int j () const
  j - индекс столбца
- CKeyCOO ()
  Конструктор по умолчанию
- CKeyCOO (int i, int j)
  Параметрический конструктор
- bool operator< (CKeyCOO const &other) const

Закрытые данные

- int i_
  Индекс строки
- int j_
  Индекс столбца

7.1.1 Подробное описание

Ключ элемента Aij матрицы A в COO формате (Coordinate list)

Внимание

Оператор < перегружен для случая построчного хранения

См. определение в файле sparse.hpp строка 241

7.1.2 Конструктор(ы)

7.1.2.1 CKeyCOO() [1/2]

SPML::Sparse::CKeyCOO::CKeyCOO ( ) [inline]

Конструктор по умолчанию

См. определение в файле sparse.hpp строка 257

7.1.2.2 CKeyCOO() [2/2]

SPML::Sparse::CKeyCOO::CKeyCOO (
int i,
int j) [inline]

Параметрический конструктор

Аргументы

in	i	- индекс строки
in	j	- индекс столбца

См. определение в файле sparse.hpp строка 264

7.1.3 Методы

7.1.3.1 i()

int SPML::Sparse::CKeyCOO::i ( ) const [inline]

i - индекс строки

См. определение в файле sparse.hpp строка 247

7.1.3.2 j()

int SPML::Sparse::CKeyCOO::j ( ) const [inline]

j - индекс столбца

См. определение в файле sparse.hpp строка 252

7.1.3.3 operator<()

bool SPML::Sparse::CKeyCOO::operator< (
CKeyCOO const & other) const [inline]

См. определение в файле sparse.hpp строка 266

7.1.4 Данные класса

7.1.4.1 i_

int SPML::Sparse::CKeyCOO::i_ [private]

Индекс строки

См. определение в файле sparse.hpp строка 275

7.1.4.2 j_

int SPML::Sparse::CKeyCOO::j_ [private]

Индекс столбца

См. определение в файле sparse.hpp строка 276

Объявления и описания членов структуры находятся в файле:

- sparse.hpp

7.2 Структура SPML::Sparse::CMatrixCOO

Структура хранения матрицы в координатном COO формате (Coordinate list)

#include <sparse.hpp>

Открытые атрибуты

- std::vector< double > coo_val
  Вектор ненулевых элементов матрицы A[n,m] (n - число строк, m - число столбцов), размер nnz.
- std::vector< int > coo_row
  Индексы строк ненулевых элементов
- std::vector< int > coo_col
  Индексы столбцов ненулевых элементов

7.2.1 Подробное описание

Структура хранения матрицы в координатном COO формате (Coordinate list)

Матрица A[n,m] (n - число строк, m - число столбцов)

См. определение в файле sparse.hpp строка 33

7.2.2 Данные класса

7.2.2.1 coo_col

std::vector<int> SPML::Sparse::CMatrixCOO::coo_col

Индексы столбцов ненулевых элементов

См. определение в файле sparse.hpp строка 37

7.2.2.2 coo_row

std::vector<int> SPML::Sparse::CMatrixCOO::coo_row

Индексы строк ненулевых элементов

См. определение в файле sparse.hpp строка 36

7.2.2.3 coo_val

std::vector<double> SPML::Sparse::CMatrixCOO::coo_val

Вектор ненулевых элементов матрицы A[n,m] (n - число строк, m - число столбцов), размер nnz.

См. определение в файле sparse.hpp строка 35

Объявления и описания членов структуры находятся в файле:

- sparse.hpp

7.3 Структура SPML::Sparse::CMatrixCSC

Структура хранения матрицы в CSC формате (по столбцам) (Compressed Sparse Column Yale format)

#include <sparse.hpp>

Открытые атрибуты

- std::vector< double > csc_val

  Вектор ненулевых элементов матрицы A[n,m] (n - число строк, m - число столбцов), размер nnz.
- std::vector< int > csc_kk

  Вектор индексов колонок ненулевых элементов, размер равен количеству ненулевых элементов nnz.
- std::vector< int > csc_first

  вектор начальных смещений в векторе CSC, размер m+1

7.3.1 Подробное описание

Структура хранения матрицы в CSC формате (по столбцам) (Compressed Sparse Column Yale format)

Матрица A[n,m] (n - число строк, m - число столбцов)

См. определение в файле sparse.hpp строка 77

7.3.2 Данные класса

7.3.2.1 csc_first

std::vector<int> SPML::Sparse::CMatrixCSC::csc_first

вектор начальных смещений в векторе CSC, размер m+1

См. определение в файле sparse.hpp строка 81

7.3.2.2 csc_kk

std::vector<int> SPML::Sparse::CMatrixCSC::csc_kk

Вектор индексов колонок ненулевых элементов, размер равен количеству ненулевых элементов nnz.

См. определение в файле sparse.hpp строка 80

7.3.2.3 csc_val

std::vector<double> SPML::Sparse::CMatrixCSC::csc_val

Вектор ненулевых элементов матрицы A[n,m] (n - число строк, m - число столбцов), размер nnz.

См. определение в файле sparse.hpp строка 79

Объявления и описания членов структуры находятся в файле:

- sparse.hpp

7.4 Структура SPML::Sparse::CMatrixCSR

Структура хранения матрицы в CSR формате (построчно) (Compressed Sparse Row Yale format)

#include <sparse.hpp>

Открытые члены

- int n_rows () const
  Число строк
- int n_cols () const
  Число столбцов

Открытые атрибуты

- std::vector< double > csr_val
  Вектор ненулевых элементов матрицы A[n,m] (n - число строк, m - число столбцов), размер nnz.
- std::vector< int > csr_kk
  Вектор индексов колонок ненулевых элементов, размер равен количеству ненулевых элементов nnz.
- std::vector< int > csr_first
  Вектор начальных смещений в векторе CSR, размер n+1.

7.4.1 Подробное описание

Структура хранения матрицы в CSR формате (построчно) (Compressed Sparse Row Yale format)

Матрица A[n,m] (n - число строк, m - число столбцов)

См. определение в файле sparse.hpp строка 44

7.4.2 Методы

7.4.2.1 n_cols()

int SPML::Sparse::CMatrixCSR::n_cols ( ) const [inline]

Число столбцов

См. определение в файле sparse.hpp строка 61

7.4.2.2 n_rows()

int SPML::Sparse::CMatrixCSR::n_rows ( ) const [inline]

Число строк

См. определение в файле sparse.hpp строка 53

7.4.3 Данные класса

7.4.3.1 csr_first

std::vector<int> SPML::Sparse::CMatrixCSR::csr_first

Вектор начальных смещений в векторе CSR, размер n+1.

См. определение в файле sparse.hpp строка 48

7.4.3.2 csr_kk

std::vector<int> SPML::Sparse::CMatrixCSR::csr_kk

Вектор индексов колонок ненулевых элементов, размер равен количеству ненулевых элементов nnz.

См. определение в файле sparse.hpp строка 47

7.4.3.3 csr_val

std::vector<double> SPML::Sparse::CMatrixCSR::csr_val

Вектор ненулевых элементов матрицы A[n,m] (n - число строк, m - число столбцов), размер nnz.

См. определение в файле sparse.hpp строка 46

Объявления и описания членов структуры находятся в файле:

- sparse.hpp

7.5 Структура SPML::LAP::LapConstraints

#include <lap_constraints.hpp>

Открытые члены

- bool isAllowed (int i, int j) const

Открытые атрибуты

- const int * fixed_col = nullptr
- const std::vector< std::pair< int, int > > * banned = nullptr

7.5.1 Подробное описание

См. определение в файле lap_constraints.hpp строка 22

7.5.2 Методы

7.5.2.1 isAllowed()

```
bool SPML::LAP::LapConstraints::isAllowed (
                int i,
                int j ) const    [inline]
```

См. определение в файле lap_constraints.hpp строка 27

7.5.3 Данные класса

7.5.3.1 banned

```
const std::vector<std::pair<int,int> >* SPML::LAP::LapConstraints::banned = nullptr
```

См. определение в файле lap_constraints.hpp строка 25

7.5.3.2 fixed_col

```
const int* SPML::LAP::LapConstraints::fixed_col = nullptr
```

См. определение в файле lap_constraints.hpp строка 24

Объявления и описания членов структуры находятся в файле:

- lap_constraints.hpp

7.6 Класс SPML::LAP::Murty

Класс решения K-best задачи методом Murty.

#include <murty.hpp>

Открытые члены

- Murty (const Sparse::CMatrixCSR &csr, TSearchParam sp, double inf, double res)
  Конструктор метода Murty.
- bool findNext (MurtySolution &out)
  Вызов очередного best-решения задачи

Закрытые члены

- MurtyNode solveNode (const MurtyNode *parent, int split_row)
  Решение узла

Закрытые данные

- const Sparse::CMatrixCSR & csr_
  Матрица в CSR формате (Compressed Sparse Row Yale format)
- TSearchParam sp_
  Критерий поиска (минимум/максимум)
- double inf_
  Большое положительное число
- double res_
  Точность для сравнения двух вещественных чисел
- bool initialized_
  Признак "тёплого" старта
- std::priority_queue< MurtyNode, std::vector< MurtyNode >, bool(*)(const MurtyNode &, const MurtyNode &) > pq_
  Очередь приоритетов

7.6.1 Подробное описание

Класс решения K-best задачи методом Murty.

См. определение в файле murty.hpp строка 55

7.6.2 Конструктор(ы)

7.6.2.1 Murty()

```
SPML::LAP::Murty::Murty (
            const Sparse::CMatrixCSR & csr,
            TSearchParam sp,
            double inf,
            double res )
```

Конструктор метода Murty.

Аргументы

csr	- матрица в CSR формате (Compressed Sparse Row Yale format)
sp	- критерий поиска (минимум/максимум)
inf	- большое положительное число
res	- точность для сравнения двух вещественных чисел

См. определение в файле murty.cpp строка 27

7.6.3 Методы

7.6.3.1 findNext()

bool SPML::LAP::Murty::findNext (
MurtySolution & out)

Вызов очередного best-решения задачи

Аргументы

out	- найденное решение (валидно, если return == true)

Возвращает

true - решение найдено, false - решения нет

См. определение в файле murty.cpp строка 97

7.6.3.2 solveNode()

MurtyNode SPML::LAP::Murty::solveNode (
const MurtyNode * parent,
int split_row) [private]

Решение узла

См. определение в файле murty.cpp строка 36

7.6.4 Данные класса

7.6.4.1 csr_

const Sparse::CMatrixCSR& SPML::LAP::Murty::csr_ [private]

Матрица в CSR формате (Compressed Sparse Row Yale format)

См. определение в файле murty.hpp строка 77

7.6.4.2 inf_

double SPML::LAP::Murty::inf_ [private]

Большое положительное число

См. определение в файле murty.hpp строка 79

7.6.4.3 initialized_

bool SPML::LAP::Murty::initialized_ [private]

Признак "тёплого" старта

См. определение в файле murty.hpp строка 81

7.6.4.4 pq_

std::priority_queue< MurtyNode, std::vector<MurtyNode>, bool(*)(const MurtyNode&, const MurtyNode&) > SPML↩
::LAP::Murty::pq_ [private]

Очередь приоритетов

См. определение в файле murty.hpp строка 86

7.6.4.5 res_

double SPML::LAP::Murty::res_ [private]

Точность для сравнения двух вещественных чисел

См. определение в файле murty.hpp строка 80

7.6.4.6 sp_

TSearchParam SPML::LAP::Murty::sp_ [private]

Критерий поиска (минимум/максимум)

См. определение в файле murty.hpp строка 78

Объявления и описания членов классов находятся в файлах:

- murty.hpp
- murty.cpp

7.7 Структура SPML::LAP::MurtyNode

Узел решения

#include <murty.hpp>

Открытые атрибуты

- std::vector< int > fixed_col
  Фиксированные столбцы
- std::vector< std::pair< int, int > > banned
- MurtySolution sol
  Запрещенные состояния
- double lb
  Решение
- int split_from = 0

7.7.1 Подробное описание

Узел решения

См. определение в файле murty.hpp строка 43

7.7.2 Данные класса

7.7.2.1 banned

std::vector<std::pair<int,int> > SPML::LAP::MurtyNode::banned

См. определение в файле murty.hpp строка 45

7.7.2.2 fixed_col

std::vector<int> SPML::LAP::MurtyNode::fixed_col

Фиксированные столбцы

См. определение в файле murty.hpp строка 44

7.7.2.3 lb

double SPML::LAP::MurtyNode::lb

Решение

Нижняя граница

См. определение в файле murty.hpp строка 47

7.7.2.4 sol

MurtySolution SPML::LAP::MurtyNode::sol

Запрещенные состояния

См. определение в файле murty.hpp строка 46

7.7.2.5 split_from

int SPML::LAP::MurtyNode::split_from = 0

См. определение в файле murty.hpp строка 48

Объявления и описания членов структуры находятся в файле:

- murty.hpp

7.8 Структура SPML::LAP::MurtySolution

Решение метода Murty.

#include <murty.hpp>

Открытые атрибуты

- arma::ivec x

  результат задачи о назначениях, размерность [dim] (индекс макс/мин элемента в i-ой строке) "rowsol[i] = j" означает, что в i-ой строке выбран j-ый элемент
- arma::vec u

  Двойственная переменная строки
- arma::vec v

  Двойственная переменная столбца
- double cost

  Суммарная стоимость назначения

7.8.1 Подробное описание

Решение метода Murty.

См. определение в файле murty.hpp строка 32

7.8.2 Данные класса

7.8.2.1 cost

double SPML::LAP::MurtySolution::cost

Суммарная стоимость назначения

См. определение в файле murty.hpp строка 36

7.8.2.2 u

arma::vec SPML::LAP::MurtySolution::u

Двойственная переменная строки

См. определение в файле murty.hpp строка 34

7.8.2.3 v

arma::vec SPML::LAP::MurtySolution::v

Двойственная переменная столбца

См. определение в файле murty.hpp строка 35

7.8.2.4 x

arma::ivec SPML::LAP::MurtySolution::x

результат задачи о назначениях, размерность [dim] (индекс макс/мин элемента в i-ой строке) "rowsol[i] = j" означает, что в i-ой строке выбран j-ый элемент

См. определение в файле murty.hpp строка 33

Объявления и описания членов структуры находятся в файле:

