

LAP (Linear Assignment Problem)

Программная документация

1 LAP - Linear Assignment Problem / Линейная дискретная оптимизационная задача (задача о назначениях)	1
1.1 1. Brief / Обзор	1
1.2 2. Dependencies / Зависимости	1
1.3 3. Tests / Тесты	1
2 Алфавитный указатель групп	5
2.1 Группы	5
3 Алфавитный указатель пространств имен	7
3.1 Пространства имен	7
4 Алфавитный указатель классов	9
4.1 Классы	9
5 Группы	11
5.1 Spml	11
5.1.1 Подробное описание	11
6 Пространства имен	13
6.1 Пространство имен SPML	13
6.1.1 Подробное описание	13
6.2 Пространство имен SPML::Compare	13
6.2.1 Подробное описание	13
6.2.2 Функции	14
6.3 Пространство имен SPML::LAP	16
6.3.1 Подробное описание	18
6.3.2 Типы	18
6.3.3 Перечисления	18
6.3.4 Функции	18
6.4 Пространство имен SPML::Sparse	29
6.4.1 Подробное описание	30
6.4.2 Функции	30
7 Классы	37
7.1 Структура SPML::Sparse::CKeyCOO	37
7.1.1 Подробное описание	37
7.1.2 Конструктор(ы