- murty.hpp

Глава 8

Файлы

8.1 Файл custom_header.tex

8.2 custom_header.tex

См. документацию.

```
00001 % Latex header for doxygen 1.9.4
00002   % Handle batch mode
00003 %%BEGIN LATEX_BATCHMODE
00004   \batchmode
00005 %%END LATEX_BATCHMODE
00006
00007   % to overcome problems with too many open files
00008   \let\mypdfximage\pdfximage\def\pdfximage{\immediate\mypdfximage}
00009
00010   % Set document class depending on configuration
00011 %%BEGIN COMPACT_LATEX
00012   \documentclass[]{article}
00013 %%END COMPACT_LATEX
00014 %%BEGIN !COMPACT_LATEX
00015   \documentclass[oneside]{book}
00016 %%END !COMPACT_LATEX
00017
00018 \RequirePackage[utf8]{inputenc}
00019 \RequirePackage[T2A]{fontenc}
00020
00021   %% moved from doxygen.sty due to workaround for LaTex 2019 version and unmaintained tabu package
00022   \usepackage{ifthen}
00023   \ifx\requestedLaTeXdate\undefined
00024     \usepackage{array}
00025   \else
00026     \usepackage{array}[=2016-10-06]
00027   \fi
00028   %%
00029
00030   % Packages required by doxygen
00031   \usepackage{fixltx2e} % for \textsubscript
00032   \usepackage{doxygen}
00033
00034   $extralatexstylesheet
00035
00036   \usepackage{graphicx}
00037   \usepackage[utf8]{inputenc}
00038   \usepackage{makeidx}
00039   \PassOptionsToPackage{warn}{textcomp}
00040   \usepackage{textcomp}
00041   \usepackage[nointegrals]{wasysym}
00042   \usepackage{ifxetex}
00043
00044   % NLS support packages
00045   $languagesupport
00046
00047   % Define default fonts
00048   % Font selection
00049 %%BEGIN LATEX_FONTENC
00050   \usepackage[$latexfontenc]{fontenc}
```

```
00051 %%END LATEX_FONTENC
00052
00053   % set main and monospaced font
00054   $latexfont
00055
00056   \usepackage{sectsty}
00057   \allsectionsfont{%
00058     \fontseries{bc}\selectfont%
00059     \color{darkgray}%
00060   }
00061   \renewcommand{\DoxyLabelFont}{%
00062     \fontseries{bc}\selectfont%
00063     \color{darkgray}%
00064   }
00065   \newcommand{\+}{\discretionary{\mbox{\scriptsize$\hookleftarrow$}}{}{}}
00066
00067    % Arguments of doxygenemoji:
00068    % 1) ':<text>:' form of the emoji, already LaTeX-escaped
00069    % 2) file with the name of the emoji without the .png extension
00070    % in case image exist use this otherwise use the ':<text>:' form
00071    \newcommand{\doxygenemoji}[2]{%
00072      \IfFileExists{$latexemojidirectory/#2.png}{\raisebox{-
      0.1em}{\includegraphics[height=0.9em]{$latexemojidirectory/#2.png}}}{#1}%
00073    }
00074
00075   % Page & text layout
00076   \usepackage{geometry}
00077   \geometry{%
00078     $papertype,%
00079     top=2.5cm,%
00080 %     bottom=2.5cm,%
00081 %     left=2.5cm,%
00082 %     right=2.5cm%
00083     bottom=2.5cm,%
00084     left=2.5cm,%
00085     right=1.0cm%
00086   }
00087
00088   % Allow a bit of overflow to go unnoticed by other means
00089   \tolerance=750
00090   \hfuzz=15pt
00091   \hbadness=750
00092   \setlength{\emergencystretch}{15pt}
00093   \setlength{\parindent}{0cm}
00094   \newcommand{\doxynormalparskip}{\setlength{\parskip}{3ex plus 2ex minus 2ex}}
00095   \newcommand{\doxytocparskip}{\setlength{\parskip}{1ex plus 0ex minus 0ex}}
00096   \doxynormalparskip
00097   % Redefine paragraph/subparagraph environments, using sectsty fonts
00098   \makeatletter
00099   \renewcommand{\paragraph}{%
00100     \@startsection{paragraph}{4}{0ex}{-1.0ex}{1.0ex}{%
00101       \normalfont\normalsize\bfseries\SS@parafont%
00102     }%
00103   }
00104   \renewcommand{\subparagraph}{%
00105     \@startsection{subparagraph}{5}{0ex}{-1.0ex}{1.0ex}{%
00106       \normalfont\normalsize\bfseries\SS@subparafont%
00107     }%
00108   }
00109   \makeatother
00110
00111   \makeatletter
00112   \newcommand\hrulefilll{\leavevmode\leaders\hrule\hskip 0pt plus 1filll\kern\z@}
00113   \makeatother
00114
00115   % Headers & footers
00116   \usepackage{fancyhdr}
00117   \pagestyle{fancyplain}
00118   \renewcommand{\footrulewidth}{0.0pt}
00119   \renewcommand{\headrulewidth}{0.0pt}
00120
00121   \fancypagestyle{fancyplain}{
00122     \fancyhf{}
00123     \fancyhead[CE, CO]{\bfseries\thepage}
00124 %     \fancyhead[LE, RO]{\bfseries\thepage}
00125 %     \fancyhead[LO]{\bfseries\rightmark}
00126 %     \fancyhead[RE]{\bfseries\leftmark}
00127 %     \fancyfoot[LO, RE]{\bfseries\scriptsize $generatedby Doxygen }
00128   }
00129
00130   \fancypagestyle{plain}{
00131     \fancyhf{}
00132 %     \fancyfoot[LO, RE]{\bfseries\scriptsize $generatedby Doxygen }
00133 %     \renewcommand{\headrulewidth}{0.0pt}
00134   }
00135
00136   \pagestyle{fancyplain}
```

```


%%BEGIN !COMPACT_LATEX
  \renewcommand{\chaptermark}[1]{%
    \markboth{#1}{}%
  }
%%END !COMPACT_LATEX
  \renewcommand{\sectionmark}[1]{%
    \markright{\thesection\ #1}%
  }

  % ToC, LoF, LoT, bibliography, and index
  % Indices & bibliography
  \usepackage{natbib}
  \usepackage[titles]{tocloft}
  \setcounter{tocdepth}{3}
  \setcounter{secnumdepth}{5}

  % creating indexes
  $makeindex

  $extralatexpackages

  $latexspecialformulachars

%%BEGIN FORMULA_MACROFILE
  \input{$formulamacrofile}
%%END FORMULA_MACROFILE

  % Hyperlinks
%%BEGIN PDF_HYPERLINKS
    % Hyperlinks (required, but should be loaded last)
    \ifpdf
      \usepackage[pdftex,pagebackref=true]{hyperref}
    \else
      \ifxetex
        \usepackage[pagebackref=true]{hyperref}
      \else
        \usepackage[ps2pdf,pagebackref=true]{hyperref}
      \fi
    \fi

    \hypersetup{%
      colorlinks=true,%
      linkcolor=blue,%
      citecolor=blue,%
      unicode,%
      pdftitle={$projectname},%
      pdfsubject={$projectbrief}%
    }

%%END PDF_HYPERLINKS

  % Custom commands used by the header
  % Custom commands
  \newcommand{\clearemptydoublepage}{%
    \newpage{\pagestyle{empty}\cleardoublepage}%
  }

  % caption style definition
  \usepackage{caption}
  \captionsetup{labelsep=space,justification=centering,font={bf},singlelinecheck=off,skip=4pt,position=top}


  % in page table of contents
  \usepackage{etoc}
  \etocsettocstyle{\doxytocparskip}{\doxynormalparskip}

  % prevent numbers overlap the titles in toc
  \renewcommand{\numberline}[1]{#1~}


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%5


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
\linespread{1.25}   % Это эквивалент 1,5 интервала в ворде

\clubpenalty=10000  % Это костыль против
\widowpenalty=10000 % "висячих" строк
\righthyphenmin=200 % Избавляемся
\sloppy             % от переносов слов

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Создает переменную, задается \cmd{val}, возвращается \@cmd
% \var{cmd}
\def\var#1{
```

```
    \expandafter
    \def\csname #1\endcsname##1{
        \ifx&##1&
        \ClassError
        {espd}
        {The \@backslashchar#1 cannot be empty}
        {You should specify non-empty #1 by typing \@backslashchar#1{#1 name}}
        \fi
        \expandafter\def\csname @#1\endcsname{##1}}
    \expandafter
    \def\csname @#1\endcsname{
        \ClassError
        {espd}
        {No \@backslashchar#1 given}
        {You should specify #1 by typing \@backslashchar#1{#1 name} in the preamble}}
}
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Создает переменную, задается \cmd{val}, возвращается \@cmd (по умолчанию - пустая строка)
% \var{cmd}
\def\varempty#1{
    \expandafter
    \def\csname #1\endcsname##1{\expandafter\def\csname @#1\endcsname{##1}}
    \expandafter
    \def\csname @#1\endcsname{}
}
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Пишет в AUX файл данную пару ключ-значение
% \set{key}{val}
\def\set#1#2{\write\@auxout{\string\newlabel{#1}{#2}}}
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Получает из AUX файла значение по ключу
% \get{key}
\def\get#1{\csname r@#1\endcsname}
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Получает из AUX файла значение по ключу (по умолчанию 0)
% \get{key}
\def\getnum#1{\ifcsname r@#1\endcsname \csname r@#1\endcsname \else 0 \fi}
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% ПЕРЕМЕННЫЕ И КОНСТАНТЫ

% Размеры шрифта
%\renewcommand{\normalsize}{\fontsize{12pt}{18pt}\selectfont}
%\newcommand{\titlesize}{\fontsize{14pt}{21pt}\selectfont}
%\newcommand{\approvalsize}{\fontsize{17pt}{24pt}\selectfont}
%\newcommand{\sectionsize}{\fontsize{17pt}{18pt}\selectfont}
%\newcommand{\subsectionsize}{\fontsize{14pt}{16pt}\selectfont}
%\newcommand{\attachmentsize}{\fontsize{17pt}{24pt}\selectfont}
%\newcommand{\tabletextsize}{\fontsize{10pt}{15pt}\selectfont}
%\newcommand{\footnotesize}{\fontsize{8pt}{12pt}\selectfont}
%\newcommand{\footnotetextsize}{\fontsize{10pt}{15pt}\selectfont}
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% FIXED MY VERSION
\renewcommand{\normalsize}{\fontsize{14pt}{18pt}\selectfont}
\newcommand{\titlesize}{\fontsize{14pt}{21pt}\selectfont}
\newcommand{\approvalsize}{\fontsize{17pt}{24pt}\selectfont}
\newcommand{\sectionsize}{\fontsize{14pt}{18pt}\selectfont}
\newcommand{\subsectionsize}{\fontsize{14pt}{16pt}\selectfont}
\newcommand{\attachmentsize}{\fontsize{14pt}{18pt}\selectfont}
%\newcommand{\tabletextsize}{\fontsize{10pt}{15pt}\selectfont}
\newcommand{\tabletextsize}{\fontsize{12pt}{15pt}\selectfont}
%\newcommand{\picturetextsize}{\fontsize{12pt}{15pt}\selectfont} % smaller font
\newcommand{\picturetextsize}{\fontsize{14pt}{18pt}\selectfont} % normal
\newcommand{\listingtextsize}{\fontsize{10pt}{10pt}\selectfont}
\renewcommand{\footnotesize}{\fontsize{8pt}{12pt}\selectfont}
\newcommand{\footnotetextsize}{\fontsize{10pt}{15pt}\selectfont}
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Бумага
\setlength{\paperwidth}{210mm}
\setlength{\paperheight}{297mm}
\setlength{\pdfpagewidth}{\paperwidth}
\setlength{\pdfpageheight}{\paperheight}
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Отступы и размеры
\setlength{\hoffset}{-25.4mm}
\setlength{\voffset}{-25.4mm}
%\setlength{\oddsidemargin}{20mm}
%\setlength{\textwidth}{180mm}
\setlength{\oddsidemargin}{25mm} % Лучше поле задать 25 мм, т.к. подшивают обычно 20 мм
\setlength{\textwidth}{175mm}
```

```
\setlength{\topmargin}{7.5mm}
\setlength{\textheight}{257mm} % 297-25-257=15 - нижнее поле
\setlength{\parindent}{10mm} % Абзацный отступ
\setlength{\parskip}{2mm}
%\setlength{\parskip}{0mm}
\setlength{\headheight}{10mm}
\setlength{\headsep}{7.5mm}
\setlength{\arrayrulewidth}{0.1pt}
\newlength{\tabmarg}
\setlength{\tabmarg}{2mm}
\setlength{\tabcolsep}{\tabmarg}
\raggedbottom
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Константы
\def\wordagree{Согласовано}
\def\wordapprove{Утверждаю}
\def\wordopenquote{«}
\def\wordclosequote{»}
\def\wordyear{г.}
\def\wordapproval{Лист утверждения}
\def\wordapprovalpostfix{-ЛУ}
\def\wordperformer{Исполнитель}
\def\wordperformers{Исполнители}
\def\wordapproved{Утвержден}
\def\wordsheets{Листов}
\def\wordannotation{Аннотация}
\def\wordterms{Перечень терминов}
\def\wordabbreviations{Перечень сокращений}
\def\wordcontents{Содержание}
\def\wordattachment{Приложение}
\def\wordoriginvnumber{Инв. \textnumero~подл.}
\def\wordsignanddate{\wordsign и дата}
\def\wordreferencenumber{Справ.~\textnumero}
\def\wordfirstapplication{Перв.~примен.}
\def\wordinstinvnumber{Взам. инв. \textnumero}
\def\wordduplinvnumber{Инв. \textnumero~дубл.}
\def\wordregistration{Лист регистрации изменений}
\def\wordsheetnumber{Номера листов (страниц)}
\def\wordsheetsindoc{Всего листов (страниц) в докум.}
\def\worddocnumber{\textnumero\\документа}
\def\wordincldocanddate{Входящий \textnumero~сопрово-дительного докум. и дата}
\def\wordsign{Подп.}
\def\worddate{Дата}
\def\wordchanges{Изм.}
\def\wordchanged{изменен-\\ных}
\def\wordreplaced{заменен-\\ных}
%\def\wordchanged{измененных}
%\def\wordreplaced{замененных}
\def\wordnew{новых}
\def\wordannulled{аннули-рованных}
\def\wordimage{Рис.}
\def\wordrefimage{рис.~}
\def\wordbibliography{Перечень использованных источников}
\def\wordempty{}
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Переменные
\var{customername} % Заказчик
\var{customerrank}

\var{fromcustomername} % От заказчика
\var{fromcustomerrank}

\varempty{chiefconstructorname} % Главный конструктор
\varempty{chiefconstructorrank}

\varempty{headofdepartmentrank} % Начальник подразделения (Научно-тематического центра)
\varempty{headofdepartmentname}

\varempty{deputyofchiefconstructorrank} % Заместитель главного конструктора
\varempty{deputyofchiefconstructorname}

\varempty{headoflaboratoryrank} % Начальник лаборатории
\varempty{headoflaboratoryname}

\varempty{developerrank} % Разработчик
\varempty{developername}

\varempty{normocontrollerrank} % Нормоконтроллер
\varempty{normocontrollername}

\var{year}
\var{title}
\var{type}
\var{typecode}
\var{organizationcode}
```

```
\var{redaction}
\var{documentnumber}
\var{partnumber}
\var{registrationcode}

\varempty{authorname} % Исполнитель, если он один
\varempty{authorrank}

\varempty{authori} % Исполнители, если их много
\varempty{authorii}
\varempty{authoriii}
\varempty{authoriv}
\varempty{authorv}
\varempty{authorranki}
\varempty{authorrankii}
\varempty{authorrankiii}
\varempty{authorrankiv}
\varempty{authorrankv}

\varempty{firstapplication}
\varempty{referencenumber}
\varempty{emptyvar}
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
\DeclareOption{exampletask}{
    \type{Пример оформления документации}
    \typecode{ПРИМЕР}
}
\DeclareOption{technicaltask}{
    \type{Техническое задание}
    \typecode{ТЗ}
}
\DeclareOption{codelisting}{
    \type{Текст программы}
    \typecode{12}
}
\DeclareOption{programdescription}{
    \type{Описание программы}
    \typecode{13}
}
\DeclareOption{listoperationaldocuments}{
    \type{Ведомость эксплуатационных документов}
    \typecode{20}
}
\DeclareOption{formular}{
    \type{Формуляр}
    \typecode{30}
}
\DeclareOption{applicationdescription}{
    \type{Описание применения}
    \typecode{31}
}
\DeclareOption{systemprogrammermanual}{
    \type{Руководство системного программиста}
    \typecode{32}
}
\DeclareOption{programmermanual}{
    \type{Руководство программиста}
    \typecode{33}
}
\DeclareOption{operatormanual}{
    \type{Руководство оператора}
    \typecode{34}
}
\DeclareOption{languagedescription}{
    \type{Описание языка}
    \typecode{35}
}
\DeclareOption{guideoftechnicalservice}{
    \type{Руководство по техническому обслуживанию}
    \typecode{46}
}
\DeclareOption{testingmethods}{
    \type{Программа и методика испытаний}
    \typecode{51}
}
\DeclareOption{explanationnote}{
    \type{Пояснительная записка}
    \typecode{81}
}
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Код документа
%\def\fullcode{\@organizationcode.\@registrationcode-\@redaction~\@typecode~\@documentnumber-\@partnumber}
\def\fullcode{\@organizationcode.\@registrationcode-\@redaction~\@typecode~\@documentnumber}
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
% Код документа с постфиксом листа утверждения
\def\fullcodeapp{\fullcode\wordapprovalpostfix}

% Инициализация параметров
\ProcessOptions*

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% МАКРОСЫ ДЛЯ ВНУТРЕННЕГО ИСПОЛЬЗОВАНИЯ
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Вставляет место для подписи для исполнителя без проверки на пустоту строк
\def\onlyauthorraw#1{
    \vskip 1mm\rule{3cm}{0.1pt}~#1\\\vskip 1mm\wordopenquote\rule{8mm}{0.1pt}\wordclosequote~\rule{3cm}{0.1pt} \@year~\wordyear\\\vskip 5mm
}
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% То же, но делает проверку на пустоту строк
% \onlyauthor{author}
\def\onlyauthor#1{
    \if&#1&\else%
    \onlyauthorraw{#1}
    \fi
}
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Вставляет должность и место для подписи для исполнителя без проверки на пустоту строк
% \titleauthorraw{author}{authorrank}
\def\titleauthorraw#1#2{
    \vskip 1mm #2\\\vskip 1mm\rule{3cm}{0.1pt}~#1\\\vskip 1mm \wordopenquote\rule{8mm}{0.1pt}\wordclosequote~\rule{3cm}{0.1pt} \@year~\wordyear\\\vskip 5mm
}
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% То же, но делает проверку на пустоту строк
% \titleauthor{author}{authorrank}
\def\titleauthor#1#2{
    \if&#1#2&\else%
    \titleauthorraw{#1}{#2}
    \fi
}
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Модификация перевода строки для использования \centering и \MakeUppercase
% \protectlinebreak
\def\protectlinebreak{
    \let\defaultlinebreak\\
    \renewcommand{\\}{\protect\defaultlinebreak}
}
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Создает ячейку заданных размеров с центрованным текстом внутри
% \def{width}{height}{text}
\def\cell#1#2#3{
    \parbox[c][#2][c]{#1}{\centering #3}%
}
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Создает ячейку с центрованным текстом внутри с настраиваемой реальной высотой
% \def{width}{height}{realheight}{text}
%\def\vcell#1#2#3#4{\parbox[c][#3][b]{#1}{\footnotesize}{\fontsize{8pt}{12pt}\selectfont}
%   \newcommand{\foovbox to #2{\cell{#1}{#2}{#4}\vss}}}

\def\vcell#1#2#3#4{\parbox[c][#3][b]{#1}{\vbox to #2{\cell{#1}{#2}{#4}\vss}}}

\def\writesection#1{
    %  \newpage
    \begin{center}
        \centering\sectionsize\MakeUppercase{#1}
    \end{center}
    \par
}

\def\writesubsection#1{
    \begingroup
    \par
    \parindent -1cm
    \leftskip 2cm
    \interlinepenalty\@M
    \vspace{4mm}
    {\subsectionsize#1}%
    \vspace{2mm}
    \par
    \endgroup
}
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
% СЕКЦИОНИРОВАНИЕ
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Счетчики
%\setcounter{secnumdepth}{4}
%\newcounter{section}
%\newcounter{subsection}[section]
%\newcounter{paragraph}[subsection]
%\newcounter{subparagraph}[paragraph]
\newcounter{attachment}
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Разделы, подразделы, пункты и подпункты
\renewcommand{\thesection}{\arabic{section}}
\renewcommand{\thesubsection}{\arabic{section}.\arabic{subsection}}
\renewcommand{\theparagraph}{\arabic{section}.\arabic{subsection}.\arabic{paragraph}}
\renewcommand{\thesubparagraph}{\arabic{section}.\arabic{subsection}.\arabic{paragraph}.\arabic{subparagraph}}
\renewcommand{\theattachment}{\arabic{attachment}}
\newcommand{\numsection}{\thesection.}
\newcommand{\numsubsection}{\thesubsection.}
\newcommand{\numparagraph}{\theparagraph.}
\newcommand{\numsubparagraph}{\thesubparagraph.}
\renewcommand{\section}[1]{
   \ifx&#1&
   \ClassError
   {espd}
   {Empty section title}
   {According to GOST 19.106-78, section title can not be empty}
   \fi
   \refstepcounter{section}
   \writesection{\numsection~#1}
   \addcontentsline{toc}{section}{\numsection~#1}
}
\renewcommand{\subsection}[1]{
   \ifx&#1&
   \ClassError
   {espd}
   {Empty subsection title}
   {According to GOST 19.106-78, subsection title can not be empty}
   \fi
   \refstepcounter{subsection}
   \writesubsection{\numsubsection~#1}
   \addcontentsline{toc}{subsection}{\numsubsection~#1}
}
\renewcommand{\paragraph}[1]{
   \parskip1mm
   \refstepcounter{paragraph}
   \writesubsection{\numparagraph~#1}
   \ifx&#1&\else
   \addcontentsline{toc}{paragraph}{\numparagraph~#1}
   \fi
}
\renewcommand{\subparagraph}[1]{
   \refstepcounter{subparagraph}
   \writesubsection{\numsubparagraph~#1}
   \ifx&#1&\else
   \addcontentsline{toc}{subparagraph}{\numsubparagraph~#1}
   \fi
}
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Штампы в разных вариациях по ГОСТ19
\newcommand{\stampleftfull}{
   \setlength{\arrayrulewidth}{0.5mm}
   \tabletextsize
   \setlength{\tabcolsep}{0mm}
   %   \itshape\fontspec{GOST type A Italic} % Italic приходится задавать так
   \begin{tabular}{|p{24.5mm}|p{34.5mm}|p{24.5mm}|p{24.5mm}|p{34.5mm}|p{21.0mm}|p{59.5mm}|p{59.5mm}|}
      \cline{1-5}\cline{7-8}
      \cell{24.5mm}{5mm}{\wordoriginvnumber} &
      \cell{34.5mm}{5mm}{\wordsignanddate} &
      \cell{24.5mm}{5mm}{\wordinstinvnumber} &
      \cell{24.5mm}{5mm}{\wordduplinvnumber} &
      \cell{34.5mm}{5mm}{\wordsignanddate} &
      \cell{21.0mm}{5mm}{} &
      \cell{59.5mm}{5mm}{\wordreferencenumber} &
      \cell{59.5mm}{5mm}{\wordfirstapplication} \\
      \cline{1-5}\cline{7-8}
      \cell{24.5mm}{7mm}{} & & & & & \cell{59.5mm}{7mm}{\normalsize\@referencenumber} &
   \cell{59.5mm}{7mm}{\normalsize\@firstapplication} \\
      \cline{1-5}\cline{7-8}
   \end{tabular}
   \setlength{\tabcolsep}{\tabmarg}
}
\newcommand{\stampleftshort}{
   \setlength{\arrayrulewidth}{0.5mm}
   \tabletextsize
```

```
    \setlength{\tabcolsep}{0mm}
    %   \itshape\fontspec{GOST type A Italic} % Italic приходится задавать так
    \begin{tabular}{|p{24.5mm}|p{34.5mm}|p{24.5mm}|p{24.5mm}|p{34.5mm}|p{21.5mm} p{60mm} p{60mm}}
        \cline{1-5}
        \cell{24.5mm}{5mm}{\wordoriginvnumber} &
        \cell{34.5mm}{5mm}{\wordsignanddate} &
        \cell{24.5mm}{5mm}{\wordinstinvnumber} &
        \cell{24.5mm}{5mm}{\wordduplinvnumber} &
        \cell{34.5mm}{5mm}{\wordsignanddate} &
        \cell{21.5mm}{5mm}{} &
        \cell{60mm}{5mm}{} &
        \cell{60mm}{5mm}{} \\
        \cline{1-5}
        \cell{24.5mm}{7mm}{} & & & & & & & \\
        \cline{1-5}
    \end{tabular}
    \setlength{\tabcolsep}{\tabmarg}
}
\newcommand{\stampleftonlyfirstapp}{
    \setlength{\arrayrulewidth}{0.5mm}
    \tabletextsize
    \setlength{\tabcolsep}{0mm}
    %   \itshape\fontspec{GOST type A Italic} % Italic приходится задавать так
    \begin{tabular}{|p{24.5mm}|p{34.5mm}|p{24.5mm}|p{24.5mm}|p{34.5mm}|p{21.5mm} p{59.5mm}|p{59.5mm}|}
        \cline{1-5}\cline{8-8}
        \cell{24.5mm}{5mm}{\wordoriginvnumber} &
        \cell{34.5mm}{5mm}{\wordsignanddate} &
        \cell{24.5mm}{5mm}{\wordinstinvnumber} &
        \cell{24.5mm}{5mm}{\wordduplinvnumber} &
        \cell{34.5mm}{5mm}{\wordsignanddate} &
        \cell{21.0mm}{5mm}{} &
        \cell{59.5mm}{5mm}{} &
        \cell{59.5mm}{5mm}{\wordfirstapplication} \\
        \cline{1-5}\cline{8-8}
        \cell{24.5mm}{7mm}{} & & & & & & \cell{59.5mm}{7mm}{\normalsize\@firstapplication} \\
        \cline{1-5}\cline{8-8}
    \end{tabular}
    \setlength{\tabcolsep}{\tabmarg}
}
\newcommand{\stampleftonlyrefnum}{
    \setlength{\arrayrulewidth}{0.5mm}
    \tabletextsize
    \setlength{\tabcolsep}{0mm}
    %   \itshape\fontspec{GOST type A Italic} % Italic приходится задавать так
    \begin{tabular}{|p{24.5mm}|p{34.5mm}|p{24.5mm}|p{24.5mm}|p{34.5mm}|p{21.0mm}|p{59.5mm}|p{60.0mm}}
        \cline{1-5}\cline{7-7}
        \cell{24.5mm}{5mm}{\wordoriginvnumber} &
        \cell{34.5mm}{5mm}{\wordsignanddate} &
        \cell{24.5mm}{5mm}{\wordinstinvnumber} &
        \cell{24.5mm}{5mm}{\wordduplinvnumber} &
        \cell{34.5mm}{5mm}{\wordsignanddate} &
        \cell{21.0mm}{5mm}{} &
        \cell{59.5mm}{5mm}{\wordreferencenumber} &
        \cell{60.0mm}{5mm}{} \\
        \cline{1-5}\cline{7-7}
        \cell{24.5mm}{7mm}{} & & & & & \cell{59.5mm}{7mm}{\normalsize\@referencenumber} & \\
        \cline{1-5}\cline{7-7}
    \end{tabular}
    \setlength{\tabcolsep}{\tabmarg}
}
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Лист утверждения
%\newcommand{\makeapproval}{
%   \begin{titlepage}
%       \reversemarginpar
%       %  \setlength{\marginparsep}{11.5mm}
%       \setlength{\marginparsep}{17mm}
%       %\marginpar{\vskip -2.5cm \hbox{\rotatebox{90}{\stampleftfull}}}
%
%%      \marginpar{\vskip -2.5cm \hbox{\rotatebox{90}{\stampleftshort}}}
%
%       %\marginpar{\vskip -2.5cm \hbox{\rotatebox{90}{\stampleftonlyfirstapp}}}
%       %\marginpar{\vskip -2.5cm \hbox{\rotatebox{90}{\stampleftonlyrefnum}}}
%       \vskip -\baselineskip
%       \centering
%       \protectlinebreak
%       \begin{tabular}{c c}
%           \parbox[t][2cm]{0.5\textwidth}{
%               \centering\MakeUppercase{\wordagree}\vskip 2mm\@customerrank\vskip 1mm\onlyauthor{\@customername}
%           } &
%
%           \parbox[t][2cm]{0.5\textwidth}{
%               \centering\MakeUppercase{\wordapprove}\vskip 2mm\@chiefconstructorrank\vskip
```

```
1mm\onlyauthor{\@chiefconstructorname}
%         }
%     \end{tabular}
%
%     \vskip 3cm
%     \titlesize{\MakeUppercase{\@title}}
%     \vskip 4mm
%     \normalsize\textbf{\@type}
%     \vskip 4mm
%     \normalsize{\MakeUppercase{\wordapproval}}
%     \vskip 4mm
%     \normalsize{\fullcodeapp}
%     \vskip \fill
%     \begin{tabular}{c c}
%         \parbox[t]{0.45\textwidth}{\raggedright
%             % левая колонка:
%             \if& \@headofdepartmentname &
%             &
%             \else
%             \titleauthor{\@fromcustomername}{\@fromcustomerrank}} &
%         \fi
%
%         \parbox[t]{0.4\textwidth}{\raggedright
%             % правая колонка:
%
%             \if& \@headofdepartmentname &
%             \else
%             \titleauthor{\@headofdepartmentname}{\@headofdepartmentrank}
%             \fi
%
%             \if& \@deputyofchiefconstructorname &
%             \else
%             \titleauthor{\@deputyofchiefconstructorname}{\@deputyofchiefconstructorrank}
%             \fi
%
%             \if& \@headoflaboratoryname &
%             \else
%             \titleauthor{\@headoflaboratoryname}{\@headoflaboratoryrank}
%             \fi
%
%             \if& \@developername &
%             \else
%             \titleauthor{\@developername}{\@developerrank}
%             \fi
%
%             \if&\@authorname&
%             \else
%
\if&\@authori\@authorii\@authoriii\@authoriv\@authorv\@authorranki\@authorrankii\@authorrankiii\@authorrankiv\@authorrankv&
%                 \wordperformer\\\onlyauthor{\@authorname}
%             \else
%
\wordperformers\\\onlyauthor{\@authori}\onlyauthor{\@authorii}\onlyauthor{\@authoriii}\onlyauthor{\@authoriv}\onlyauthor{\@author
%             \fi
%             \fi
%
%             \if& \@normocontrollername &
%             \else
%             \titleauthor{\@normocontrollername}{\@normocontrollerrank}
%             \fi
%
%         }
%     \end{tabular}
%     \vskip \fill
%     \@year
%     \vskip 5mm
%   \end{titlepage}
%}

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Титульный лист
%\renewcommand{\maketitle}{
%   \begin{titlepage}
%     \reversemarginpar
%     \setlength{\marginparsep}{17mm}
%     %\marginpar{\vskip -2.5cm \hbox{\rotatebox{90}{\stampleftfull}}}
%     %\marginpar{\vskip -2.5cm \hbox{\rotatebox{90}{\stampleftshort}}}
%
%%     \marginpar{\vskip -2.5cm \hbox{\rotatebox{90}{\stampleftonlyfirstapp}}}
%
%     %\marginpar{\vskip -2.5cm \hbox{\rotatebox{90}{\stampleftonlyrefnum}}}
%     \vskip -\baselineskip
%     \parbox[t]{0.4\textwidth}{\centering\MakeUppercase{\wordapproved}\\\fullcodeapp}
%     \vskip 8cm
%     \centering
%     \protectlinebreak
%     \titlesize{\MakeUppercase{\@title}}
```

```
%       \vskip 4mm
%       \normalsize\textbf{\@type}
%       \vskip 4mm
%       {\fullcode}
%       \vskip 4mm
%       {\wordsheets~\getnum{lastpage}}
%       \vskip \fill
%       \@year
%       \vskip 5mm
%   \end{titlepage}
%   \stepcounter{page}
%}
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Содержание
%\renewcommand\@pnumwidth{1em}
%\renewcommand\@tocrmarg{4em}
%\renewcommand\@dotsep{4}
%\setcounter{tocdepth}{4}
%\renewcommand\tableofcontents{
%   \newpage
%   \writesection{\wordcontents}
%   \@starttoc{toc}
%}
%%\newcommand\l@section[1]{\@dottedtocline{1}{0cm}{0.5cm}{\textbf{#1}}}
%%\newcommand\l@subsection{\@dottedtocline{2}{0.5cm}{1cm}}
%%\newcommand\l@paragraph{\@dottedtocline{3}{1cm}{1.5cm}}
%%\newcommand\l@subparagraph{\@dottedtocline{4}{1.5cm}{2cm}}
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%\renewcommand\l@section[1]{\@dottedtocline{1}{0cm}{1cm}{#1}}
%\renewcommand\l@subsection{\@dottedtocline{2}{1cm}{2cm}}
%\renewcommand\l@paragraph{\@dottedtocline{3}{2cm}{3cm}}
%\renewcommand\l@subparagraph{\@dottedtocline{4}{3cm}{4cm}}
%\newcommand\l@attachment{\@dottedtocline{1}{0cm}{3cm}}

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% End of preamble, now comes the document contents
%===== C O N T E N T S =====

%\AtBeginDocument{\makeapproval\maketitle}

\begin{document}
  \raggedbottom

  $latexdocumentpre

  % Titlepage & ToC
%%BEGIN PDF_HYPERLINKS
%%BEGIN USE_PDFLATEX
    % To avoid duplicate page anchors due to reuse of same numbers for
    % the index (be it as roman numbers)
    \hypersetup{pageanchor=false,
                bookmarksnumbered=true,
                pdfencoding=unicode
               }
%%END USE_PDFLATEX
%%END PDF_HYPERLINKS
%  \pagenumbering{alph}

%  \begin{titlepage}
%  \vspace*{7cm}
%  \begin{center}%
%  {\Large $title}\\
%  \vspace*{1cm}
%  {\large $generatedby Doxygen $doxygenversion}\\
%%%BEGIN LATEX_TIMESTAMP
%    \vspace*{0.5cm}
%    {\small $datetime}
%%%END LATEX_TIMESTAMP
%  \end{center}
%  \end{titlepage}


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Задание полей титульных страниц
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
\newcommand{\productcipher}{АБВГД.12345} % Шифр изделия

\firstapplication{\productcipher} % Первое применение
\referencenumber{\productcipher} % Справ №

\organizationcode{01234} % Код организации
\registrationcode{56789} % Регистрационный код
```

```
00879 \redaction{01} % Номер редакции
00880 \documentnumber{03} % Номер документа данного вида
00881 %\partnumber{1} % Номер части документа - чтобы задать - надо фиксить класс espd
00882
00883 % СОГЛАСОВАНО
00884 \customerrank{Начальник\\межгалактической комиссии}
00885 \customername{А.Б.~Заказчиков}
00886
00887 % УТВЕРЖДАЮ
00888 \chiefconstructorrank{Главный конструктор\\изделия \productcipher}
00889 \chiefconstructorname{А.Б.~Главный}
00890
00891 \fromcustomerrank{От межгалактической комиссии}
00892 \fromcustomername{А.Б.~Галактионов}
00893
00894 \headofdepartmentrank{Начальник Центра}
00895 \headofdepartmentname{А.Б.~Чатланин}
00896
00897 \deputyofchiefconstructorrank{Зам.~гл.~конструктора}
00898 \deputyofchiefconstructorname{А.Б.~Заместителев}
00899 %
00900 \developerrank{Разработчик}
00901 \developername{А.Б.~Разработчиков}
00902
00903 %\headoflaboratoryrank{Начальник лаборатории 777}
00904 %\headoflaboratoryname{А.Б.~Лабораториев}
00905
00906 % Исполнитель(и)
00907 \authorname{А.Б.~Пацак}
00908 %%
00909 %\authori{А.Б.~Пацак1}
00910 %\authorii{А.Б.~Пацак2}
00911
00912 \normocontrollerrank{Нормоконтроллер}
00913 \normocontrollername{~}
00914
00915 \title{Изделие \productcipher\\Программный комплекс\\Галактический транклюкатор}
00916 \year{2022}
00917
00918     \begin{titlepage}
00919     \reversemarginpar
00920     %  \setlength{\marginparsep}{11.5mm}
00921     \setlength{\marginparsep}{17mm}
00922     %\marginpar{\vskip -2.5cm \hbox{\rotatebox{90}{\stampleftfull}}}
00923
00924     %       \marginpar{\vskip -2.5cm \hbox{\rotatebox{90}{\stampleftshort}}}
00925
00926     %\marginpar{\vskip -2.5cm \hbox{\rotatebox{90}{\stampleftonlyfirstapp}}}
00927     %\marginpar{\vskip -2.5cm \hbox{\rotatebox{90}{\stampleftonlyrefnum}}}
00928     \vskip -\baselineskip
00929     \centering
00930     \protectlinebreak
00931     \begin{tabular}{c c}
00932         \parbox[t][2cm]{0.5\textwidth}{
00933             \centering\MakeUppercase{\wordagree}\vskip 2mm\@customerrank\vskip 1mm\onlyauthor{\@customername}
00934         } &
00935
00936         \parbox[t][2cm]{0.5\textwidth}{
00937             \centering\MakeUppercase{\wordapprove}\vskip 2mm\@chiefconstructorrank\vskip
      1mm\onlyauthor{\@chiefconstructorname}
00938         }
00939     \end{tabular}
00940
00941     \vskip 3cm
00942     \titlesize{\MakeUppercase{\@title}}
00943     \vskip 4mm
00944     \normalsize\textbf{\@type}
00945     \vskip 4mm
00946     \normalsize{\MakeUppercase{\wordapproval}}
00947     \vskip 4mm
00948     \normalsize{\fullcodeapp}
00949     \vskip \fill
00950     \begin{tabular}{c c}
00951         \parbox[t]{0.45\textwidth}{\raggedright
00952             % левая колонка:
00953             \if& \@headofdepartmentname &
00954             &
00955             \else
00956             \titleauthor{\@fromcustomername}{\@fromcustomerrank}} &
00957         \fi
00958
00959         \parbox[t]{0.4\textwidth}{\raggedright
00960             % правая колонка:
00961
00962             \if& \@headofdepartmentname &
00963             \else
00964             \titleauthor{\@headofdepartmentname}{\@headofdepartmentrank}
```

```
00965           \fi
00966
00967           \if& \@deputyofchiefconstructorname &
00968           \else
00969           \titleauthor{\@deputyofchiefconstructorname}{\@deputyofchiefconstructorrank}
00970           \fi
00971
00972           \if& \@headoflaboratoryname &
00973           \else
00974           \titleauthor{\@headoflaboratoryname}{\@headoflaboratoryrank}
00975           \fi
00976
00977           \if& \@developername &
00978           \else
00979           \titleauthor{\@developername}{\@developerrank}
00980           \fi
00981
00982           \if&\@authorname&
00983           \else
00984
    \if&\@authori\@authorii\@authoriii\@authoriv\@authorv\@authorranki\@authorrankii\@authorrankiii\@authorrankiv\@authorrankv&
00985           \wordperformer\\\onlyauthor{\@authorname}
00986           \else
00987
    \wordperformers\\\onlyauthor{\@authori}\onlyauthor{\@authorii}\onlyauthor{\@authoriii}\onlyauthor{\@authoriv}\onlyauthor{\@authorv
00988           \fi
00989           \fi
00990
00991           \if& \@normocontrollername &
00992           \else
00993           \titleauthor{\@normocontrollername}{\@normocontrollerrank}
00994           \fi
00995
00996       }
00997   \end{tabular}
00998   \vskip \fill
00999   \@year
01000   \vskip 5mm
01001   \end{titlepage}
01002
01003
01004 % \makeapproval
01005 % \maketitle
01006
01007 %%BEGIN !COMPACT_LATEX
01008   \clearemptydoublepage
01009 %%END !COMPACT_LATEX
01010 %  \pagenumbering{roman}
01011
01012   \tableofcontents
01013 %%BEGIN !COMPACT_LATEX
01014   \clearemptydoublepage
01015 %%END !COMPACT_LATEX
01016   \pagenumbering{arabic}
01017
01018 %%BEGIN PDF_HYPERLINKS
01019 %%BEGIN USE_PDFLATEX
01020   % re-enable anchors again
01021   \hypersetup{pageanchor=true}
01022 %%END USE_PDFLATEX
01023 %%END PDF_HYPERLINKS
01024
01025 %--- Begin generated contents ---
```

8.3 Файл compare.hpp

Функции сравнения чисел, массивов

#include <cmath>

Пространства имен

- namespace SPML

  Специальная библиотека программных модулей (СБ ПМ)
- namespace SPML::Compare

  Сравнение чисел

Функции

- bool SPML::Compare::AreEqualAbs (float first, float second, const float &eps=EPS_F)

  Сравнение двух действительных чисел (по абсолютной разнице)
- bool SPML::Compare::AreEqualAbs (double first, double second, const double &eps=EPS_D)

  Сравнение двух действительных чисел (по абсолютной разнице)
- bool SPML::Compare::AreEqualRel (float first, float second, const float &eps=EPS_REL)

  Сравнение двух действительных чисел (по относительной разнице)
- bool SPML::Compare::AreEqualRel (double first, double second, const double &eps=EPS_REL)

  Сравнение двух действительных чисел (по относительной разнице)
- bool SPML::Compare::IsZeroAbs (float value, const float &eps=EPS_F)

  Проверка действительного числа на равенство нулю (по абсолютной разнице)
- bool SPML::Compare::IsZeroAbs (double value, const double &eps=EPS_D)

  Проверка действительного числа на равенство нулю (по абсолютной разнице)

8.3.1 Подробное описание

Функции сравнения чисел, массивов

Дата

27.07.20 - создан

Автор

Соболев А.А.

См. определение в файле compare.hpp

8.4 compare.hpp

См. документацию.

```
00001 //-----------------------------------------------------------------------------------------------------
00010
00011 #ifndef SPML_COMPARE_H
00012 #define SPML_COMPARE_H
00013
00014 // System includes:
00015 #include <cmath>
00016
00017 namespace SPML
00018 {
00019 namespace Compare
00020 {
00021 //-----------------------------------------------------------------------------------------------------
00022 static const float EPS_F = 1.0e-4f;
00023 static const double EPS_D = 1.0e-8;
00024 static const float EPS_REL = 0.01;
00025
00026 //-----------------------------------------------------------------------------------------------------
00035 inline bool AreEqualAbs( float first, float second, const float &eps = EPS_F )
00036 {
00037     return ( std::abs( first - second ) <= eps );
00038 }
00039
00048 inline bool AreEqualAbs( double first, double second, const double &eps = EPS_D )
00049 {
00050     return ( std::abs( first - second ) <= eps );
00051 }
00052
```

```
00053 //-----------------------------------------------------------------------------------------------------------
00062 inline bool AreEqualRel( float first, float second, const float &eps = EPS_REL )
00063 {
00064     return ( ( std::abs( first - second ) <= ( eps * std::abs( first ) ) ) &&
00065         ( std::abs( first - second ) <= ( eps * std::abs( second ) ) ) );
00066 }
00067
00076 inline bool AreEqualRel( double first, double second, const double &eps = EPS_REL )
00077 {
00078     return ( ( std::abs( first - second ) <= ( eps * std::abs( first ) ) ) &&
00079         ( std::abs( first - second ) <= ( eps * std::abs( second ) ) ) );
00080 }
00081
00082 //-----------------------------------------------------------------------------------------------------------
00090 inline bool IsZeroAbs( float value, const float &eps = EPS_F )
00091 {
00092     return ( std::abs( value ) <= eps );
00093 }
00094
00102 inline bool IsZeroAbs( double value, const double &eps = EPS_D )
00103 {
00104     return ( std::abs( value ) <= eps );
00105 }
00106
00107 } // end namespace Compare
00108 } // end namespace SPML
00109 #endif // SPML_COMPARE_H
```

8.5 Файл hungarian.cpp

Решение задачи о назначениях венгерским методом (Hungarian, Munkres)

#include <hungarian.hpp>

Пространства имен

- namespace SPML

  Специальная библиотека программных модулей (СБ ПМ)
- namespace SPML::LAP

  Решение задачи о назначениях

Функции

- void SPML::LAP::hungarian_step_1 (unsigned int &step, arma::mat &cost, const unsigned int &N)
- void SPML::LAP::hungarian_step_2 (unsigned int &step, const arma::mat &cost, arma::umat &indM, arma::ivec &rcov, arma::ivec &ccov, const unsigned int &N)
- void SPML::LAP::hungarian_step_3 (unsigned int &step, const arma::umat &indM, arma::ivec &ccov, const unsigned int &N)
- void SPML::LAP::hungarian_find_noncovered_zero (int &row, int &col, const arma::mat &cost, const arma::ivec &rcov, const arma::ivec &ccov, const unsigned int &N)
- bool SPML::LAP::hungarian_star_in_row (int &row, const arma::umat &indM, const unsigned int &N)
- void SPML::LAP::hungarian_find_star_in_row (const int &row, int &col, const arma::umat &indM, const unsigned int &N)
- void SPML::LAP::hungarian_step_4 (unsigned int &step, const arma::mat &cost, arma::umat &indM, arma::ivec &rcov, arma::ivec &ccov, int &rpath_0, int &cpath_0, const unsigned int &N)
- void SPML::LAP::hungarian_find_star_in_col (const int &col, int &row, const arma::umat &indM, const unsigned int &N)

- void SPML::LAP::hungarian_find_prime_in_row (const int &row, int &col, const arma::umat &indM, const unsigned int &N)
- void SPML::LAP::hungarian_augment_path (const int &path_count, arma::umat &indM, const arma::imat &path)
- void SPML::LAP::hungarian_clear_covers (arma::ivec &rcov, arma::ivec &ccov)
- void SPML::LAP::hungarian_erase_primes (arma::umat &indM, const unsigned int &N)
- void SPML::LAP::hungarian_step_5 (unsigned int &step, arma::umat &indM, arma::ivec &rcov, arma::ivec &ccov, arma::imat &path, int &rpath_0, int &cpath_0, const unsigned int &N)
- void SPML::LAP::hungarian_find_smallest (double &minval, const arma::mat &cost, const arma::ivec &rcov, const arma::ivec &ccov, const unsigned int &N)
- void SPML::LAP::hungarian_step_6 (unsigned int &step, arma::mat &cost, const arma::ivec &rcov, const arma::ivec &ccov, const unsigned int &N)
- void SPML::LAP::Hungarian (const arma::mat &assigncost, int dim, TSearchParam sp, double infValue, double resolution, arma::ivec &rowsol, double &lapcost)

  Венгерский метод решения задачи о назначениях (Метод Мункреса)

8.5.1 Подробное описание

Решение задачи о назначениях венгерским методом (Hungarian, Munkres)

Дата

18.05.22 - создан

Автор

Соболев А.А.

См. определение в файле hungarian.cpp

8.6 hungarian.cpp

См. документацию.

```
00001 //-----------------------------------------------------------------------------------------------------------
00010
00011 #include <hungarian.hpp>
00012
00013 namespace SPML
00014 {
00015 namespace LAP
00016 {
00017
00018 void hungarian_step_1( unsigned int &step, arma::mat &cost, const unsigned int &N )
00019 {
00020     for( unsigned int r = 0; r < N; ++r ) {
00021         cost.row(r) -= arma::min( cost.row(r) );
00022     }
00023     step = 2;
00024 }
00025
00039 void hungarian_step_2( unsigned int &step, const arma::mat &cost, arma::umat &indM, arma::ivec &rcov,
00040     arma::ivec &ccov, const unsigned int &N )
00041 {
00042     for( unsigned int r = 0; r < N; ++r ) {
00043         for( unsigned int c = 0; c < N; ++c ) {
00044             if( cost.at(r, c) == 0.0 && rcov.at(r) == 0 && ccov.at(c) == 0) {
00045                 indM.at(r, c) = 1;
00046                 rcov.at(r)    = 1;
00047                 ccov.at(c)    = 1;
00048                 break;                                  // Only take the first
00049                                                         // zero in a row and column
```

```
00050             }
00051         }
00052     }
00053     /* for later reuse */
00054     rcov.fill(0);
00055     ccov.fill(0);
00056     step = 3;
00057 }
00058
00077 void hungarian_step_3( unsigned int &step, const arma::umat &indM, arma::ivec &ccov, const unsigned int &N )
00078 {
00079     unsigned int colcount = 0;
00080     for( unsigned int r = 0; r < N; ++r ) {
00081         for( unsigned int c = 0; c < N; ++c ) {
00082             if( indM.at(r, c) == 1 ) {
00083                 ccov.at(c) = 1;
00084             }
00085         }
00086     }
00087     for( unsigned int c = 0; c < N; ++c ) {
00088         if( ccov.at(c) == 1 ) {
00089             ++colcount;
00090         }
00091     }
00092     if( colcount == N ) {
00093         step = 7;
00094     } else {
00095         step = 4;
00096     }
00097 }
00098
00110 void hungarian_find_noncovered_zero( int &row, int &col, const arma::mat &cost, const arma::ivec &rcov,
00111     const arma::ivec &ccov, const unsigned int &N )
00112 {
00113     unsigned int r = 0;
00114     unsigned int c;
00115     bool done = false;
00116     row = -1;
00117     col = -1;
00118     while( !done ) {
00119         c = 0;
00120         while( true ) {
00121             if( cost.at(r, c) == 0.0 && rcov.at(r) == 0 && ccov.at(c) == 0 ) {
00122                 row = r;
00123                 col = c;
00124                 done = true;
00125             }
00126             ++c;
00127             if( c == N || done ) {
00128                 break;
00129             }
00130         }
00131         ++r;
00132         if( r == N ) {
00133             done = true;
00134         }
00135     }
00136 }
00137
00153 bool hungarian_star_in_row( int &row, const arma::umat &indM, const unsigned int &N )
00154 {
00155     bool tmp = false;
00156     for( unsigned int c = 0; c < N; ++c ) {
00157         if( indM.at(row, c) == 1 ) {
00158             tmp = true;
00159             break;
00160         }
00161     }
00162     return tmp;
00163 }
00164
00165 void hungarian_find_star_in_row( const int &row, int &col, const arma::umat &indM, const unsigned int &N )
00166 {
00167     col = -1;
00168     for( unsigned int c = 0; c < N; ++c ) {
00169         if( indM.at(row, c) == 1 ) {
00170             col = c;
00171         }
00172     }
00173 }
00174
00180 void hungarian_step_4( unsigned int &step, const arma::mat &cost, arma::umat &indM, arma::ivec &rcov, arma::ivec
      &ccov,
00181     int &rpath_0, int &cpath_0, const unsigned int &N )
00182 {
00183     int row = -1;
00184     int col = -1;
```

```
00185     bool done = false;
00186     while( !done ) {
00187         hungarian_find_noncovered_zero( row, col, cost, rcov, ccov, N );
00188         if( row == -1 ) {
00189             done = true;
00190             step = 6;
00191         } else {
00192             /* uncovered zero */
00193             indM( row, col ) = 2;
00194             if( hungarian_star_in_row( row, indM, N ) ) {
00195                 hungarian_find_star_in_row( row, col, indM, N );
00196                 /* Cover the row with the starred zero
00197                  * and uncover the column with the starred
00198                  * zero.
00199                  */
00200                 rcov.at(row) = 1;
00201                 ccov.at(col) = 0;
00202             } else {
00203                 /* No starred zero in row with
00204                  * uncovered zero
00205                  */
00206                 done = true;
00207                 step = 5;
00208                 rpath_0 = row;
00209                 cpath_0 = col;
00210             }
00211         }
00212     }
00213 }
00214
00232 void hungarian_find_star_in_col( const int &col, int &row, const arma::umat &indM, const unsigned int &N )
00233 {
00234     row = -1;
00235     for( unsigned int r = 0; r < N; ++r ) {
00236         if( indM.at(r, col) == 1 ) {
00237             row = r;
00238         }
00239     }
00240 }
00241
00247 void hungarian_find_prime_in_row( const int &row, int &col, const arma::umat &indM, const unsigned int &N )
00248 {
00249     for( unsigned int c = 0; c < N; ++c ) {
00250         if( indM.at(row, c) == 2 ) {
00251             col = c;
00252         }
00253     }
00254 }
00255
00261 void hungarian_augment_path( const int &path_count, arma::umat &indM, const arma::imat &path )
00262 {
00263 //    for (unsigned int p = 0; p < path_count; ++p) {
00264     for( int p = 0; p < path_count; ++p ) {
00265         if( indM.at( path(p, 0), path(p, 1) ) == 1) {
00266             indM.at( path(p, 0), path(p, 1) ) = 0;
00267         } else {
00268             indM.at( path(p, 0), path(p, 1) ) = 1;
00269         }
00270     }
00271 }
00272
00273 void hungarian_clear_covers( arma::ivec &rcov, arma::ivec &ccov )
00274 {
00275     rcov.fill(0);
00276     ccov.fill(0);
00277 }
00278
00279 void hungarian_erase_primes( arma::umat &indM, const unsigned int &N )
00280 {
00281     for( unsigned int r = 0; r < N; ++r ) {
00282         for( unsigned int c = 0; c < N; ++c ) {
00283             if( indM.at(r, c) == 2 ) {
00284                 indM.at(r, c) = 0;
00285             }
00286         }
00287     }
00288 }
00289
00296 void hungarian_step_5( unsigned int &step, arma::umat &indM, arma::ivec &rcov, arma::ivec &ccov, arma::imat &path,
00297     int &rpath_0, int &cpath_0, const unsigned int &N )
00298 {
00299     bool done = false;
00300     int row = -1;
00301     int col = -1;
00302     unsigned int path_count = 1;
00303     path.at(path_count - 1, 0) = rpath_0;
00304     path.at(path_count - 1, 1) = cpath_0;
```

```
00305     while( !done ) {
00306         hungarian_find_star_in_col( path.at(path_count - 1, 1), row, indM, N );
00307         if( row > -1 ) {
00308             /* Starred zero in row 'row' */
00309             ++path_count;
00310             path.at(path_count - 1, 0) = row;
00311             path.at(path_count - 1, 1) = path.at(path_count - 2, 1);
00312         } else {
00313             done = true;
00314         }
00315         if( !done ) {
00316             /* If there is a starred zero find a primed
00317              * zero in this row; write index to 'col' */
00318             hungarian_find_prime_in_row( path.at(path_count - 1, 0), col, indM, N);
00319             ++path_count;
00320             path.at(path_count - 1, 0) = path.at(path_count - 2, 0);
00321             path.at(path_count - 1, 1) = col;
00322         }
00323     }
00324     hungarian_augment_path( path_count, indM, path );
00325     hungarian_clear_covers( rcov, ccov );
00326     hungarian_erase_primes( indM, N );
00327     step = 3;
00328 }
00329
00342 void hungarian_find_smallest( double &minval, const arma::mat &cost, const arma::ivec &rcov, const arma::ivec &ccov,
00343     const unsigned int &N )
00344 {
00345     for( unsigned int r = 0; r < N; ++r ) {
00346         for( unsigned int c = 0; c < N; ++c ) {
00347             if( rcov.at(r) == 0 && ccov.at(c) == 0 ) {
00348                 if( minval > cost.at(r, c) ) {
00349                     minval = cost.at(r, c);
00350                 }
00351             }
00352         }
00353     }
00354 }
00355
00359 void hungarian_step_6( unsigned int &step, arma::mat &cost, const arma::ivec &rcov, const arma::ivec &ccov,
00360     const unsigned int &N )
00361 {
00362     double minval = std::numeric_limits<double>::max();// DBL_MAX;
00363     hungarian_find_smallest( minval, cost, rcov, ccov, N );
00364     for( unsigned int r = 0; r < N; ++r ) {
00365         for( unsigned int c = 0; c < N; ++c ) {
00366             if( rcov.at(r) == 1 ) {
00367                 cost.at(r, c) += minval;
00368             }
00369             if( ccov.at(c) == 0 ) {
00370                 cost.at(r, c) -= minval;
00371             }
00372         }
00373     }
00374     step = 4;
00375 }
00376
00384 void Hungarian( const arma::mat &assigncost, int dim, TSearchParam sp, double infValue, double resolution,
00385     arma::ivec &rowsol, double &lapcost )
00386 {
00387     const unsigned int N = assigncost.n_rows;
00388     unsigned int step = 1;
00389     int cpath_0 = 0;
00390     int rpath_0 = 0;
00391
00392     // Если ищем максимум - умножим матрицу на -1
00393     arma::mat cost( dim, dim, arma::fill::zeros );
00394     if( sp == TSearchParam::SP_Max ) { // Поиск минимума/максимума
00395         cost = -assigncost;
00396     } else {
00397         cost = assigncost;
00398     }
00399
00400     arma::umat indM(N, N);
00401     arma::ivec rcov(N);
00402     arma::ivec ccov(N);
00403     arma::imat path(2 * N, 2);
00404
00405     indM = arma::zeros<arma::umat>(N, N);
00406     bool done = false;
00407     while( !done ) {
00408         switch( step ) {
00409             case 1:
00410                 hungarian_step_1( step, cost, N );
00411                 break;
00412             case 2:
00413                 hungarian_step_2( step, cost, indM, rcov, ccov, N );
```

```
00414                 break;
00415             case 3:
00416                 hungarian_step_3( step, indM, ccov, N );
00417                 break;
00418             case 4:
00419                 hungarian_step_4( step, cost, indM, rcov, ccov, rpath_0, cpath_0, N );
00420                 break;
00421             case 5:
00422                 hungarian_step_5( step, indM, rcov, ccov, path, rpath_0, cpath_0, N );
00423                 break;
00424             case 6:
00425                 hungarian_step_6( step, cost, rcov, ccov, N );
00426                 break;
00427             case 7:
00428                 done = true;
00429                 break;
00430             default:
00431                 assert( false );
00432         }
00433     }
00434
00435     // Вывод результата
00436     lapcost = 0.0;
00437     for( int i = 0; i < dim; i++ ) {
00438         for( int j = 0; j < dim; j++ ) {
00439             if( indM(i, j) > 0 ) {
00440                 rowsol[i] = j;
00441                 double element_i_j = assigncost(i,j);
00442                 if( !SPML::Compare::AreEqualAbs( element_i_j, infValue, resolution ) ) {
00443                     lapcost += element_i_j;
00444                 }
00445             }
00446         }
00447     }
00448     return;
00449 }
00450
00451 } // end namespace LAP
00452 } // end namespace SPML
```

8.7 Файл hungarian.hpp

Решение задачи о назначениях венгерским методом (Hungarian, Munkres)

```
#include <armadillo>
#include <cassert>
#include <compare.hpp>
#include <searchparam.hpp>
```

Пространства имен

- namespace SPML

  Специальная библиотека программных модулей (СБ ПМ)
- namespace SPML::LAP

  Решение задачи о назначениях

Функции

- void SPML::LAP::Hungarian (const arma::mat &assigncost, int dim, TSearchParam sp, double infValue, double resolution, arma::ivec &rowsol, double &lapcost)

  Венгерский метод решения задачи о назначениях (Метод Мункреса)

8.7.1 Подробное описание

Решение задачи о назначениях венгерским методом (Hungarian, Munkres)

Дата

07.02.23 - создан

Автор

Соболев А.А.

См. определение в файле hungarian.hpp

8.8 hungarian.hpp

См. документацию.

```
00001 //-----------------------------------------------------------------------------------------------------
00010
00011 #ifndef SPML_HUNGARIAN_HPP_
00012 #define SPML_HUNGARIAN_HPP_
00013
00014 #include <armadillo>
00015 #include <cassert>
00016
00017 #include <compare.hpp>
00018 #include <searchparam.hpp>
00019
00020 namespace SPML
00021 {
00022 namespace LAP
00023 {
00036 void Hungarian( const arma::mat &assigncost, int dim, TSearchParam sp, double infValue, double resolution,
00037     arma::ivec &rowsol, double &lapcost );
00038
00039 } // namespace LAP
00040 } // namespace SPML
00041 #endif
```

8.9 Файл jvc_dense.cpp

Решение задачи о назначениях методом JVC для плотных матриц

#include <jvc_dense.hpp>

Пространства имен

- namespace SPML

  Специальная библиотека программных модулей (СБ ПМ)
- namespace SPML::LAP

  Решение задачи о назначениях

Функции

- void SPML::LAP::JVCdense (const arma::mat &assigncost, int dim, TSearchParam sp, double infValue, double resolution, arma::ivec &rowsol, double &lapcost)

  Метод Джонкера-Волгенанта-Кастаньона (Jonker-Volgenant-Castanon) решения задачи о назначениях для плотных матриц

8.9.1 Подробное описание

Решение задачи о назначениях методом JVC для плотных матриц

Дата

18.05.22 - создан

Автор

Соболев А.А.

См. определение в файле jvc_dense.cpp

8.10 jvc_dense.cpp

См. документацию.

```
00001 //-----------------------------------------------------------------------------------------------------------
00010
00011 #include <jvc_dense.hpp>
00012
00013 namespace SPML
00014 {
00015 namespace LAP
00016 {
00017
00018 void JVCdense( const arma::mat &assigncost, int dim, TSearchParam sp, double infValue,
00019     double resolution, arma::ivec &rowsol, double &lapcost )
00020 {
00021     // Если ищем максимум - умножим матрицу на -1
00022     arma::mat cost( dim, dim, arma::fill::zeros );
00023     if( sp == TSearchParam::SP_Max ) { // Поиск минимума/максимума
00024         cost = -assigncost;
00025     } else {
00026         cost = assigncost;
00027     }
00028     bool unassignedfound;
00029     int i = 0, imin = 0, numfree = 0, prvnumfree = 0, f = 0, i0 = 0, k = 0, freerow = 0; // row
00030     int j = 0, j1 = 0, j2 = 0, endofpath = 0, last = 0, low = 0, up = 0;
00031     double dmin = 0.0, h = 0.0, umin = 0.0, usubmin = 0.0, v2 = 0.0; // cost
00032
00033     arma::vec u = arma::vec( dim, arma::fill::zeros );//new double[dim];
00034     arma::vec v = arma::vec( dim, arma::fill::zeros );//new double[dim];
00035     arma::ivec free = arma::ivec( dim, arma::fill::zeros );//new int[dim];       // list of unassigned rows.
00036     arma::ivec collist = arma::ivec( dim, arma::fill::zeros );//new int[dim];    // list of columns to be scanned in various
    ways.
00037     arma::ivec matches = arma::ivec( dim, arma::fill::zeros );//new int[dim];    // counts how many times a row could be
    assigned.
00038     arma::vec d = arma::vec( dim, arma::fill::zeros );//new double[dim];       // 'cost-distance' in augmenting path
    calculation.
00039     arma::ivec pred = arma::ivec( dim, arma::fill::zeros );//new int[dim];       // row-predecessor of column in
    augmenting/alternating path.
00040     arma::ivec colsol = arma::ivec( dim, arma::fill::zeros );//int *colsol = new int[dim];
00041     arma::vec x = arma::vec( dim, arma::fill::zeros );
00042     arma::vec xh = arma::vec( dim, arma::fill::zeros );
00043     arma::ivec vf0 = arma::ivec( dim, arma::fill::zeros );
00044
00045     // init
00046     rowsol.zeros();
```

```
00047     rowsol -= 1;
00048     colsol -= 1;
00049
00050     // COLUMN REDUCTION
00051     for( j = ( dim - 1 ); j >= 0; j-- ) { // reverse order gives better results.
00052         // find minimum cost over rows.
00053         dmin = cost(0,j);
00054         imin = 0;
00055         for( i = 1; i < dim; i++ ) {
00056             if( cost(i,j) < dmin ) {
00057                 dmin = cost(i,j);
00058                 imin = i;
00059             }
00060         }
00061         v(j) = dmin;
00062         matches(imin)++;
00063         if( matches(imin) == 1 ) {
00064             // init assignment if minimum row assigned for first time.
00065             rowsol(imin) = j;
00066             colsol(j) = imin;
00067         } else if( v(j) < v(rowsol(imin)) ) {
00068             int j1 = rowsol(imin);
00069             rowsol(imin) = j;
00070             colsol(j) = imin;
00071             colsol(j1) = -1;
00072         } else {
00073             colsol(j) = -1; // row already assigned, column not assigned.
00074         }
00075     }
00076
00077     // REDUCTION TRANSFER
00078     for( i = 0; i < dim; i++ ) {
00079         if( matches(i) == 0 ) { // fill list of unassigned 'free' rows.
00080             free(numfree) = i;
00081             numfree++;
00082         } else {
00083             if( matches(i) == 1 ) { // transfer reduction from rows that are assigned once.
00084                 j1 = rowsol(i);
00085                 for( j = 0; j < dim; j++ ) {
00086                     x(j) = cost(i,j) - v(j);
00087                 }
00088                 x(j1) = infValue;
00089                 v(j1) = v(j1) - x.min();
00090             }
00091         }
00092     }
00093
00094     // AUGMENTING ROW REDUCTION
00095     int loopcnt = 0; // do-loop to be done twice.
00096     while( loopcnt < 2 ) {
00097         loopcnt++;
00098         //     scan all free rows.
00099         //     in some cases, a free row may be replaced with another one to be scanned next.
00100         k = 0;
00101         prvnumfree = numfree;
00102         numfree = 0;             // start list of rows still free after augmenting row reduction.
00103         while( k < prvnumfree ) {
00104             i = free(k);
00105             k++;
00106             // find minimum and second minimum reduced cost over columns.
00107             for( j = 0; j < dim; j++ ) {
00108                 x(j) = cost(i,j) - v(j);
00109             }
00110
00111             j1 = arma::index_min( x );
00112             umin = x(j1);
00113             x(j1) = infValue;
00114
00115             j2 = arma::index_min( x );
00116             usubmin = x(j2);
00117
00118             i0 = colsol(j1);
00119             if( ( usubmin - umin ) > resolution ) {
00120                 // change the reduction of the minimum column to increase the minimum
00121                 // reduced cost in the row to the subminimum.
00122                 v(j1) = v(j1) - ( usubmin - umin );
00123             } else {
00124                 if( i0 > -1 ) { // minimum and subminimum equal.
00125                     // minimum column j1 is assigned.
00126                     // swap columns j1 and j2, as j2 may be unassigned.
00127                     j1 = j2;
00128                     i0 = colsol(j2);
00129                 }
00130             }
00131             // (re-)assign i to j1, possibly de-assigning an i0.
00132             rowsol(i) = j1;
00133             colsol(j1) = i;
```

```
00134           if( i0 > -1 ) {  // minimum column j1 assigned earlier. // ORIGINAL
00135               //if( umin < ( usubmin + EPS ) ) { // fixed EPS
00136               if( ( usubmin - umin ) > resolution ) {
00137                   // put in current k, and go back to that k.
00138                   // continue augmenting path i - j1 with i0.
00139                   k--;
00140                   free(k) = i0;
00141               } else {
00142                   // no further augmenting reduction possible.
00143                   // store i0 in list of free rows for next phase.
00144                   free(numfree) = i0;
00145                   numfree++;
00146               }
00147           }
00148       }
00149   }
00150
00151   // AUGMENT SOLUTION FOR EACH FREE ROW
00152   for( f = 0; f < numfree; f++ ) {
00153       freerow = free(f); // start row of augmenting path.
00154
00155       // Dijkstra shortest path algorithm.
00156       // runs until unassigned column added to shortest path tree.
00157       for( j = 0; j < dim; j++ ) {
00158           d(j) = cost(freerow,j) - v(j);
00159           pred(j) = freerow;
00160           collist(j) = j; // init column list.
00161       }
00162       low = 0;    // columns in 0..low-1 are ready, now none.
00163       up = 0;     // columns in low..up-1 are to be scanned for current minimum, now none.
00164                   // columns in up..dim-1 are to be considered later to find new minimum,
00165                   // at this stage the list simply contains all columns
00166       unassignedfound = false;
00167       while( !unassignedfound ) {
00168           if( up == low ) { // no more columns to be scanned for current minimum.
00169               last = low - 1;
00170               // scan columns for up..dim-1 to find all indices for which new minimum occurs.
00171               // store these indices between low..up-1 (increasing up).
00172               dmin = d(collist(up));
00173               up++;
00174               for( k = up; k < dim; k++ ) {
00175                   j = collist(k);
00176                   h = d(j);
00177                   if( ( h < dmin ) || ( std::abs( h - dmin ) < resolution ) ) { //
00178   //                  if( h <= dmin ) { // ORIGINAL
00179                       if( h < dmin ) {  // new minimum.
00180                           up = low;   // restart list at index low.
00181                           dmin = h;
00182                       }
00183                       // new index with same minimum, put on undex up, and extend list.
00184                       collist(k) = collist(up);
00185                       collist(up) = j;
00186                       up++;
00187                   }
00188               }
00189               // check if any of the minimum columns happens to be unassigned.
00190               // if so, we have an augmenting path right away.
00191               for( k = low; k < up; k++ ) {
00192                   if( colsol(collist(k)) < 0 ) {
00193                       endofpath = collist(k);
00194                       unassignedfound = true;
00195                       break;
00196                   }
00197               }
00198           }
00199           if( !unassignedfound ) {
00200               // update 'distances' between freerow and all unscanned columns, via next scanned column.
00201               j1 = collist(low);
00202               low++;
00203               i = colsol(j1);
00204
00205               h = cost(i,j1) - v(j1) - dmin;
00206               for( k = up; k < dim; k++ ) {
00207                   j = collist(k);
00208                   v2 = cost(i,j) - v(j) - h;
00209                   if( v2 < d(j) ) {
00210                       pred(j) = i;
00211                       // if( v2 == min ) {  // new column found at same minimum value - ORIGINAL
00212                       if( std::abs( v2 - dmin ) < resolution ) {  // new column found at same minimum value - MY VERSION
00213                           if( colsol(j) < 0 ) {
00214                               // if unassigned, shortest augmenting path is complete.
00215                               endofpath = j;
00216                               unassignedfound = true;
00217                               break;
00218                           } else {
00219                               // else add to list to be scanned right away.
00220                               collist(k) = collist(up);
```

```
00221                             collist(up) = j;
00222                             up++;
00223                         }
00224                     }
00225                     d(j) = v2;
00226                 }
00227             }
00228         }
00229     }
00230
00231     // update column prices.
00232     for( k = 0; k <= last; k++ ) { //for( k = last + 1; k--; ) {
00233         j1 = collist(k);
00234         v(j1) = v(j1) + d(j1) - dmin;
00235     }
00236
00237     // reset row and column assignments along the alternating path.
00238     while( true ) {
00239         i = pred(endofpath);
00240         colsol(endofpath) = i;
00241         j1 = endofpath;
00242         endofpath = rowsol(i);
00243         rowsol(i) = j1; // Вывод результата
00244         if( i == freerow ) {
00245             break;
00246         }
00247     }
00248 }
00249
00250 // calculate lapcost.
00251 lapcost = 0.0;
00252 for( i = 0; i < dim; i++ ) {
00253     j = rowsol(i);
00254     u[i] = cost(i,j) - v(j);
00255     double element_i_j = assigncost(i,j);
00256     if( !SPML::Compare::AreEqualAbs( element_i_j, infValue, resolution ) ) {
00257         lapcost += element_i_j;
00258     }
00259 }
00260 return;
00261 }
00262
00263 } // end namespace LAP
00264 } // end namespace SPML
00266
```

8.11 Файл jvc_dense.hpp

Решение задачи о назначениях методом JVC для плотных матриц

```
#include <armadillo>
#include <compare.hpp>
#include <searchparam.hpp>
```

Пространства имен

- namespace SPML

  Специальная библиотека программных модулей (СБ ПМ)
- namespace SPML::LAP

  Решение задачи о назначениях

Функции

- void SPML::LAP::JVCdense (const arma::mat &assigncost, int dim, TSearchParam sp, double inf↩ Value, double resolution, arma::ivec &rowsol, double &lapcost)

  Метод Джонкера-Волгенанта-Кастаньона (Jonker-Volgenant-Castanon) решения задачи о назначениях для плотных матриц

8.11.1 Подробное описание

Решение задачи о назначениях методом JVC для плотных матриц

Дата

07.02.23 - создан

Автор

Соболев А.А.

См. определение в файле jvc_dense.hpp

8.12 jvc_dense.hpp

См. документацию.

```
00001 //-----------------------------------------------------------------------------------------------------------
00010
00011 #ifndef SPML_JVC_DENSE_HPP_
00012 #define SPML_JVC_DENSE_HPP_
00013
00014 #include <armadillo>
00015
00016 #include <compare.hpp>
00017 #include <searchparam.hpp>
00018
00019 namespace SPML
00020 {
00021 namespace LAP
00022 {
00048 void JVCdense( const arma::mat &assigncost, int dim, TSearchParam sp, double infValue, double resolution,
00049     arma::ivec &rowsol, double &lapcost );
00050
00051 } // namespace LAP
00052 } // namespace SPML
00053 #endif
```

8.13 Файл jvc_sparse.cpp

Решение задачи о назначениях методом JVC для разреженных матриц

#include <jvc_sparse.hpp>

Пространства имен

- namespace SPML

  Специальная библиотека программных модулей (СБ ПМ)
- namespace SPML::LAP

  Решение задачи о назначениях

Функции

- void SPML::LAP::updateDual (int nc, arma::vec &d, arma::vec &v, arma::ivec &todo, int last, double min_)
- void SPML::LAP::updateAssignments (arma::ivec &lab, arma::ivec &y, arma::ivec &x, int j, int i0)
- int SPML::LAP::solveForOneL (std::vector< double > &cc_, const std::vector< int > &kk, const std::vector< int > &first, int l, int nc, arma::vec &d, arma::ivec &ok, arma::ivec &free, arma::vec &v, arma::ivec &lab, arma::ivec &todo, arma::ivec &y, arma::ivec &x, int td1, double resolution, double infValue, bool &fail)
- int SPML::LAP::JVCsparse (const std::vector< double > &cc, const std::vector< int > &kk, const std::vector< int > &first, TSearchParam sp, double infValue, double resolution, arma::ivec &rowsol, double &lapcost)

  Метод Джонкера-Волгенанта-Кастаньона (Jonker-Volgenant-Castanon) решения задачи о назначениях для разреженных матриц
- int SPML::LAP::JVCsparse (const Sparse::CMatrixCSR &csr, TSearchParam sp, double infValue, double resolution, arma::ivec &rowsol, double &lapcost)

  Метод Джонкера-Волгенанта-Кастаньона (Jonker-Volgenant-Castanon) решения задачи о назначениях для разреженных матриц

8.13.1 Подробное описание

Решение задачи о назначениях методом JVC для разреженных матриц

Дата

18.05.22 - создан

Автор

Соболев А.А.

См. определение в файле jvc_sparse.cpp

8.14 jvc_sparse.cpp

См. документацию.

```
00001 //-------------------------------------------------------------------------------------------------
00010
00011 #include <jvc_sparse.hpp>
00012
00013 namespace SPML
00014 {
00015 namespace LAP
00016 {
00017 //-------------------------------------------------------------------------------------------------
00018 void updateDual( int nc, arma::vec &d, arma::vec &v, arma::ivec &todo, int last, double min_ )
00019 {
00020     for( int k = last; k < nc; k++ ) {
00021         int j0 = todo(k);
00022         v(j0) += ( d(j0) - min_ );
00023     }
00024 }
00025
00026 //-------------------------------------------------------------------------------------------------
00027 void updateAssignments( arma::ivec &lab, arma::ivec &y, arma::ivec &x, int j, int i0 )
00028 {
00029     int tmp;
00030     while( true ) {
00031         int i = lab(j);
00032         y(j) = i;
```

```
        //(j, x[i]) = (x[i], j);
        tmp = j;
        j = x[i];
        x[i] = tmp;
        if( i == i0 ) {
            return;
        }
    }
}

//-------------------------------------------------------------------------------------------------------------------
int solveForOneL( std::vector<double> &cc_, const std::vector<int> &kk, const std::vector<int> &first,
    int l, int nc, arma::vec &d, arma::ivec &ok, arma::ivec &free, arma::vec &v, arma::ivec &lab, arma::ivec &todo,
    arma::ivec &y, arma::ivec &x, int td1, double resolution, double infValue, bool &fail )
{
    for( int jp = 0; jp < nc; jp++ ) {
        d(jp) = infValue;
        ok(jp) = 0; // false
    }
    double min_ = infValue;
    int i0 = free(l);
    int j;
    for( int t = first[i0]; t < first[i0 + 1]; t++ ) {
        j = kk[t];
        double dj = cc_[t] - v(j);
        d(j) = dj;
        lab(j) = i0;
//          if( dj <= min_ ) { //POSSIBLE
 FLOWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
//          if( ( dj < min_ ) || ( std::abs( dj - min_ ) < resolution ) ) { //POSSIBLE
 FLOWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
        if( ( ( min_ - dj ) > resolution ) || ( std::abs( dj - min_ ) < resolution ) ) { //POSSIBLE
 FLOWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
            //if( dj < min_ ) {
            if ( ( min_ - dj ) > resolution ) {
                td1 = -1;
                min_ = dj;
            }
            todo(++td1) = j;
        }
    }
    for( int hp = 0; hp <= td1; hp++ ) {
        j = todo(hp);
        if( y(j) == -1 ) {
            updateAssignments( lab, y, x, j, i0 );
            return td1;
        }
        ok(j) = 1;// true
    }
    int td2 = ( nc - 1 );
    int last = nc;
    while( true ) {
        if( td1 < 0 ) {
            fail = true; // FAIL!!!
            return 1;
        }
        int j0 = todo(td1--);
        int i = y(j0);
        todo(td2--) = j0;
        int tp = first[i];
        while( kk[tp] != j0 ) {
            tp++;
        }
        double h = cc_[tp] - v(j0) - min_;
        for( int t = first[i]; t < first[i + 1]; t++ ) {
            j = kk[t];
//              if( !ok(j) ) {
            if( ok(j) == 0 ) { // if( false )
                double vj = cc_[t] - v(j) - h;
//                  if( vj < d(j) ) { // POSSIBLE
 FLOWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
                if( ( d(j) - vj ) > resolution ) { // POSSIBLE
 FLOWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
                    d(j) = vj;
                    lab(j) = i;
//                      if( vj == min_ ) { // POSSIBLE
 FLOWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
                    if( std::abs( vj - min_ ) < resolution ) {
                        if( y[j] == -1 ) {
                            updateDual( nc, d, v, todo, last, min_ );
                            updateAssignments( lab, y, x, j, i0 );
                            return td1;
                        }
                        todo(++td1) = j;
                        ok(j) = 1; // true
                    }
                }
```

```
00114           }
00115       }
00116       if( td1 == -1 ) {
00117           // The original Pascal code uses finite numbers instead of double.PositiveInfinity
00118           // so we need to adjust slightly here.
00119           min_ = infValue;
00120           last = td2 + 1;
00121           for( int jp = 0; jp < nc; jp++ ) {
00122 //              if( ( ( d[jp] < min_ ) || ( std::abs( d[jp] - min_ ) < resolution ) ) && !ok(jp) ) {
00123 //              if( ( ( d[jp] < min_ ) || ( std::abs( d[jp] - min_ ) < resolution ) ) && ( ok(jp) == 0 ) ) {
00124               if( ( std::abs( d[jp] - infValue ) > resolution ) && ( ( ( min_ - d[jp] ) > resolution ) || ( std::abs( d[jp] - min_
      ) < resolution ) ) && ( ok(jp) == 0 ) ) {
00125                   //if( d[jp] < min_ ) {
00126                   if( ( min_ - d[jp] ) > resolution ) {
00127                       td1 = -1;
00128                       min_ = d(jp);
00129                   }
00130                   todo(++td1) = jp;
00131               }
00132           }
00133           for( int hp = 0; hp <= td1; hp++ ) {
00134               j = todo(hp);
00135               if( y(j) == -1 ) {
00136                   updateDual( nc, d, v, todo, last, min_ );
00137                   updateAssignments( lab, y, x, j, i0 );
00138                   return td1;
00139               }
00140               ok(j) = 1;//true;
00141           }
00142       }
00143   }
00144 }
00145
00146 //-----------------------------------------------------------------------------------------------------------------
00147 int JVCsparse( const std::vector<double> &cc, const std::vector<int> &kk, const std::vector<int> &first,
00148     TSearchParam sp, double infValue, double resolution, arma::ivec &rowsol, double &lapcost )
00149 {
00150     // Объявления
00151     int nr = first.size() - 1; // Кол-во строк
00152     int max_kk = -1;
00153     for( unsigned i = 0; i < kk.size(); i++ ) { // Поиск максимального элемента в массиве kk - это и будет кол-во
      столбцов
00154         if( kk[i] > max_kk ) {
00155             max_kk = kk[i];
00156         }
00157     }
00158     int nc = max_kk + 1;// Кол-во столбцов
00159     arma::ivec x = arma::ivec( nr, arma::fill::zeros );
00160     arma::ivec y = arma::ivec( nc, arma::fill::zeros );
00161     arma::vec u = arma::vec( nr, arma::fill::zeros );
00162     arma::vec v = arma::vec( nc, arma::fill::zeros );
00163     arma::vec d = arma::vec( nc, arma::fill::zeros );
00164     arma::ivec ok = arma::ivec( nc, arma::fill::zeros ); // bool: 0 - false, 1 - true (classical c-style)
00165     arma::ivec xinv = arma::ivec( nr, arma::fill::zeros ); // bool: 0 - false, 1 - true (classical c-style)
00166     arma::ivec free = arma::ivec( nr, arma::fill::zeros );
00167     arma::ivec todo = arma::ivec( nc, arma::fill::zeros );
00168     arma::ivec lab = arma::ivec( nc, arma::fill::zeros );
00169     int l0 = 0;
00170
00171     x -= 1;
00172     y -= 1;
00173     free -= 1;
00174     todo -= 1;
00175
00176     // Поиск минимума/максимума
00177     std::vector<double> cc_ = cc;
00178     if( sp == TSearchParam::SP_Max ) {
00179         std::transform( cc_.begin(), cc_.end(), cc_.begin(),
00180             std::bind( std::multiplies<double>(), std::placeholders::_1, -1.0 ) ); // Умножим на -1 для поиска максимума
00181     }
00182
00183     // The initialization steps of LAPJVsp only make sense for square matrices
00184     if( nr == nc ) {
00185         for( int jp = 0; jp < nc; jp++ ) {
00186             v(jp) = infValue;
00187         }
00188         for( int i = 0; i < nr; i++ ) {
00189             for( int t = first[i]; t < first[i + 1]; t++ ) {
00190                 int jp = kk[t];
00191 //                if( cc_[t] < v(jp) ) {
00192                 if( ( v(jp) - cc_[t] ) > resolution ) {
00193                     v(jp) = cc_[t];
00194                     y(jp) = i;
00195                 }
00196             }
00197         }
00198         for( int jp = ( nc - 1 ); jp >= 0; jp-- ) {
```

```
        int i = y(jp);
        if( x(i) == -1 ) {
            x(i) = jp;
        } else {
            y(jp) = -1;
            // Here, the original Pascal code simply inverts the sign of x; as that
            // doesn't play too well with zero-indexing, we explicitly keep track of
            // uniqueness instead.
            xinv(i) = 1;
        }
    }
    int lp = 0;
    for( int i = 0; i < nr; i++ ) {
        if( xinv(i) ) {
            continue;
        }
        if( x(i) != -1 ) {
            double min_ = infValue;
            int j1 = x(i);
            for( int t = first[i]; t < first[i + 1]; t++ ) {
                int jp = kk[t];
                if( jp != j1 ) {
//                      if( ( cc_[t] - v(jp) ) < min_ ) {
                    if( ( min_ - ( cc_[t] - v(jp) ) ) > resolution ) {
                        min_ = ( cc_[t] - v(jp) );
                    }
                }
            }
            u(i) = min_;
            int tp = first[i];
            while( kk[tp] != j1 ) {
                tp++;
            }
            v(j1) = cc_[tp] - min_;
        } else {
            free(lp++) = i;
        }
    }
    for( int tel = 0; tel < 2; tel++ ) {
        int h = 0;
        int l0p = lp;
        lp = 0;
        while( h < l0p ) {
            // Note: In the original Pascal code, the indices of the lowest
            // and second-lowest reduced costs are never reset. This can
            // cause issues for infeasible problems; see https://stackoverflow.com/q/62875232/5085211
            int i = free(h++);

            //----------------------------------------------------------------------------------------------------
            // ORIGINAL SEARCH OF MIN AND SUBMIN
            //----------------------------------------------------------------------------------------------------

            int j0p = -1; // Index of minimum
            int j1p = -1; // Index of subminimum
            double v0 = infValue; // Value of minimum
            double vj = infValue; // Value of subminimum
            for( int t = first[i]; t < first[i + 1]; t++ ) {
                int jp = kk[t];
                double dj = cc_[t] - v(jp);
//                  if( dj < vj ) {
                if( ( vj - dj ) > resolution ) {
//                      if( dj >= v0 ) { // POSSIBLE FLOW!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!1
//                      if( ( dj > v0 ) || ( std::abs( dj - v0 ) < resolution ) ) {
                    if( ( ( dj - v0 ) > resolution ) || ( std::abs( dj - v0 ) < resolution ) ) {
                        vj = dj;
                        j1p = jp;
                    } else {
                        vj = v0;
                        v0 = dj;
                        j1p = j0p;
                        j0p = jp;
                    }
                }
            }
            // If the index of the column with the largest reduced cost has not been
            // set, no assignment is possible for this row.
            if( j0p < 0 ) {
                return 1; // No feasible solution!!!
            }
            int i0 = y(j0p);

            //----------------------------------------------------------------------------------------------------
            // MY SEARCH OF MIN AND SUBMIN
            //----------------------------------------------------------------------------------------------------
//              // find minimum and second minimum reduced cost over columns.
//              int j0p = -1; // Index of minimum
//              int j1p = -1; // Index of subminimum
```

```
00286 //              double v0 = infValue; // Value of minimum
00287 //              double vj = infValue; // Value of subminimum
00288 //              arma::vec i_th_row( nc, arma::fill::zeros );
00289 //              i_th_row.fill( infValue );
00290 //              for( int t = first[i]; t < first[i + 1]; t++ ) {
00291 //                  int jp = kk[t];
00292 //                  double dj = cc_[t] - v(jp);
00293 //                  i_th_row(jp) = dj;
00294 //              }
00295 //              j0p = arma::index_min( i_th_row );  // Index of minimum
00296 //              v0 = i_th_row(j0p);                 // Value of minimum
00297 //              i_th_row(j0p) = infValue;
00298
00299 //              j1p = arma::index_min( i_th_row );  // Index of subminimum
00300 //              vj = i_th_row(j1p);                 // Value of subminimum
00301
00302 //              int i0 = y(j0p);
00303               //-------------------------------------------------------------------------------------------
00304               // ORIGINAL
00305               //-------------------------------------------------------------------------------------------
00306
00307               u(i) = vj;
00308 //              if( v0 < vj ) { // FIX
00309               if( ( vj - v0 ) > resolution ) { // MY
00310                   v(j0p) += ( v0 - vj );
00311               } else if( i0 != -1 ) {
00312                   j0p = j1p;
00313                   i0 = y(j0p);
00314               }
00315               x(i) = j0p;
00316               y(j0p) = i;
00317               if( i0 != -1 ) {
00318 //                  if( v0 < vj ) { // FIX
00319                   if( ( vj - v0 ) > resolution ) { // MY
00320                       free(--h) = i0;
00321                   } else {
00322                       free(lp++) = i0;
00323                   }
00324               }
00325
00326               //-------------------------------------------------------------------------------------------
00327               // MY SEARCH OF MIN AND SUBMIN
00328               //-------------------------------------------------------------------------------------------
00329 //              u(i) = vj;
00330 //              if( ( vj - v0 ) > resolution ) { // if( v0 < vj )
00331 //                  // change the reduction of the minimum column to increase the minimum
00332 //                  // reduced cost in the row to the subminimum.
00333 //                  v(j0p) += ( v0 - vj );
00334 //              } else {
00335 //                  if( i0 > -1 ) { // minimum and subminimum equal.
00336 //                      // minimum column j1 is assigned.
00337 //                      // swap columns j1 and j2, as j2 may be unassigned.
00338 //                      j0p = j1p;
00339 //                      i0 = y(j0p);
00340 //                  }
00341 //              }
00342 //              // (re-)assign i to j1, possibly de-assigning an i0.
00343 //              x(i) = j0p;
00344 //              y(j0p) = i;
00345 //              if( i0 > -1 ) {
00346 //                  if( ( vj - v0 ) > resolution ) { // FIX
00347 //                      free(--h) = i0;
00348 //                  } else {
00349 //                      free(lp++) = i0;
00350 //                  }
00351 //              }
00352               //-------------------------------------------------------------------------------------------
00353           }
00354       } // end for( int tel = 0; tel < 2; tel++ )
00355       l0 = lp;
00356   } else { // end if( nr == nc )
00357       l0 = nr;
00358       for( int i = 0; i < nr; i++ ) {
00359           free(i) = i;
00360       }
00361   }
00362   int td1 = -1;
00363   for( int l = 0; l < l0; l++ ) {
00364       bool fail = false;
00365       td1 = solveForOneL( cc_, kk, first, l, nc, d, ok, free, v, lab, todo, y, x, td1, resolution, infValue, fail );
00366       if( fail ) {
00367           return 1;
00368       }
00369   }
00370   // Prapare output - rowsol and lapcost.
00371   lapcost = 0.0;
00372   for( int i = 0; i < nr; i++ ) { // i - row index
```

```
00373         rowsol[i] = x[i];
00374         const int j_ = rowsol[i]; // j - col index
00375         const int start = first[i];
00376         const int end = first[i+1];
00377         for( int j = start; j < end; j++ ) {
00378             if( j_ == kk[j] ) {
00379                 lapcost += cc[j];
00380                 break;
00381             }
00382         }
00383     }
00384     return 0;
00385 }
00386
00387 //---------------------------------------------------------------------------------------------------------
00388 int JVCsparse( const Sparse::CMatrixCSR &csr, TSearchParam sp, double infValue, double resolution, arma::ivec &rowsol,
00389     double &lapcost )
00390 {
00391     int result = JVCsparse( csr.csr_val, csr.csr_kk, csr.csr_first, sp, infValue, resolution, rowsol, lapcost );
00392     return result;
00393 }
00394
00395 }
00396 }
```

8.15 Файл jvc_sparse.hpp

Решение задачи о назначениях методом JVC для разреженных матриц

```
#include <armadillo>
#include <compare.hpp>
#include <searchparam.hpp>
#include <sparse.hpp>
```

Пространства имен

- namespace SPML

  Специальная библиотека программных модулей (СБ ПМ)
- namespace SPML::LAP

  Решение задачи о назначениях

Функции

- int SPML::LAP::JVCsparse (const std::vector< double > &cc, const std::vector< int > &kk, const std::vector< int > &first, TSearchParam sp, double infValue, double resolution, arma::ivec &rowsol, double &lapcost)

  Метод Джонкера-Волгенанта-Кастаньона (Jonker-Volgenant-Castanon) решения задачи о назначениях для разреженных матриц
- int SPML::LAP::JVCsparse (const Sparse::CMatrixCSR &csr, TSearchParam sp, double infValue, double resolution, arma::ivec &rowsol, double &lapcost)

  Метод Джонкера-Волгенанта-Кастаньона (Jonker-Volgenant-Castanon) решения задачи о назначениях для разреженных матриц

8.15.1 Подробное описание

Решение задачи о назначениях методом JVC для разреженных матриц

Дата

07.02.23 - создан

Автор

Соболев А.А.

См. определение в файле jvc_sparse.hpp

8.16 jvc_sparse.hpp

См. документацию.

```
00001 //-----------------------------------------------------------------------------------------------------------
00010
00011 #ifndef SPML_JVC_SPARSE_HPP_
00012 #define SPML_JVC_SPARSE_HPP_
00013
00014 #include <armadillo>
00015
00016 #include <compare.hpp>
00017 #include <searchparam.hpp>
00018 #include <sparse.hpp>
00019
00020 namespace SPML
00021 {
00022 namespace LAP
00023 {
00038 int JVCsparse( const std::vector<double> &cc, const std::vector<int> &kk, const std::vector<int> &first,
00039     TSearchParam sp, double infValue, double resolution, arma::ivec &rowsol, double &lapcost );
00040
00053 int JVCsparse( const Sparse::CMatrixCSR &csr, TSearchParam sp, double infValue, double resolution, arma::ivec &rowsol,
00054     double &lapcost );
00055
00056 } // namespace LAP
00057 } // namespace SPML
00058 #endif
```

8.17 Файл lap.hpp

Решение задачи о назначениях (стандартная линейная дискретная оптимизационная задача)

#include <hungarian.hpp>
#include <jvc_dense.hpp>
#include <jvc_sparse.hpp>
#include <mack.hpp>
#include <murty.hpp>
#include <seqextr.hpp>

8.17.1 Подробное описание

Решение задачи о назначениях (стандартная линейная дискретная оптимизационная задача)

Подключение всех методов решения задачи о назначениях:

- метод JVC (Jonker-Volgenant-Castanon), в том числе JVCsparse
- метод Murty (k-best решений на основе JVCsparse)
- метод Мака
- Венгерский метод
- метод последовательного выбора экстремума

Дата

21.01.25 - создан

Автор

Соболев А.А.

См. определение в файле lap.hpp

8.18 lap.hpp

См. документацию.

```
00001 //-----------------------------------------------------------------------------------------------------------
00016
00017 #ifndef SPML_LAP_HPP_
00018 #define SPML_LAP_HPP_
00019
00020 #include <hungarian.hpp>
00021 #include <jvc_dense.hpp>
00022 #include <jvc_sparse.hpp>
00023 #include <mack.hpp>
00024 #include <murty.hpp>
00025 #include <seqextr.hpp>
00026
00027 /*
00028 namespace SPML /// Специальная библиотека программных модулей (СБ ПМ)
00029 {
00030 namespace LAP /// Решение задачи о назначениях
00031 {
00032 //-----------------------------------------------------------------------------------------------------------
00036 enum TSearchParam
00037 {
00038     SP_Min, ///< Поиск минимума
00039     SP_Max  ///< Поиск максимума
00040 };
00041
00042 //-----------------------------------------------------------------------------------------------------------
00055 void SequentalExtremum( const arma::mat &assigncost, TSearchParam sp, double infValue, double resolution,
00056     arma::ivec &rowsol, double &lapcost );
00057
00070 void SequentalExtremum( const Sparse::CMatrixCOO &assigncost, TSearchParam sp, double infValue, double resolution,
00071     arma::ivec &rowsol, double &lapcost );
00072
00073 //-----------------------------------------------------------------------------------------------------------
00099 void JVCdense( const arma::mat &assigncost, int dim, TSearchParam sp, double infValue, double resolution,
00100     arma::ivec &rowsol, double &lapcost );
00101
00102 //-----------------------------------------------------------------------------------------------------------
00117 int JVCsparse( const std::vector<double> &cc, const std::vector<int> &kk, const std::vector<int> &first,
00118     TSearchParam sp, double infValue, double resolution, arma::ivec &rowsol, double &lapcost );
00119
00132 int JVCsparse( const Sparse::CMatrixCSR &csr, TSearchParam sp, double infValue, double resolution, arma::ivec &rowsol,
00133     double &lapcost );
00134
```

```
00135 //-----------------------------------------------------------------------------------------------------------------
00148 void Mack( const arma::mat &assigncost, int dim, TSearchParam sp, double infValue, double resolution, arma::ivec
       &rowsol,
00149     double &lapcost );
00150
00151 //-----------------------------------------------------------------------------------------------------------------
00164 void Hungarian( const arma::mat &assigncost, int dim, TSearchParam sp, double infValue, double resolution,
00165     arma::ivec &rowsol, double &lapcost );
00166
00167 //-----------------------------------------------------------------------------------------------------------------
00182 int Murty_JVCsparse( const std::vector<double> &cc, const std::vector<int> &kk, const std::vector<int> &first,
00183     TSearchParam sp, double infValue, double resolution, arma::ivec &rowsol, double &lapcost,
00184     arma::vec *u_init = nullptr, arma::vec *v_init = nullptr, arma::vec *u_out = nullptr, arma::vec *v_out = nullptr );
00185
00198 int Murty_JVCsparse( const Sparse::CMatrixCSR &csr, TSearchParam sp, double infValue, double resolution, arma::ivec
       &rowsol,
00199     double &lapcost,
00200     arma::vec *u_init = nullptr, arma::vec *v_init = nullptr, arma::vec *u_out = nullptr, arma::vec *v_out = nullptr );

00201
00202
00203 } // end namespace LAP
00204 } // end namespace SPML
00205
00206 */
00207
00208 #endif // SPML_LAP_H
```

8.19 Файл lap_constraints.hpp

Для обеспечения работы метода Murty.

#include <vector>
#include <utility>

Классы

- struct SPML::LAP::LapConstraints

Пространства имен

- namespace SPML

  Специальная библиотека программных модулей (СБ ПМ)
- namespace SPML::LAP

  Решение задачи о назначениях

Функции

- bool SPML::LAP::lap_is_allowed (int i, int j)

Переменные

- thread_local const LapConstraints * SPML::LAP::g_lap_constraints = nullptr

8.19.1 Подробное описание

Для обеспечения работы метода Murty.

Дата

21.01.25

Автор

Соболев А.А.

См. определение в файле lap_constraints.hpp

8.20 lap_constraints.hpp

См. документацию.

```
00001 //-----------------------------------------------------------------------------------------------------------
00010
00011 #ifndef SPML_LAP_CONSTRAITS_HPP_
00012 #define SPML_LAP_CONSTRAITS_HPP_
00013
00014 #include <vector>
00015 #include <utility>
00016
00017 namespace SPML
00018 {
00019 namespace LAP
00020 {
00021
00022 struct LapConstraints
00023 {
00024     const int* fixed_col = nullptr; // row -> fixed column or -1
00025     const std::vector<std::pair<int,int>>* banned = nullptr;
00026
00027     inline bool isAllowed(int i, int j) const
00028     {
00029         if( fixed_col && fixed_col[i] != -1 && fixed_col[i] != j )
00030             return false;
00031         if( banned ) {
00032             for( const auto& p : *banned )
00033                 if( p.first == i && p.second == j )
00034                     return false;
00035         }
00036         return true;
00037     }
00038 };
00039
00040 extern thread_local const LapConstraints* g_lap_constraints;
00041
00042 inline bool lap_is_allowed( int i, int j )
00043 {
00044     if( !g_lap_constraints ) {
00045         return true;
00046     }
00047     return g_lap_constraints->isAllowed( i, j );
00048 }
00049
00050 } // namespace LAP
00051 } // namespace SPML
00052 #endif
```

8.21 Файл mack.cpp

Решение задачи о назначениях методом Мака

#include <mack.hpp>

Пространства имен

- namespace SPML

  Специальная библиотека программных модулей (СБ ПМ)
- namespace SPML::LAP

  Решение задачи о назначениях

Функции

- void SPML::LAP::Mack (const arma::mat &assigncost, int dim, TSearchParam sp, double infValue, double resolution, arma::ivec &rowsol, double &lapcost)

  Метод Мака решения задачи о назначениях

8.21.1 Подробное описание

Решение задачи о назначениях методом Мака

Дата

18.05.22 - создан

Автор

Соболев А.А.

См. определение в файле mack.cpp

8.22 mack.cpp

См. документацию.

```
00001 //-----------------------------------------------------------------------------------------------------------
00010
00011 #include <mack.hpp>
00012
00013 namespace SPML
00014 {
00015 namespace LAP
00016 {
00017
00018 void Mack( const arma::mat &assigncost, int dim, TSearchParam sp, double infValue, double resolution, arma::ivec
       &rowsol,
00019     double &lapcost )
00020 {
00021     double* cost = new double[dim*dim];
00022     for( int i = 0; i < dim; i++ ) {
00023         for( int j = 0; j < dim; j++ ) {
00024             if( sp == TSearchParam::SP_Max ) {
00025                 cost[dim*i+j] = -assigncost(i,j); // Поиск максимума
00026             } else {
00027                 cost[dim*i+j] = assigncost(i,j); // Поиск минимума
00028             }
00029         }
00030     }
00031
00032     double* ma = new double[dim+1];
00033     double* mb = new double[dim+1];
00034     int* ip = new int[dim+1];
00035     int* im = new int[dim+1];
00036     int* ic = new int[(dim+1) * (dim+1)];
00037     int* lr = new int[dim+1];
00038
```

```
00039     int* jr = new int[dim+1];
00040     int* jm = new int[dim+1];
00041     int* jk = new int[dim+1];
00042     int* jv = new int[dim+1]; // result of algorithm (needs to be converted from 1...N fortran format to 0...(N-1) C#
          format)
00043     int* nm = new int[dim+1];
00044
00045     // clear memory
00046     memset( ma, 0, static_cast<unsigned long long>( dim + 1 ) * sizeof(double) );
00047     memset( mb, 0, static_cast<unsigned long long>( dim + 1 ) * sizeof(double) );
00048     memset( ip, 0, static_cast<unsigned long long>( dim + 1 ) * sizeof(int) );
00049     memset( im, 0, static_cast<unsigned long long>( dim + 1 ) * sizeof(int) );
00050     memset( ic, 0, static_cast<unsigned long long>( ( dim + 1 ) * ( dim + 1 ) ) * sizeof(int) );
00051     memset( lr, 0, static_cast<unsigned long long>( dim + 1 ) * sizeof(int) );
00052     memset( jr, 0, static_cast<unsigned long long>( dim + 1 ) * sizeof(int) );
00053     memset( jm, 0, static_cast<unsigned long long>( dim + 1 ) * sizeof(int) );
00054     memset( jk, 0, static_cast<unsigned long long>( dim + 1 ) * sizeof(int) );
00055     memset( jv, 0, static_cast<unsigned long long>( dim + 1 ) * sizeof(int) );
00056     memset( nm, 0, static_cast<unsigned long long>( dim + 1 ) * sizeof(int) );
00057
00058     double rim = 0;
00059     double riz = 0;
00060     double riv = 0;
00061
00062     int nc = 0;
00063
00064     int i = 0;
00065     int il = 0;
00066     int ir = 0;
00067     int iw = 0;
00068     int iz = 0;
00069     int icj = 0;
00070     int ilr = 0;
00071     int iip = 0;
00072
00073     int l = 0;
00074     int ls = 0;
00075     int k = 0;
00076
00077     int j = 0;
00078     int jc = 0;
00079     int jd = 0;
00080     int jp = 0;
00081     int jq = 0;
00082     int ju1 = 0;
00083     int jx = 0;
00084     int jy = 0;
00085
00086     double rma = 1e10;
00087
00088     // 2
00089     for( i = 1; i <= dim; i++ ) {
00090         rim = rma;
00091         for( j = 1; j <= dim; j++ ) {
00092             riz = cost[dim*(i-1)+(j-1)]; //riz = p[i, j];
00093             if( riz > rim ) {
00094                 continue;
00095             }
00096             rim = riz;
00097             l = j;
00098         }
00099             nm[l] = nm[l] + 1;
00100             k = nm[l];
00101             ic[dim*(l-1)+(k-1)] = i; //ic[l, k] = i;
00102             ma[i] = rim;
00103             jr[i] = l;
00104     }
00105     bool isJN = false;
00106     for( ; ; ) {
00107         j = 0;
00108         for( ; ; ) {
00109             j = j + 1;
00110             if( j > dim ) {
00111                 isJN = true;
00112                 break;
00113             }
00114             if( nm[j] >= 2 ) {
00115                 break;
00116             }
00117         }
00118         if( isJN ) {
00119             break;
00120         }
00121         ju1 = nm[j];
00122
00123         for( i = 1; i <= dim; i++ ) {
00124             ip[i] = ic[dim*(j-1)+(i-1)]; //ip[i] = ic[j, i];
```

```
00125       }
00126       nc = 1;
00127       lr[1] = j;
00128       jk[j] = 1;
00129       mb[j] = 0;
00130       for( ; ; ) {
00131           riv = rma;
00132           // 4
00133           for( k = 1; k <= ju1; k++ ) {
00134               i = ip[k];
00135               for( jd = 1; jd <= dim; jd++ ) {
00136                   if( jk[jd] == 1 ) {
00137                       continue;
00138                   }
00139                   riz = cost[dim*(i-1)+(jd-1)] - ma[i]; //riz = p[i, jd] - ma[i];
00140                   if( riz > riv ) {
00141                       continue;
00142                   }
00143                   riv = riz;
00144                   jc = jd;
00145                   ir = i;
00146               }
00147           }
00148           // 5
00149           for( jx = 1; jx <= nc; jx++ ) {
00150               ilr = lr[jx];
00151               mb[ilr] = mb[ilr] + riv;
00152           }
00153           for( k = 1; k <= ju1; k++ ) {
00154               iip = ip[k];
00155               ma[iip] = ma[iip] + riv;
00156           }
00157           mb[jc] = 0;
00158           jk[jc] = 1;
00159           nc = nc + 1;
00160           lr[nc] = jc;
00161           im[jc] = ir;
00162           jm[ir] = jc;
00163           jy = nm[jc];
00164           if( jy != 0 ) {
00165               for( jx = 1; jx <= jy; jx++ ) {
00166                   ju1 = ju1 + 1;
00167                   ip[ju1] = ic[dim*(jc-1)+(jx-1)]; // ip[ju1] = ic[jc, jx];
00168               }
00169               continue;
00170           }
00171           break;
00172       }
00173       for( jx = 1; jx <= nc; jx++ ) {
00174           ls = lr[jx];
00175           jk[ls] = 0;
00176           for( i = 1; i <= dim; i++ ) {
00177               cost[dim*(i-1)+(ls-1)] = cost[dim*(i-1)+(ls-1)] + mb[ls]; //p[i, ls] = p[i, ls] + mb[ls];
00178           }
00179       }
00180       nm[jc] = 1;
00181       ic[dim*(jc-1)+1-1] = ir; // ic[jc, 1] = ir;
00182       for( ; ; ) {
00183           jp = jr[ir];
00184           jr[ir] = jc;
00185           iw = 0;
00186           jq = nm[jp];
00187           for( il = 1; il <= jq; il++ ) {
00188               iz = ic[dim*(jp-1)+(il-1)]; // iz = ic[jp, il];
00189               if( iz != ir ) {
00190                   iw = iw + 1;
00191                   ic[dim*(jp-1)+(iw-1)] = ic[dim*(jp-1)+(il-1)];// ic[jp, iw] = ic[jp, il];
00192               }
00193           }
00194           if( jq > 1 ) {
00195               break;
00196           }
00197           ir = im[jp];
00198           jc = jp;
00199           ic[dim*(jp-1)+(jq-1)] = ir;//ic[jp, jq] = ir;
00200       }
00201       nm[jp] = jq - 1;
00202   }
00203   for( j = 1; j <= dim; j++ ) {
00204       icj = ic[dim*(j-1)+1-1];// icj = ic[j, 1];
00205       jv[icj] = j;
00206   }
00207
00208   // Вывод результата
00209   j = 0;
00210   lapcost = 0.0;
00211   for( int i = 0; i < dim; i++ ) {
```

```
00212         j = jv[i+1]-1;
00213         rowsol[i] = j; // в i-ой строке j-ый элемент
00214         double element_i_j = assigncost(i,j);
00215         if( !SPML::Compare::AreEqualAbs( element_i_j, infValue, resolution ) ) {
00216             lapcost += element_i_j;
00217         }
00218     }
00219
00220     // освобождаем память
00221     delete[] cost;
00222     delete[] ma;
00223     delete[] mb;
00224     delete[] ip;
00225     delete[] im;
00226     delete[] ic;
00227     delete[] lr;
00228     delete[] jr;
00229     delete[] jm;
00230     delete[] jk;
00231     delete[] jv;
00232     delete[] nm;
00233     return;
00234 }
00235
00236 } // end namespace LAP
00237 } // end namespace SPML
00239
```

8.23 Файл mack.hpp

Решение задачи о назначениях методом Мака для плотных матриц

#include <armadillo>
#include <compare.hpp>
#include <searchparam.hpp>

Пространства имен

- namespace SPML

  Специальная библиотека программных модулей (СБ ПМ)
- namespace SPML::LAP

  Решение задачи о назначениях

Функции

- void SPML::LAP::Mack (const arma::mat &assigncost, int dim, TSearchParam sp, double infValue, double resolution, arma::ivec &rowsol, double &lapcost)

  Метод Мака решения задачи о назначениях

8.23.1 Подробное описание

Решение задачи о назначениях методом Мака для плотных матриц

Дата

07.02.23 - создан

Автор

Соболев А.А.

См. определение в файле mack.hpp

8.24 mack.hpp

См. документацию.

```
00001 //-----------------------------------------------------------------------------------------------------------
00010
00011 #ifndef SPML_MACK_HPP_
00012 #define SPML_MACK_HPP_
00013
00014 #include <armadillo>
00015
00016 #include <compare.hpp>
00017 #include <searchparam.hpp>
00018
00019 namespace SPML
00020 {
00021 namespace LAP
00022 {
00035 void Mack( const arma::mat &assigncost, int dim, TSearchParam sp, double infValue, double resolution, arma::ivec
       &rowsol,
00036     double &lapcost );
00037
00038 } // namespace LAP
00039 } // namespace SPML
00040 #endif
```

8.25 Файл murty.cpp

Решение k-best задачи о назначениях методом Murty.

#include <murty.hpp>

Пространства имен

- namespace SPML

  Специальная библиотека программных модулей (СБ ПМ)
- namespace SPML::LAP

  Решение задачи о назначениях

8.25.1 Подробное описание

Решение k-best задачи о назначениях методом Murty.

Единичная задача о назначениях решается методом JVC для разреженных матриц

Дата

21.01.26 - создан

Автор

Соболев А.А.

См. определение в файле murty.cpp

8.26 murty.cpp

См. документацию.

```
00001 //-----------------------------------------------------------------------------------------------------------------
00011
00012 #include <murty.hpp>
00013
00014 namespace SPML
00015 {
00016 namespace LAP
00017 {
00018
00019 static bool MurtyCmpMax( const MurtyNode &a, const MurtyNode &b ) {
00020     return a.lb < b.lb; // для поиска Max
00021 }
00022
00023 static bool MurtyCmpMin( const MurtyNode &a, const MurtyNode &b ) {
00024     return a.lb > b.lb; // для поиска Min
00025 }
00026
00027 Murty::Murty( const Sparse::CMatrixCSR& csr, TSearchParam sp, double inf, double res )
00028     : csr_( csr )
00029     , sp_( sp )
00030     , inf_( inf )
00031     , res_( res )
00032     , initialized_( false )
00033     , pq_( ( sp == TSearchParam::SP_Max ) ? MurtyCmpMax : MurtyCmpMin )
00034 {}
00035
00036 MurtyNode Murty::solveNode( const MurtyNode* parent, int split_row )
00037 {
00038     MurtyNode n;
00039     n.split_from = split_row + 1;
00040     const int nr = csr_.n_rows();
00041
00042     n.fixed_col.assign( nr, -1 );
00043
00044     if( parent ) {
00045         // OLD
00046         // n.fixed_col = parent->fixed_col;
00047
00048         // NEW
00049         // Фиксируем все строки < split_row
00050         for( int i = 0; i < split_row; ++i ) {
00051             n.fixed_col[i] = parent->sol.x(i);
00052         }
00053
00054         // Копируем предыдущие bans
00055         n.banned = parent->banned;
00056
00057         // Запрещаем текущее ребро
00058         int c = parent->sol.x( split_row );
00059         n.banned.emplace_back( split_row, c );
00060     }
00061
00062     LapConstraints lc;
00063     lc.fixed_col = n.fixed_col.data();
00064     lc.banned = &n.banned;
00065     g_lap_constraints = &lc;
00066
00067     arma::ivec x( nr );
00068     arma::vec u_out( nr ), v_out( csr_.n_cols() );
00069     double cost = 0.0;
00070
00071     arma::vec* u_init = nullptr;
00072     arma::vec* v_init = nullptr;
00073
00074     if( parent ) {
00075         u_init = const_cast<arma::vec*>( &parent->sol.u );
00076         v_init = const_cast<arma::vec*>( &parent->sol.v );
00077     }
00078
00079     // Вызов модифицированного метода JVC с "теплым" стартом
00080     int rc = Murty_JVCsparse( csr_, sp_, inf_, res_, x, cost, u_init, v_init, &u_out, &v_out );
00081
00082     g_lap_constraints = nullptr;
00083
00084     if( rc != 0 || cost >= inf_ ) {
00085         n.sol.cost = inf_;
00086         return n;
00087     }
00088
00089     n.sol.x = x;
00090     n.sol.u = std::move( u_out );
00091     n.sol.v = std::move( v_out );
```

```
00092     n.sol.cost = cost;
00093     n.lb = cost;
00094     return n;
00095 }
00096
00097 bool Murty::findNext( MurtySolution& out )
00098 {
00099     if( !initialized_ ) {
00100         MurtyNode root = solveNode( nullptr, -1 );
00101         if( root.sol.cost >= inf_ ) {
00102             return false;
00103         }
00104         pq_.push( root );
00105         initialized_ = true;
00106     }
00107
00108     if (pq_.empty()) return false;
00109
00110     MurtyNode best = pq_.top(); // Лучший - на первой позиции!
00111     pq_.pop();
00112     out = best.sol;
00113
00114     const int nr = best.sol.x.n_elem;
00115     // for (int i = 0; i < nr; ++i) {
00116     for( int i = best.split_from; i < nr; ++i ) {
00117         MurtyNode child = solveNode( &best, i );
00118         if( child.sol.cost < inf_ ) {
00119             pq_.push( child );
00120         }
00121     }
00122     return true;
00123 }
00124
00125 } // namespace LAP
00126 } // namespace SPML
```

8.27 Файл murty.hpp

Решение k-best задачи о назначениях методом Murty.

#include <armadillo>
#include <queue>
#include <vector>
#include <lap_constraints.hpp>
#include <sparse.hpp>
#include <searchparam.hpp>
#include <murty_jvc_sparse.hpp>

Классы

- struct SPML::LAP::MurtySolution

  Решение метода Murty.
- struct SPML::LAP::MurtyNode

  Узел решения
- class SPML::LAP::Murty

  Класс решения K-best задачи методом Murty.

Пространства имен

- namespace SPML

  Специальная библиотека программных модулей (СБ ПМ)
- namespace SPML::LAP

  Решение задачи о назначениях

8.27.1 Подробное описание

Решение k-best задачи о назначениях методом Murty.

Единичная задача о назначениях решается методом JVC для разреженных матриц

Дата

21.01.26 - создан

Автор

Соболев А.А.

См. определение в файле murty.hpp

8.28 murty.hpp

См. документацию.

```
00001 //---------------------------------------------------------------------------------------------------------------
00011
00012 #ifndef SPML_MURTY_HPP_
00013 #define SPML_MURTY_HPP_
00014
00015 #include <armadillo>
00016 #include <queue>
00017 #include <vector>
00018
00019 #include <lap_constraints.hpp>
00020 #include <sparse.hpp>
00021 #include <searchparam.hpp>
00022 #include <murty_jvc_sparse.hpp>
00023
00024 namespace SPML
00025 {
00026 namespace LAP
00027 {
00028 //---------------------------------------------------------------------------------------------------------------
00032 struct MurtySolution {
00033     arma::ivec x;
00034     arma::vec u;
00035     arma::vec v;
00036     double cost;
00037 };
00038
00039 //---------------------------------------------------------------------------------------------------------------
00043 struct MurtyNode {
00044     std::vector<int> fixed_col;
00045     std::vector<std::pair<int,int» banned;
00046     MurtySolution sol;
00047     double lb;
00048     int split_from = 0;
00049 };
00050
00051 //---------------------------------------------------------------------------------------------------------------
00055 class Murty
00056 {
00057 public:
00065     Murty( const Sparse::CMatrixCSR& csr, TSearchParam sp, double inf, double res );
00066
00072     bool findNext( MurtySolution &out );
00073
00074 private:
00075     MurtyNode solveNode( const MurtyNode* parent, int split_row );
00076
00077     const Sparse::CMatrixCSR& csr_;
00078     TSearchParam sp_;
00079     double inf_;
00080     double res_;
00081     bool initialized_;
00082     std::priority_queue<
```

```
00083         MurtyNode,
00084         std::vector<MurtyNode>,
00085         bool(*)( const MurtyNode&, const MurtyNode& )
00086     > pq_;
00087 };
00088
00089 } // namespace LAP
00090 } // namespace SPML
00091 #endif
00092
```

8.29 Файл murty_jvc_sparse.cpp

```
#include <murty_jvc_sparse.hpp>
#include <lap_constraints.hpp>
```

Пространства имен

- namespace SPML

  Специальная библиотека программных модулей (СБ ПМ)
- namespace SPML::LAP

  Решение задачи о назначениях

Функции

- void SPML::LAP::Murty_updateDual (int nc, arma::vec &d, arma::vec &v, arma::ivec &todo, int last, double min_)
- void SPML::LAP::Murty_updateAssignments (arma::ivec &lab, arma::ivec &y, arma::ivec &x, int j, int i0)
- int SPML::LAP::Murty_solveForOneL (std::vector< double > &cc_, const std::vector< int > &kk, const std::vector< int > &first, int l, int nc, arma::vec &d, arma::ivec &ok, arma::ivec &free, arma::vec &v, arma::ivec &lab, arma::ivec &todo, arma::ivec &y, arma::ivec &x, int td1, double resolution, double infValue, bool &fail)
- int SPML::LAP::Murty_JVCsparse (const std::vector< double > &cc, const std::vector< int > &kk, const std::vector< int > &first, TSearchParam sp, double infValue, double resolution, arma::ivec &rowsol, double &lapcost, arma::vec *u_init=nullptr, arma::vec *v_init=nullptr, arma::vec *u_out=nullptr, arma::vec *v_out=nullptr)

  Метод Джонкера-Волгенанта-Кастаньона (Jonker-Volgenant-Castanon) решения задачи о назначениях для разреженных матриц
- int SPML::LAP::Murty_JVCsparse (const Sparse::CMatrixCSR &csr, TSearchParam sp, double infValue, double resolution, arma::ivec &rowsol, double &lapcost, arma::vec *u_init=nullptr, arma::vec *v_init=nullptr, arma::vec *u_out=nullptr, arma::vec *v_out=nullptr)

  Метод Джонкера-Волгенанта-Кастаньона (Jonker-Volgenant-Castanon) решения задачи о назначениях для разреженных матриц

8.30 murty_jvc_sparse.cpp

См. документацию.

```
//----------------------------------------------------------------------------------------------------------------

#include <murty_jvc_sparse.hpp>
#include <lap_constraints.hpp>

namespace SPML
{
namespace LAP
{

thread_local const LapConstraints* g_lap_constraints = nullptr;

//----------------------------------------------------------------------------------------------------------------
void Murty_updateDual( int nc, arma::vec &d, arma::vec &v, arma::ivec &todo, int last, double min_ )
{
    for( int k = last; k < nc; k++ ) {
        int j0 = todo(k);
        v(j0) += ( d(j0) - min_ );
    }
}

//----------------------------------------------------------------------------------------------------------------
void Murty_updateAssignments( arma::ivec &lab, arma::ivec &y, arma::ivec &x, int j, int i0 )
{
    int tmp;
    while( true ) {
        int i = lab(j);
        y(j) = i;
        //(j, x[i]) = (x[i], j);
        tmp = j;
        j = x[i];
        x[i] = tmp;
        if( i == i0 ) {
            return;
        }
    }
}

//----------------------------------------------------------------------------------------------------------------
int Murty_solveForOneL( std::vector<double> &cc_, const std::vector<int> &kk, const std::vector<int> &first,
    int l, int nc, arma::vec &d, arma::ivec &ok, arma::ivec &free, arma::vec &v, arma::ivec &lab, arma::ivec &todo,
    arma::ivec &y, arma::ivec &x, int td1, double resolution, double infValue, bool &fail )
{
    for( int jp = 0; jp < nc; jp++ ) {
        d(jp) = infValue;
        ok(jp) = 0; // false
    }
    double min_ = infValue;
    int i0 = free(l);
    int j;
    for( int t = first[i0]; t < first[i0 + 1]; t++ ) {
        j = kk[t];
        if( !lap_is_allowed( i0, j ) ) { // Fix Murty
            continue;
        }
        double dj = cc_[t] - v(j);
        d(j) = dj;
        lab(j) = i0;
//          if( dj <= min_ ) { //POSSIBLE
  FLOWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
//          if( ( dj < min_ ) || ( std::abs( dj - min_ ) < resolution ) ) { //POSSIBLE
  FLOWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
        if( ( ( min_ - dj ) > resolution ) || ( std::abs( dj - min_ ) < resolution ) ) { //POSSIBLE
  FLOWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
            //if( dj < min_ ) {
            if ( ( min_ - dj ) > resolution ) {
                td1 = -1;
                min_ = dj;
            }
            todo(++td1) = j;
        }
    }
    for( int hp = 0; hp <= td1; hp++ ) {
        j = todo(hp);
        if( y(j) == -1 ) {
            Murty_updateAssignments( lab, y, x, j, i0 );
            return td1;
        }
        ok(j) = 1;// true
    }
    int td2 = ( nc - 1 );
    int last = nc;
```

```
    while( true ) {
        if( td1 < 0 ) {
            fail = true; // FAIL!!!
            return 1;
        }
        int j0 = todo(td1--);
        int i = y(j0);
        todo(td2--) = j0;
        int tp = first[i];
        while( kk[tp] != j0 ) {
            tp++;
        }
        double h = cc_[tp] - v(j0) - min_;
        for( int t = first[i]; t < first[i + 1]; t++ ) {
            j = kk[t];
            if( !lap_is_allowed( i, j ) ) { // Fix Murty
                continue;
            }
//            if( !ok(j) ) {
            if( ok(j) == 0 ) { // if( false )
                double vj = cc_[t] - v(j) - h;
//                if( vj < d(j) ) { // POSSIBLE FLOWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
                if( ( d(j) - vj ) > resolution ) { // POSSIBLE FLOWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
                    d(j) = vj;
                    lab(j) = i;
//                    if( vj == min_ ) { // POSSIBLE FLOWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
                    if( std::abs( vj - min_ ) < resolution ) {
                        if( y[j] == -1 ) {
                            Murty_updateDual( nc, d, v, todo, last, min_ );
                            Murty_updateAssignments( lab, y, x, j, i0 );
                            return td1;
                        }
                        todo(++td1) = j;
                        ok(j) = 1; // true
                    }
                }
            }
        }
        if( td1 == -1 ) {
            // The original Pascal code uses finite numbers instead of double.PositiveInfinity
            // so we need to adjust slightly here.
            min_ = infValue;
            last = td2 + 1;
            for( int jp = 0; jp < nc; jp++ ) {
//                if( ( ( d[jp] < min_ ) || ( std::abs( d[jp] - min_ ) < resolution ) ) && !ok(jp) ) {
//                if( ( ( d[jp] < min_ ) || ( std::abs( d[jp] - min_ ) < resolution ) ) && ( ok(jp) == 0 ) ) {
                if( ( std::abs( d[jp] - infValue ) > resolution ) && ( ( ( min_ - d[jp] ) > resolution ) || ( std::abs( d[jp] - min_ ) < resolution ) ) && ( ok(jp) == 0 ) ) {
                    //if( d[jp] < min_ ) {
                    if( ( min_ - d[jp] ) > resolution ) {
                        td1 = -1;
                        min_ = d(jp);
                    }
                    todo(++td1) = jp;
                }
            }
            for( int hp = 0; hp <= td1; hp++ ) {
                j = todo(hp);
                if( y(j) == -1 ) {
                    Murty_updateDual( nc, d, v, todo, last, min_ );
                    Murty_updateAssignments( lab, y, x, j, i0 );
                    return td1;
                }
                ok(j) = 1;//true;
            }
        }
    }
}

//------------------------------------------------------------------------------------------------------------------
int Murty_JVCsparse( const std::vector<double> &cc, const std::vector<int> &kk, const std::vector<int> &first,
    TSearchParam sp, double infValue, double resolution, arma::ivec &rowsol, double &lapcost,
    arma::vec *u_init, arma::vec *v_init, arma::vec *u_out, arma::vec *v_out )
{
    // Объявления
    int nr = first.size() - 1; // Кол-во строк
    int max_kk = -1;
    for( unsigned i = 0; i < kk.size(); i++ ) { // Поиск максимального элемента в массиве kk - это и будет кол-во столбцов
        if( kk[i] > max_kk ) {
            max_kk = kk[i];
        }
    }
    int nc = max_kk + 1;// Кол-во столбцов
```

```
00171     arma::ivec x = arma::ivec( nr, arma::fill::zeros );
00172     arma::ivec y = arma::ivec( nc, arma::fill::zeros );
00173     arma::vec d = arma::vec( nc, arma::fill::zeros );
00174     arma::ivec ok = arma::ivec( nc, arma::fill::zeros ); // bool: 0 - false, 1 - true (classical c-style)
00175     arma::ivec xinv = arma::ivec( nr, arma::fill::zeros ); // bool: 0 - false, 1 - true (classical c-style)
00176     arma::ivec free = arma::ivec( nr, arma::fill::zeros );
00177     arma::ivec todo = arma::ivec( nc, arma::fill::zeros );
00178     arma::ivec lab = arma::ivec( nc, arma::fill::zeros );
00179     int l0 = 0;
00180
00181     // Original u, v
00182     arma::vec u = arma::vec( nr, arma::fill::zeros );
00183     arma::vec v = arma::vec( nc, arma::fill::zeros );
00184     // Murty fix
00185     if( u_init
00186         && v_init
00187         && ( ( u_init->n_elem ) == (size_t)nr )
00188         && ( ( v_init->n_elem ) == (size_t)nc )
00189         )
00190     {
00191         u = *u_init;
00192         v = *v_init;
00193     }
00194
00195     x -= 1;
00196     y -= 1;
00197     free -= 1;
00198     todo -= 1;
00199
00200     // Поиск минимума/максимума
00201     std::vector<double> cc_ = cc;
00202     if( sp == TSearchParam::SP_Max ) {
00203         std::transform( cc_.begin(), cc_.end(), cc_.begin(),
00204             std::bind( std::multiplies<double>(), std::placeholders::_1, -1.0 ) ); // Умножим на -1 для поиска максимума
00205     }
00206
00207     // The initialization steps of LAPJVsp only make sense for square matrices
00208     if( nr == nc ) {
00209         for( int jp = 0; jp < nc; jp++ ) {
00210             v(jp) = infValue;
00211         }
00212         for( int i = 0; i < nr; i++ ) {
00213             for( int t = first[i]; t < first[i + 1]; t++ ) {
00214                 int jp = kk[t];
00215                 if( !lap_is_allowed( i, jp ) ) { // Fix Murty
00216                     continue;
00217                 }
00218 //                  if( cc_[t] < v(jp) ) {
00219                 if( ( v(jp) - cc_[t] ) > resolution ) {
00220                     v(jp) = cc_[t];
00221                     y(jp) = i;
00222                 }
00223             }
00224         }
00225         for( int jp = ( nc - 1 ); jp >= 0; jp-- ) {
00226             int i = y(jp);
00227             if( i < 0 ) {
00228                 continue; // fix Murty
00229             }
00230             if( x(i) == -1 ) {
00231                 x(i) = jp;
00232             } else {
00233                 y(jp) = -1;
00234                 // Here, the original Pascal code simply inverts the sign of x; as that
00235                 // doesn't play too well with zero-indexing, we explicitly keep track of
00236                 // uniqueness instead.
00237                 xinv(i) = 1;
00238             }
00239         }
00240         int lp = 0;
00241         for( int i = 0; i < nr; i++ ) {
00242             if( xinv(i) ) {
00243                 continue;
00244             }
00245             if( x(i) != -1 ) {
00246                 double min_ = infValue;
00247                 int j1 = x(i);
00248                 for( int t = first[i]; t < first[i + 1]; t++ ) {
00249                     int jp = kk[t];
00250                     if( !lap_is_allowed( i, jp ) ) { // Fix Murty
00251                         continue;
00252                     }
00253                     if( jp != j1 ) {
00254 //                      if( ( cc_[t] - v(jp) ) < min_ ) {
00255                         if( ( min_ - ( cc_[t] - v(jp) ) ) > resolution ) {
00256                             min_ = ( cc_[t] - v(jp) );
00257                         }
```

```
                }
            }
            u(i) = min_;
            int tp = first[i];
            while( kk[tp] != j1 ) {
                tp++;
            }
            v(j1) = cc_[tp] - min_;
        } else {
            free(lp++) = i;
        }
    }
    for( int tel = 0; tel < 2; tel++ ) {
        int h = 0;
        int l0p = lp;
        lp = 0;
        while( h < l0p ) {
            // Note: In the original Pascal code, the indices of the lowest
            // and second-lowest reduced costs are never reset. This can
            // cause issues for infeasible problems; see https://stackoverflow.com/q/62875232/5085211
            int i = free(h++);

            //-----------------------------------------------------------------------------------------------
            // ORIGINAL SEARCH OF MIN AND SUBMIN
            //-----------------------------------------------------------------------------------------------

            int j0p = -1; // Index of minimum
            int j1p = -1; // Index of subminimum
            double v0 = infValue; // Value of minimum
            double vj = infValue; // Value of subminimum
            for( int t = first[i]; t < first[i + 1]; t++ ) {
                int jp = kk[t];
                if( !lap_is_allowed( i, jp ) ) { // Fix Murty
                    continue;
                }
                double dj = cc_[t] - v(jp);
//                if( dj < vj ) {
                if( ( vj - dj ) > resolution ) {
//                    if( dj >= v0 ) { // POSSIBLE FLOW!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!1
//                    if( ( dj > v0 ) || ( std::abs( dj - v0 ) < resolution ) ) {
                    if( ( ( dj - v0 ) > resolution ) || ( std::abs( dj - v0 ) < resolution ) ) {
                        vj = dj;
                        j1p = jp;
                    } else {
                        vj = v0;
                        v0 = dj;
                        j1p = j0p;
                        j0p = jp;
                    }
                }
            }
            // If the index of the column with the largest reduced cost has not been
            // set, no assignment is possible for this row.
            if( j0p < 0 ) {
                return 1; // No feasible solution!!!
            }
            int i0 = y(j0p);

            //-----------------------------------------------------------------------------------------------
            // MY SEARCH OF MIN AND SUBMIN
            //-----------------------------------------------------------------------------------------------
//            // find minimum and second minimum reduced cost over columns.
//            int j0p = -1; // Index of minimum
//            int j1p = -1; // Index of subminimum
//            double v0 = infValue; // Value of minimum
//            double vj = infValue; // Value of subminimum
//            arma::vec i_th_row( nc, arma::fill::zeros );
//            i_th_row.fill( infValue );
//            for( int t = first[i]; t < first[i + 1]; t++ ) {
//                int jp = kk[t];
//                double dj = cc_[t] - v(jp);
//                i_th_row(jp) = dj;
//            }
//            j0p = arma::index_min( i_th_row );  // Index of minimum
//            v0 = i_th_row(j0p);                 // Value of minimum
//            i_th_row(j0p) = infValue;

//            j1p = arma::index_min( i_th_row );  // Index of subminimum
//            vj = i_th_row(j1p);                 // Value of subminimum

//            int i0 = y(j0p);
            //-----------------------------------------------------------------------------------------------
            // ORIGINAL
            //-----------------------------------------------------------------------------------------------

            u(i) = vj;
//            if( v0 < vj ) { // FIX
```

```
00345                 if( ( vj - v0 ) > resolution ) { // MY
00346                     v(j0p) += ( v0 - vj );
00347                 } else if( i0 != -1 ) {
00348                     j0p = j1p;
00349                     i0 = y(j0p);
00350                 }
00351                 x(i) = j0p;
00352                 y(j0p) = i;
00353                 if( i0 != -1 ) {
00354 //                  if( v0 < vj ) { // FIX
00355                     if( ( vj - v0 ) > resolution ) { // MY
00356                         free(--h) = i0;
00357                     } else {
00358                         free(lp++) = i0;
00359                     }
00360                 }
00361 
00362                 //---------------------------------------------------------------------------------------------
00363                 // MY SEARCH OF MIN AND SUBMIN
00364                 //---------------------------------------------------------------------------------------------
00365 //              u(i) = vj;
00366 //              if( ( vj - v0 ) > resolution ) { // if( v0 < vj )
00367 //                  // change the reduction of the minimum column to increase the minimum
00368 //                  // reduced cost in the row to the subminimum.
00369 //                  v(j0p) += ( v0 - vj );
00370 //              } else {
00371 //                  if( i0 > -1 ) { // minimum and subminimum equal.
00372 //                      // minimum column j1 is assigned.
00373 //                      // swap columns j1 and j2, as j2 may be unassigned.
00374 //                      j0p = j1p;
00375 //                      i0 = y(j0p);
00376 //                  }
00377 //              }
00378 //              // (re-)assign i to j1, possibly de-assigning an i0.
00379 //              x(i) = j0p;
00380 //              y(j0p) = i;
00381 //              if( i0 > -1 ) {
00382 //                  if( ( vj - v0 ) > resolution ) { // FIX
00383 //                      free(--h) = i0;
00384 //                  } else {
00385 //                      free(lp++) = i0;
00386 //                  }
00387 //              }
00388                 //---------------------------------------------------------------------------------------------
00389             }
00390         } // end for( int tel = 0; tel < 2; tel++ )
00391         l0 = lp;
00392     } else { // end if( nr == nc )
00393         l0 = nr;
00394         for( int i = 0; i < nr; i++ ) {
00395             free(i) = i;
00396         }
00397     }
00398     int td1 = -1;
00399     for( int l = 0; l < l0; l++ ) {
00400         bool fail = false;
00401         td1 = Murty_solveForOneL( cc_, kk, first, l, nc, d, ok, free, v, lab, todo, y, x, td1, resolution, infValue, fail );
00402         if( fail ) {
00403             return 1;
00404         }
00405     }
00406     // Prapare output - rowsol and lapcost.
00407     lapcost = 0.0;
00408     for( int i = 0; i < nr; i++ ) { // i - row index
00409         rowsol[i] = x[i];
00410         const int j_ = rowsol[i]; // j - col index
00411         const int start = first[i];
00412         const int end = first[i+1];
00413         for( int j = start; j < end; j++ ) {
00414             if( j_ == kk[j] ) {
00415                 lapcost += cc[j];
00416                 break;
00417             }
00418         }
00419     }
00420     if( u_out ) {
00421         *u_out = u;
00422     }
00423     if( v_out ) {
00424         *v_out = v;
00425     }
00426     return 0;
00427 }
00428 
00429 //---------------------------------------------------------------------------------------------
00430 int Murty_JVCsparse( const Sparse::CMatrixCSR &csr, TSearchParam sp, double infValue, double resolution, arma::ivec
        &rowsol,
```

```
00431     double &lapcost, arma::vec *u_init, arma::vec *v_init, arma::vec *u_out, arma::vec *v_out )
00432 {
00433     int result = Murty_JVCsparse( csr.csr_val, csr.csr_kk, csr.csr_first, sp, infValue, resolution, rowsol, lapcost,
00434         u_init, v_init, u_out, v_out );
00435     return result;
00436 }
00437
00438 }
00439 }
```

8.31 Файл murty_jvc_sparse.hpp

Решение задачи о назначениях методом JVC для разреженных матриц

```
#include <armadillo>
#include <compare.hpp>
#include <searchparam.hpp>
#include <sparse.hpp>
```

Пространства имен

- namespace SPML

  Специальная библиотека программных модулей (СБ ПМ)
- namespace SPML::LAP

  Решение задачи о назначениях

Функции

- int SPML::LAP::Murty_JVCsparse (const std::vector< double > &cc, const std::vector< int > &kk, const std::vector< int > &first, TSearchParam sp, double infValue, double resolution, arma::ivec &rowsol, double &lapcost, arma::vec *u_init=nullptr, arma::vec *v_init=nullptr, arma::vec *u_out=nullptr, arma::vec *v_out=nullptr)

  Метод Джонкера-Волгенанта-Кастаньона (Jonker-Volgenant-Castanon) решения задачи о назначениях для разреженных матриц
- int SPML::LAP::Murty_JVCsparse (const Sparse::CMatrixCSR &csr, TSearchParam sp, double infValue, double resolution, arma::ivec &rowsol, double &lapcost, arma::vec *u_init=nullptr, arma::vec *v_init=nullptr, arma::vec *u_out=nullptr, arma::vec *v_out=nullptr)

  Метод Джонкера-Волгенанта-Кастаньона (Jonker-Volgenant-Castanon) решения задачи о назначениях для разреженных матриц

8.31.1 Подробное описание

Решение задачи о назначениях методом JVC для разреженных матриц

Адаптирован под метод Murty наличием "тёплого" старта u и v

Дата

21.01.26 - создан

Автор

Соболев А.А.

См. определение в файле murty_jvc_sparse.hpp

8.32 murty_jvc_sparse.hpp

См. документацию.

```
00001 //-----------------------------------------------------------------------------------------------------------------
00011
00012 #ifndef SPML_MURTY_JVC_SPARSE_HPP_
00013 #define SPML_MURTY_JVC_SPARSE_HPP_
00014
00015 #include <armadillo>
00016
00017 #include <compare.hpp>
00018 #include <searchparam.hpp>
00019 #include <sparse.hpp>
00020
00021 namespace SPML
00022 {
00023 namespace LAP
00024 {
00043 int Murty_JVCsparse( const std::vector<double> &cc, const std::vector<int> &kk, const std::vector<int> &first,
00044     TSearchParam sp, double infValue, double resolution, arma::ivec &rowsol, double &lapcost,
00045     arma::vec *u_init = nullptr, arma::vec *v_init = nullptr, arma::vec *u_out = nullptr, arma::vec *v_out = nullptr );
00046
00063 int Murty_JVCsparse( const Sparse::CMatrixCSR &csr, TSearchParam sp, double infValue, double resolution, arma::ivec
      &rowsol,
00064     double &lapcost,
00065     arma::vec *u_init = nullptr, arma::vec *v_init = nullptr, arma::vec *u_out = nullptr, arma::vec *v_out = nullptr );

00066
00067
00068 } // namespace LAP
00069 } // namespace SPML
00070 #endif
```

8.33 Файл searchparam.hpp

Параметр поиска max/min.

Пространства имен

- namespace SPML

  Специальная библиотека программных модулей (СБ ПМ)
- namespace SPML::LAP

  Решение задачи о назначениях

Перечисления

- enum SPML::LAP::TSearchParam { SPML::LAP::SP_Min , SPML::LAP::SP_Max }

  Критерий поиска - минимум/максимум для задачи о назначениях

8.33.1 Подробное описание

Параметр поиска max/min.

Вынесено отдельно от реализации методов решения задачи о назначениях

Дата

07.02.23 - создан

Автор

Соболев А.А.

См. определение в файле searchparam.hpp

8.34 searchparam.hpp

См. документацию.

```
00001 //-----------------------------------------------------------------------------------------------------------
00011
00012 #ifndef SPML_SEARCHPARAM_HPP_
00013 #define SPML_SEARCHPARAM_HPP_
00014
00015 namespace SPML
00016 {
00017 namespace LAP
00018 {
00022 enum TSearchParam
00023 {
00024     SP_Min,
00025     SP_Max
00026 };
00027
00028 } // namespace LAP
00029 } // namespace SPML
00030 #endif
```

8.35 Файл seqextr.cpp

Последовательный выбор экстремума

#include <seqextr.hpp>

Пространства имен

- namespace SPML

  Специальная библиотека программных модулей (СБ ПМ)
- namespace SPML::LAP

  Решение задачи о назначениях

Функции

- void SPML::LAP::SequentalExtremum (const arma::mat &assigncost, TSearchParam sp, double infValue, double resolution, arma::ivec &rowsol, double &lapcost)

  Процедура последовательного поиска экстремума по строкам/столбцам матрицы ценностей
- void SPML::LAP::SequentalExtremum (const Sparse::CMatrixCOO &assigncost, TSearchParam sp, double infValue, double resolution, arma::ivec &rowsol, double &lapcost)

  Процедура последовательного поиска экстремума по строкам/столбцам матрицы ценностей

8.35.1 Подробное описание

Последовательный выбор экстремума

Дата

24.06.22 - создан

Автор

Соболев А.А.

См. определение в файле seqextr.cpp

8.36 seqextr.cpp

См. документацию.

```
00001 //-----------------------------------------------------------------------------------------------------------------
00010
00011 #include <seqextr.hpp>
00012
00013 namespace SPML
00014 {
00015 namespace LAP
00016 {
00017
00018 void SequentalExtremum( const arma::mat &assigncost, TSearchParam sp, double infValue, double resolution,
00019     arma::ivec &rowsol, double &lapcost )
00020 {
00021     size_t cols = assigncost.n_cols;
00022     size_t rows = assigncost.n_rows;
00023     arma::mat cost( rows, cols, arma::fill::zeros );
00024     if( sp == TSearchParam::SP_Max ) { // Поиск минимума/максимума
00025         cost = -assigncost;
00026     } else {
00027         cost = assigncost;
00028     }
00029
00030     rowsol.zeros();
00031     rowsol -= 1;
00032
00033     // Процедура всегда ищет минимум!
00034
00035     // Вектора выкинутых индексов
00036     std::vector<int> cols_pulled;
00037     std::vector<int> rows_pulled;
00038     cols_pulled.reserve( cols );
00039     rows_pulled.reserve( rows );
00040
00041     while( rows_pulled.size() < rows ) {
00042
00043         double min_val = infValue;
00044         int min_row = INT32_MAX;
00045         int min_col = INT32_MAX;
00046         bool min_found = false;
00047
00048         for( size_t row = 0; row < rows; row++ ) {
00049             if( std::find( rows_pulled.begin(), rows_pulled.end(), row ) != rows_pulled.end() ) {
00050                 continue; // индекс row был выкинут
00051             }
00052             for( size_t col = 0; col < cols; col++ ) {
00053                 if( std::find( cols_pulled.begin(), cols_pulled.end(), col ) != cols_pulled.end() ) {
00054                     continue; // индекс col был выкинут
00055                 }
00056                 if( ( min_val - cost(row,col) ) > resolution ) { // if( cost(i,j) < min_val ) {
00057                     min_val = cost(row,col);
00058                     min_row = row;
00059                     min_col = col;
00060                     min_found = true;
00061                 }
00062             }
00063         }
00064         if( min_found ) {
00065             rows_pulled.push_back( min_row );
00066             cols_pulled.push_back( min_col );
00067
00068             rowsol( min_row ) = min_col; // Решение!
00069         }
00070     }
00071
00072     // calculate lapcost.
00073     lapcost = 0.0;
00074     for( size_t i = 0; i < rows; i++ ) {
00075         int j = rowsol(i);
00076         double element_i_j = assigncost(i,j);
00077         if( !SPML::Compare::AreEqualAbs( element_i_j, infValue, resolution ) ) {
00078             lapcost += element_i_j;
00079         }
00080     }
00081     return;
00082 }
00083
00084 void SequentalExtremum( const Sparse::CMatrixCOO &assigncost, TSearchParam sp, double infValue, double resolution,
00085     arma::ivec &rowsol, double &lapcost )
00086 {
00087     rowsol.zeros();
00088     rowsol -= 1;
00089
00090     // Нахождение числа строк/столбцов через std::set
```

```
00091     const std::size_t n_elems = assigncost.coo_val.size();
00092     std::set<int> pulled; // чтобы не морочиться на уникальность вхождений
00093
00094     std::vector<double> cost = assigncost.coo_val;
00095     if( sp == TSearchParam::SP_Max ) {
00096         std::transform( cost.begin(), cost.end(), cost.begin(),
00097             std::bind( std::multiplies<double>(), std::placeholders::_1, -1.0 ) ); // Умножим на -1 для поиска максимума
00098     }
00099
00100     // Процедура всегда ищет минимум!
00101     while( pulled.size() < n_elems ) {
00102
00103         double min_val = infValue;
00104         int min_row = INT32_MAX;
00105         int min_col = INT32_MAX;
00106         bool min_found = false;
00107
00108         for( size_t n = 0; n < n_elems; n++ ) {
00109             if( std::find( pulled.begin(), pulled.end(), n ) != pulled.end() ) {
00110                 continue; // индексы выкинуты
00111             }
00112             if( ( min_val - cost[n] ) > resolution ) { // if( cost(n) < min_val ) {
00113                 min_val = cost[n];
00114                 min_row = assigncost.coo_row[n];
00115                 min_col = assigncost.coo_col[n];
00116                 min_found = true;
00117             }
00118         }
00119         if( min_found ) {
00120             for( size_t n = 0; n < n_elems; n++ ) {
00121                 if( ( assigncost.coo_row[n] == min_row ) ||
00122                     ( assigncost.coo_col[n] == min_col ) )
00123                 {
00124                     pulled.insert( n );
00125                 }
00126             }
00127             rowsol( min_row ) = min_col; // Решение!
00128         }
00129     }
00130     // calculate lapcost.
00131
00132
00133     // Нахождение числа строк/столбцов через std::set
00134 //    size_t cols = std::set<int>( ( assigncost.coo_col ).begin(), ( assigncost.coo_col ).end() ).size();
00135 //    size_t rows = std::set<int>( ( assigncost.coo_row ).begin(), ( assigncost.coo_row ).end() ).size();
00136     lapcost = 0.0;
00137     for( size_t i = 0; i < rowsol.n_elem; i++ ) {
00138         if( i < 0 ) {
00139             continue;
00140         }
00141         int j = rowsol(i);
00142         if( j < 0 ) {
00143             continue;
00144         }
00145         for( size_t n = 0; n < n_elems; n++ ) {
00146             if( ( assigncost.coo_row[n] == i ) &&
00147                 ( assigncost.coo_col[n] == j ) )
00148             {
00149                 if( !SPML::Compare::AreEqualAbs( assigncost.coo_val[n], infValue, resolution ) ) {
00150                     lapcost += assigncost.coo_val[n];
00151 //                    break;
00152                 }
00153             }
00154         }
00155     }
00156     return;
00157 }
00158
00159 } // end namespace LAP
00160 } // end namespace SPML
```

8.37 Файл seqextr.hpp

Последовательный выбор экстремума

```
#include <armadillo>
#include <set>
#include <compare.hpp>
#include <sparse.hpp>
#include <searchparam.hpp>
```

Пространства имен

- namespace SPML

  Специальная библиотека программных модулей (СБ ПМ)
- namespace SPML::LAP

  Решение задачи о назначениях

Функции

- void SPML::LAP::SequentalExtremum (const arma::mat &assigncost, TSearchParam sp, double infValue, double resolution, arma::ivec &rowsol, double &lapcost)

  Процедура последовательного поиска экстремума по строкам/столбцам матрицы ценностей
- void SPML::LAP::SequentalExtremum (const Sparse::CMatrixCOO &assigncost, TSearchParam sp, double infValue, double resolution, arma::ivec &rowsol, double &lapcost)

  Процедура последовательного поиска экстремума по строкам/столбцам матрицы ценностей

8.37.1 Подробное описание

Последовательный выбор экстремума

Дата

07.02.23 - создан

Автор

Соболев А.А.

См. определение в файле seqextr.hpp

8.38 seqextr.hpp

См. документацию.

```
00001 //-------------------------------------------------------------------------------------------------------------
00010
00011 #ifndef SPML_SEQEXTR_HPP_
00012 #define SPML_SEQEXTR_HPP_
00013
00014 #include <armadillo>
00015 #include <set>
00016
00017 #include <compare.hpp>
00018 #include <sparse.hpp>
00019 #include <searchparam.hpp>
00020
00021 namespace SPML
00022 {
00023 namespace LAP
00024 {
00037 void SequentalExtremum( const arma::mat &assigncost, TSearchParam sp, double infValue, double resolution,
00038     arma::ivec &rowsol, double &lapcost );
00039
00052 void SequentalExtremum( const Sparse::CMatrixCOO &assigncost, TSearchParam sp, double infValue, double resolution,
00053     arma::ivec &rowsol, double &lapcost );
00054
00055 } // namespace LAP
00056 } // namespace SPML
00057 #endif
```

8.39 Файл sparse.cpp

Работа с разреженными матрицами

#include <sparse.hpp>

Пространства имен

- namespace SPML

  Специальная библиотека программных модулей (СБ ПМ)
- namespace SPML::Sparse

  Решение задачи о назначениях

Функции

- void SPML::Sparse::MatrixDenseToCOO (const arma::mat &A, std::vector< double > &coo_val, std::vector< int > &coo_row, std::vector< int > &coo_col)

  Преобразование плотной матрицы в COO формат (Coordinated list)
- void SPML::Sparse::MatrixDenseToCOO (const arma::mat &A, CMatrixCOO &COO)

  Преобразование плотной матрицы в COO формат (Coordinated list)
- void SPML::Sparse::MatrixCOOtoDense (const std::vector< double > &coo_val, const std::vector< int > &coo_row, const std::vector< int > &coo_col, arma::mat &A)

  Преобразование матрицы из COO формата (Coordinated list) в плотную матрицу
- void SPML::Sparse::MatrixCOOtoDense (const CMatrixCOO &COO, arma::mat &A)

  Преобразование матрицы из COO формата (Coordinated list) в плотную матрицу
- void SPML::Sparse::MatrixDenseToCSR (const arma::mat &A, std::vector< double > &csr_val, std::vector< int > &csr_kk, std::vector< int > &csr_first)

  Преобразование плотной матрицы в CSR формат (Compressed Sparse Row Yale format)
- void SPML::Sparse::MatrixDenseToCSR (const arma::mat &A, CMatrixCSR &CSR)

  Преобразование плотной матрицы в CSR формат (Compressed Sparse Row Yale format)
- void SPML::Sparse::MatrixCSRtoDense (const std::vector< double > &csr_val, const std::vector< int > &csr_kk, const std::vector< int > &csr_first, arma::mat &A)

  Преобразование матрицы из CSR формата (Compressed Sparse Row Yale format) в плотную матрицу
- void SPML::Sparse::MatrixCSRtoDense (const CMatrixCSR &CSR, arma::mat &A)

  Преобразование матрицы из CSR формата (Compressed Sparse Row Yale format) в плотную матрицу
- void SPML::Sparse::MatrixDenseToCSC (const arma::mat &A, std::vector< double > &csc_val, std::vector< int > &csc_kk, std::vector< int > &csc_first)

  Преобразование плотной матрицы в CSC формат (Compressed Sparse Column Yale format)
- void SPML::Sparse::MatrixDenseToCSC (const arma::mat &A, CMatrixCSC &CSC)

  Преобразование плотной матрицы в CSC формат (Compressed Sparse Column Yale format)
- void SPML::Sparse::MatrixCSCtoDense (const std::vector< double > &csc_val, const std::vector< int > &csc_kk, const std::vector< int > &csc_first, arma::mat &A)

  Преобразование матрицы из CSC формата (Compressed Sparse Column Yale format) в плотную матрицу
- void SPML::Sparse::MatrixCSCtoDense (const CMatrixCSC &CSC, arma::mat &A)

  Преобразование матрицы из CSC формата (Compressed Sparse Column Yale format) в плотную матрицу
- void SPML::Sparse::MatrixCOOtoCSR (const std::vector< double > &coo_val, const std::vector< int > &coo_row, const std::vector< int > &coo_col, std::vector< double > &csr_val, std::vector< int > &csr_kk, std::vector< int > &csr_first, bool sorted=false)

Преобразование матрицы в COO формате (Coordinate list) в CSR формат (Compressed Sparse Row Yale format)

- void SPML::Sparse::MatrixCOOtoCSR (const CMatrixCOO &COO, CMatrixCSR &CSR, bool sorted=false)

  Преобразование матрицы в COO формате (Coordinate list) в CSR формат (Compressed Sparse Row Yale format)
- void SPML::Sparse::MatrixCOOtoCSC (const std::vector< double > &coo_val, const std::vector< int > &coo_row, const std::vector< int > &coo_col, std::vector< double > &csc_val, std::vector< int > &csc_kk, std::vector< int > &csc_first, bool sorted=false)

  Преобразование матрицы в COO формате (Coordinate list) в CSC формат (Compressed Sparse Column Yale format)
- void SPML::Sparse::MatrixCOOtoCSC (const CMatrixCOO &COO, CMatrixCSC &CSC, bool sorted=false)

  Преобразование матрицы в COO формате (Coordinate list) в CSC формат (Compressed Sparse Column Yale format)

8.39.1 Подробное описание

Работа с разреженными матрицами

Перевод матрицы из плотного представления в COO, CSR, CSC вид

Дата

27.05.22 - создан

Автор

Соболев А.А.

См. определение в файле sparse.cpp

8.40 sparse.cpp

См. документацию.

```
00001 //-----------------------------------------------------------------------------------------------------------
00011
00012 #include <sparse.hpp>
00013
00014 namespace SPML
00015 {
00016 namespace Sparse
00017 {
00018 //-----------------------------------------------------------------------------------------------------------
00019
00020 void MatrixDenseToCOO( const arma::mat &A, std::vector<double> &coo_val, std::vector<int> &coo_row,
00021     std::vector<int> &coo_col )
00022 {
00023     coo_val.clear();
00024     coo_row.clear();
00025     coo_col.clear();
00026
00027     int nnz = arma::accu( A != 0 ); // Число ненулевых элементов
00028     coo_val.reserve( nnz );
00029     coo_row.reserve( nnz );
00030     coo_col.reserve( nnz );
00031
00032     for( unsigned long long i = 0; i < A.n_rows; i++ ) { // Строки
00033         for( unsigned long long j = 0; j < A.n_cols; j++ ) { // Столбцы
00034             if( !Compare::IsZeroAbs( A(i,j) ) ) {
```

```
00035                 coo_val.push_back( A(i, j) );
00036                 coo_row.push_back( i );
00037                 coo_col.push_back( j );
00038             }
00039         }
00040     }
00041 }
00042
00043 void MatrixDenseToCOO( const arma::mat &A, CMatrixCOO &COO )
00044 {
00045     MatrixDenseToCOO( A, COO.coo_val, COO.coo_row, COO.coo_col );
00046 }
00047
00048 void MatrixCOOtoDense( const std::vector<double> &coo_val, const std::vector<int> &coo_row,
00049     const std::vector<int> &coo_col, arma::mat &A )
00050 {
00051     int n = *( std::max_element( coo_row.begin(), coo_row.end() ) ) + 1;
00052     int m = *( std::max_element( coo_col.begin(), coo_col.end() ) ) + 1;
00053     A = arma::mat( n, m, arma::fill::zeros );
00054
00055     for( unsigned k = 0; k < coo_val.size(); k++ ) {
00056         A( coo_row[k], coo_col[k] ) = coo_val[k];
00057     }
00058 }
00059
00060 void MatrixCOOtoDense( const CMatrixCOO &COO, arma::mat &A )
00061 {
00062     MatrixCOOtoDense( COO.coo_val, COO.coo_row, COO.coo_col, A );
00063 }
00064
00065 //-----------------------------------------------------------------------------------------------------------------
00066
00067 void MatrixDenseToCSR( const arma::mat &A, std::vector<double> &csr_val, std::vector<int> &csr_kk,
00068     std::vector<int> &csr_first )
00069 {
00070     csr_val.clear();
00071     csr_kk.clear();
00072     csr_first.clear();
00073
00074     int nnz = arma::accu( A != 0 ); // Число ненулевых элементов
00075     csr_val.reserve( nnz );
00076     csr_kk.reserve( nnz );
00077     csr_first.reserve( A.n_rows + 1 );
00078
00079     int nnz_in_row = 0; // Кол-во ненулевых элементов (non-zero) в строке
00080     csr_first.push_back(0);// Первый элемент надо занулить
00081
00082     for( unsigned long long i = 0; i < A.n_rows; i++ ) { // Строки
00083         nnz_in_row = 0;
00084         for( unsigned long long j = 0; j < A.n_cols; j++ ) { // Столбцы
00085             if( !Compare::IsZeroAbs( A(i,j) ) ) { // if( A[i,j] != 0 )
00086                 csr_val.push_back( A(i,j) );//CSR[nnz] = A(i,j);
00087                 csr_kk.push_back( j );
00088                 nnz_in_row++;
00089             }
00090         }
00091         csr_first.push_back( csr_first[i] + nnz_in_row );
00092     }
00093 }
00094
00095 void MatrixDenseToCSR( const arma::mat &A, CMatrixCSR &CSR )
00096 {
00097     MatrixDenseToCSR( A, CSR.csr_val, CSR.csr_kk, CSR.csr_first );
00098 }
00099
00100 void MatrixCSRtoDense( const std::vector<double> &csr_val, const std::vector<int> &csr_kk,
00101     const std::vector<int> &csr_first, arma::mat &A )
00102 {
00103     int n = csr_first.size() - 1;
00104     int m = *( std::max_element( csr_kk.begin(), csr_kk.end() ) ) + 1;
00105     A = arma::mat( n, m, arma::fill::zeros );
00106
00107     for( int i = 0; i < n; i++ ) {
00108         int nnz_row = csr_first[i];
00109         int nnz_row_next = csr_first[i+1];
00110         for( int j = nnz_row; j < nnz_row_next; j++ ) {
00111             A(i, csr_kk[j]) = csr_val[j];//C[m*i+kk[j]] = CSR[j];
00112         }
00113     }
00114 }
00115
00116 void MatrixCSRtoDense( const CMatrixCSR &CSR, arma::mat &A )
00117 {
00118     MatrixCSRtoDense( CSR.csr_val, CSR.csr_kk, CSR.csr_first, A );
00119 }
00120
00121 //-----------------------------------------------------------------------------------------------------------------
```

```
00122
00123 void MatrixDenseToCSC( const arma::mat &A, std::vector<double> &csc_val, std::vector<int> &csc_kk,
00124     std::vector<int> &csc_first )
00125 {
00126     csc_val.clear();
00127     csc_kk.clear();
00128     csc_first.clear();
00129
00130     int nnz = arma::accu( A != 0 ); // Число ненулевых элементов
00131     csc_val.reserve( nnz );
00132     csc_kk.reserve( nnz );
00133     csc_first.reserve( A.n_cols + 1 );
00134
00135     int nnz_in_col = 0; // Кол-во ненулевых элементов (non-zero) в столбце
00136     csc_first.push_back(0);// Первый элемент надо занулить
00137
00138     for( unsigned long long j = 0; j < A.n_cols; j++ ) { // Столбцы
00139         nnz_in_col = 0;
00140         for( unsigned long long i = 0; i < A.n_rows; i++ ) { // Строки
00141             if( !Compare::IsZeroAbs( A(i,j) ) ) { // if( A[i,j] != 0 )
00142                 csc_val.push_back( A(i,j) );//CSC[nnz] = A(i,j);
00143                 csc_kk.push_back( i );
00144                 nnz_in_col++;
00145             }
00146         }
00147         csc_first.push_back( csc_first[j] + nnz_in_col );
00148     }
00149 }
00150
00151 void MatrixDenseToCSC( const arma::mat &A, CMatrixCSC &CSC )
00152 {
00153     MatrixDenseToCSC( A, CSC.csc_val, CSC.csc_kk, CSC.csc_first );
00154 }
00155
00156 void MatrixCSCtoDense( const std::vector<double> &csc_val, const std::vector<int> &csc_kk,
00157     const std::vector<int> &csc_first, arma::mat &A )
00158 {
00159     int n = *( std::max_element( csc_kk.begin(), csc_kk.end() ) ) + 1;
00160     int m = csc_first.size() - 1;
00161     A = arma::mat( n, m, arma::fill::zeros );
00162
00163     for( int j = 0; j < m; j++ ) {
00164         int nnz_col = csc_first[j];
00165         int nnz_col_next = csc_first[j+1];
00166         for( int i = nnz_col; i < nnz_col_next; i++ ) {
00167             A(csc_kk[i], j) = csc_val[i];
00168         }
00169     }
00170 }
00171
00172 void MatrixCSCtoDense( const CMatrixCSC &CSC, arma::mat &A )
00173 {
00174     MatrixCSCtoDense( CSC.csc_val, CSC.csc_kk, CSC.csc_first, A );
00175 }
00176
00177 //-----------------------------------------------------------------------------------------------------------------
00178
00179 void MatrixCOOtoCSR( const std::vector<double> &coo_val, const std::vector<int> &coo_row,
00180     const std::vector<int> &coo_col, std::vector<double> &csr_val, std::vector<int> &csr_kk,
00181     std::vector<int> &csr_first, bool sorted )
00182 {
00183     csr_val.clear();
00184     csr_kk.clear();
00185     csr_first.clear();
00186     int nnz = coo_val.size(); // Число ненулевых элементов
00187     int n = *( std::max_element( coo_row.begin(), coo_row.end() ) ) + 1; // n_rows
00188
00189     if( sorted ) { // https://stackoverflow.com/questions/23583975/convert-coo-to-csr-format-in-c
00190         csr_val = coo_val;
00191         csr_kk = coo_col;
00192         for( int i = 0; i <= n; i++ ) {
00193             csr_first.push_back( 0 );
00194         }
00195         for( int i = 0; i < nnz; i++ ) {
00196             csr_first[coo_row[i] + 1]++;
00197         }
00198         for( int i = 0; i < n; i++ ) {
00199             csr_first[i + 1] += csr_first[i];
00200         }
00201     } else { // https://github.com/scipy/scipy/blob/3b36a57/scipy/sparse/sparsetools/coo.h#L34
00202         for( int i = 0; i < nnz; i++ ) {
00203             csr_val.push_back( 0 );
00204             csr_kk.push_back( 0 );
00205         }
00206         for( int i = 0; i <= n; i++ ) {
00207             csr_first.push_back( 0 );
00208         }
```

```
00209
00210         // Bp - kk
00211         // Bi - first
00212         // Bx - CSR
00213
00214         // Ax - COO
00215         // Ai - row
00216         // Aj - col
00217
00218         //compute number of non-zero entries per row of A
00219         for( int k = 0; k < nnz; k++ ) {
00220             csr_first[coo_row[k]]++;
00221         }
00222
00223         //cumsum the nnz per row to get Bp[]
00224         for( int i = 0, cumsum = 0; i < n; i++ ) {
00225             int temp = csr_first[i];
00226             csr_first[i] = cumsum;
00227             cumsum += temp;
00228         }
00229         csr_first[n] = nnz;
00230
00231         //write Aj,Ax into Bj,Bx
00232         for( int k = 0; k < nnz; k++ ) {
00233             int row_ = coo_row[k];
00234             int dest = csr_first[row_];
00235
00236             csr_kk[dest] = coo_col[k];
00237             csr_val[dest] = coo_val[k];
00238
00239             csr_first[row_]++;
00240         }
00241
00242         for( int i = 0, last = 0; i <= n; i++ ) {
00243             int temp = csr_first[i];
00244             csr_first[i] = last;
00245             last = temp;
00246         }
00247         //now Bp,Bj,Bx form a CSR representation (with possible duplicates)
00248     }
00249 }
00250
00251 void MatrixCOOtoCSR( const CMatrixCOO &COO, CMatrixCSR &CSR, bool sorted )
00252 {
00253     MatrixCOOtoCSR( COO.coo_val, COO.coo_row, COO.coo_col, CSR.csr_val, CSR.csr_kk, CSR.csr_first, sorted );
00254 }
00255
00256 void MatrixCOOtoCSC( const std::vector<double> &coo_val, const std::vector<int> &coo_row,
00257     const std::vector<int> &coo_col, std::vector<double> &csc_val, std::vector<int> &csc_kk,
00258     std::vector<int> &csc_first, bool sorted )
00259 {
00260     csc_val.clear();
00261     csc_kk.clear();
00262     csc_first.clear();
00263     int nnz = coo_val.size(); // Число ненулевых элементов
00264     int m = *( std::max_element( coo_col.begin(), coo_col.end() ) ) + 1; // m_cols
00265
00266     if( sorted ) { // https://stackoverflow.com/questions/23583975/convert-coo-to-csr-format-in-c
00267         csc_val = coo_val;
00268         csc_kk = coo_row;
00269         for( int i = 0; i <= m; i++ ) {
00270             csc_first.push_back( 0 );
00271         }
00272         for( int i = 0; i < nnz; i++ ) {
00273             csc_first[coo_col[i] + 1]++;
00274         }
00275         for( int i = 0; i < m; i++ ) {
00276             csc_first[i + 1] += csc_first[i];
00277         }
00278     } else { // https://github.com/scipy/scipy/blob/3b36a57/scipy/sparse/sparsetools/coo.h#L34
00279         for( int i = 0; i < nnz; i++ ) {
00280             csc_val.push_back( 0 );
00281             csc_kk.push_back( 0 );
00282         }
00283         for( int i = 0; i <= m; i++ ) {
00284             csc_first.push_back( 0 );
00285         }
00286
00287         // Bp - kk
00288         // Bi - first
00289         // Bx - CSR
00290
00291         // Ax - COO
00292         // Ai - row
00293         // Aj - col
00294
00295         //compute number of non-zero entries per col of A
```

```
00296         for( int k = 0; k < nnz; k++ ) {
00297             csc_first[coo_col[k]]++;
00298         }
00299 
00300         //cumsum the nnz per col to get Bp[]
00301         for( int j = 0, cumsum = 0; j < m; j++ ) {
00302             int temp = csc_first[j];
00303             csc_first[j] = cumsum;
00304             cumsum += temp;
00305         }
00306         csc_first[m] = nnz;
00307 
00308         //write Aj,Ax into Bj,Bx
00309         for(int k = 0; k < nnz; k++) {
00310             int col_ = coo_col[k];
00311             int dest = csc_first[col_];
00312 
00313             csc_kk[dest] = coo_row[k];
00314             csc_val[dest] = coo_val[k];
00315 
00316             csc_first[col_]++;
00317         }
00318 
00319         for( int j = 0, last = 0; j <= m; j++ ) {
00320             int temp = csc_first[j];
00321             csc_first[j] = last;
00322             last = temp;
00323         }
00324         //now Bp,Bj,Bx form a CSC representation (with possible duplicates)
00325     }
00326 }
00327 
00328 void MatrixCOOtoCSC( const CMatrixCOO &COO, CMatrixCSC &CSC, bool sorted )
00329 {
00330     MatrixCOOtoCSC( COO.coo_val, COO.coo_row, COO.coo_col, CSC.csc_val, CSC.csc_kk, CSC.csc_first, sorted );
00331 }
00332 
00333 } // end namespace Sparse
00334 } // end namespace SPML
00336 
```

8.41 Файл sparse.hpp

Работа с разреженными матрицами

```
#include <limits>
#include <armadillo>
#include <algorithm>
#include <compare.hpp>
```

Классы

- struct SPML::Sparse::CMatrixCOO

  Структура хранения матрицы в координатном COO формате (Coordinate list)
- struct SPML::Sparse::CMatrixCSR

  Структура хранения матрицы в CSR формате (построчно) (Compressed Sparse Row Yale format)
- struct SPML::Sparse::CMatrixCSC

  Структура хранения матрицы в CSC формате (по столбцам) (Compressed Sparse Column Yale format)
- struct SPML::Sparse::CKeyCOO

  Ключ элемента Aij матрицы A в COO формате (Coordinate list)

Пространства имен

- namespace SPML

  Специальная библиотека программных модулей (СБ ПМ)
- namespace SPML::Sparse

  Решение задачи о назначениях

Функции

- void SPML::Sparse::MatrixDenseToCOO (const arma::mat &A, std::vector< double > &coo_val, std::vector< int > &coo_row, std::vector< int > &coo_col)

  Преобразование плотной матрицы в COO формат (Coordinated list)
- void SPML::Sparse::MatrixDenseToCOO (const arma::mat &A, CMatrixCOO &COO)

  Преобразование плотной матрицы в COO формат (Coordinated list)
- void SPML::Sparse::MatrixCOOtoDense (const std::vector< double > &coo_val, const std::vector< int > &coo_row, const std::vector< int > &coo_col, arma::mat &A)

  Преобразование матрицы из COO формата (Coordinated list) в плотную матрицу
- void SPML::Sparse::MatrixCOOtoDense (const CMatrixCOO &COO, arma::mat &A)

  Преобразование матрицы из COO формата (Coordinated list) в плотную матрицу
- void SPML::Sparse::MatrixDenseToCSR (const arma::mat &A, std::vector< double > &csr_val, std::vector< int > &csr_kk, std::vector< int > &csr_first)

  Преобразование плотной матрицы в CSR формат (Compressed Sparse Row Yale format)
- void SPML::Sparse::MatrixDenseToCSR (const arma::mat &A, CMatrixCSR &CSR)

  Преобразование плотной матрицы в CSR формат (Compressed Sparse Row Yale format)
- void SPML::Sparse::MatrixCSRtoDense (const std::vector< double > &csr_val, const std::vector< int > &csr_kk, const std::vector< int > &csr_first, arma::mat &A)

  Преобразование матрицы из CSR формата (Compressed Sparse Row Yale format) в плотную матрицу
- void SPML::Sparse::MatrixCSRtoDense (const CMatrixCSR &CSR, arma::mat &A)

  Преобразование матрицы из CSR формата (Compressed Sparse Row Yale format) в плотную матрицу
- void SPML::Sparse::MatrixDenseToCSC (const arma::mat &A, std::vector< double > &csc_val, std::vector< int > &csc_kk, std::vector< int > &csc_first)

  Преобразование плотной матрицы в CSC формат (Compressed Sparse Column Yale format)
- void SPML::Sparse::MatrixDenseToCSC (const arma::mat &A, CMatrixCSC &CSC)

  Преобразование плотной матрицы в CSC формат (Compressed Sparse Column Yale format)
- void SPML::Sparse::MatrixCSCtoDense (const std::vector< double > &csc_val, const std::vector< int > &csc_kk, const std::vector< int > &csc_first, arma::mat &A)

  Преобразование матрицы из CSC формата (Compressed Sparse Column Yale format) в плотную матрицу
- void SPML::Sparse::MatrixCSCtoDense (const CMatrixCSC &CSC, arma::mat &A)

  Преобразование матрицы из CSC формата (Compressed Sparse Column Yale format) в плотную матрицу
- void SPML::Sparse::MatrixCOOtoCSR (const std::vector< double > &coo_val, const std::vector< int > &coo_row, const std::vector< int > &coo_col, std::vector< double > &csr_val, std::vector< int > &csr_kk, std::vector< int > &csr_first, bool sorted=false)

  Преобразование матрицы в COO формате (Coordinate list) в CSR формат (Compressed Sparse Row Yale format)
- void SPML::Sparse::MatrixCOOtoCSR (const CMatrixCOO &COO, CMatrixCSR &CSR, bool sorted=false)

  Преобразование матрицы в COO формате (Coordinate list) в CSR формат (Compressed Sparse Row Yale format)
- void SPML::Sparse::MatrixCOOtoCSC (const std::vector< double > &coo_val, const std::vector< int > &coo_row, const std::vector< int > &coo_col, std::vector< double > &csc_val, std::vector< int > &csc_kk, std::vector< int > &csc_first, bool sorted=false)

Преобразование матрицы в COO формате (Coordinate list) в CSC формат (Compressed Sparse Column Yale format)

- void SPML::Sparse::MatrixCOOtoCSC (const CMatrixCOO &COO, CMatrixCSC &CSC, bool sorted=false)

  Преобразование матрицы в COO формате (Coordinate list) в CSC формат (Compressed Sparse Column Yale format)

8.41.1 Подробное описание

Работа с разреженными матрицами

Перевод матрицы из плотного представления в COO (Coordinate list), CSR (Compressed Sparse Row Yale format), CSC (Compressed Sparse Column Yale format) вид.

Дата

27.05.22 - создан

Автор

Соболев А.А.

См. определение в файле sparse.hpp

8.42 sparse.hpp

См. документацию.

```
00001 //-----------------------------------------------------------------------------------------------------------
00012
00013 #ifndef SPML_SPARSE_H
00014 #define SPML_SPARSE_H
00015
00016 // System includes:
00017 #include <limits>
00018 #include <armadillo>
00019 #include <algorithm>
00020
00021 // SPML includes:
00022 #include <compare.hpp>
00023
00024 namespace SPML
00025 {
00026 namespace Sparse
00027 {
00028 //-----------------------------------------------------------------------------------------------------------
00033 struct CMatrixCOO
00034 {
00035     std::vector<double> coo_val;
00036     std::vector<int> coo_row;
00037     std::vector<int> coo_col;
00038 };
00039
00044 struct CMatrixCSR
00045 {
00046     std::vector<double> csr_val;
00047     std::vector<int> csr_kk;
00048     std::vector<int> csr_first;
00049
00053     inline int n_rows() const
00054     {
00055         return static_cast<int>( csr_first.size() ) - 1;
00056     }
00057
00061     inline int n_cols() const
00062     {
```

```
00063         int max_kk = -1;
00064         for( unsigned i = 0; i < csr_kk.size(); i++ ) { // Поиск максимального элемента в массиве kk - это и будет кол-
    во столбцов
00065             if( csr_kk[i] > max_kk ) {
00066                 max_kk = csr_kk[i];
00067             }
00068         }
00069         return ( max_kk + 1 );
00070     }
00071 };
00072
00077 struct CMatrixCSC
00078 {
00079     std::vector<double> csc_val;
00080     std::vector<int> csc_kk;
00081     std::vector<int> csc_first;
00082 };
00083
00084 //---------------------------------------------------------------------------------------------------------------
00092 void MatrixDenseToCOO( const arma::mat &A, std::vector<double> &coo_val, std::vector<int> &coo_row,
00093     std::vector<int> &coo_col );
00094
00100 void MatrixDenseToCOO( const arma::mat &A, CMatrixCOO &COO );
00101
00109 void MatrixCOOtoDense( const std::vector<double> &coo_val, const std::vector<int> &coo_row,
00110     const std::vector<int> &coo_col, arma::mat &A );
00111
00117 void MatrixCOOtoDense( const CMatrixCOO &COO, arma::mat &A );
00118
00119 //---------------------------------------------------------------------------------------------------------------
00128 void MatrixDenseToCSR( const arma::mat &A, std::vector<double> &csr_val, std::vector<int> &csr_kk,
00129     std::vector<int> &csr_first );
00130
00136 void MatrixDenseToCSR( const arma::mat &A, CMatrixCSR &CSR );
00137
00145 void MatrixCSRtoDense( const std::vector<double> &csr_val, const std::vector<int> &csr_kk,
00146     const std::vector<int> &csr_first, arma::mat &A );
00147
00153 void MatrixCSRtoDense( const CMatrixCSR &CSR, arma::mat &A );
00154
00155 //---------------------------------------------------------------------------------------------------------------
00164 void MatrixDenseToCSC( const arma::mat &A, std::vector<double> &csc_val, std::vector<int> &csc_kk,
00165     std::vector<int> &csc_first );
00166
00172 void MatrixDenseToCSC( const arma::mat &A, CMatrixCSC &CSC );
00173
00181 void MatrixCSCtoDense( const std::vector<double> &csc_val, const std::vector<int> &csc_kk,
00182     const std::vector<int> &csc_first, arma::mat &A );
00183
00189 void MatrixCSCtoDense( const CMatrixCSC &CSC, arma::mat &A );
00190
00191 //---------------------------------------------------------------------------------------------------------------
00202 void MatrixCOOtoCSR( const std::vector<double> &coo_val, const std::vector<int> &coo_row,
00203     const std::vector<int> &coo_col, std::vector<double> &csr_val, std::vector<int> &csr_kk,
00204     std::vector<int> &csr_first, bool sorted = false );
00205
00212 void MatrixCOOtoCSR( const CMatrixCOO &COO, CMatrixCSR &CSR, bool sorted = false );
00213
00224 void MatrixCOOtoCSC( const std::vector<double> &coo_val, const std::vector<int> &coo_row,
00225     const std::vector<int> &coo_col, std::vector<double> &csc_val, std::vector<int> &csc_kk,
00226     std::vector<int> &csc_first, bool sorted = false );
00227
00234 void MatrixCOOtoCSC( const CMatrixCOO &COO, CMatrixCSC &CSC, bool sorted = false );
00235
00236 //---------------------------------------------------------------------------------------------------------------
00241 struct CKeyCOO
00242 {
00243 public:
00247     int i() const { return i_; }
00248
00252     int j() const { return j_; }
00253
00257     CKeyCOO() : i_( 0 ), j_( 0 ){}
00258
00264     CKeyCOO( int i, int j ) : i_( i ), j_( j ){}
00265
00266     bool operator <( CKeyCOO const& other ) const
00267     {
00268         if( ( this->i_ < other.i_ ) || ( ( this->i_ == other.i_ ) && ( this->j_ < other.j_ ) ) ) {
00269             return true;
00270         }
00271         return false;
00272     }
00273
00274 private:
00275     int i_;
00276     int j_;
```

```
00277 };
00278
00279 } // end namespace Sparse
00280 } // end namespace SPML
00281 #endif // SPML_SPARSE_H
```

Предметный указатель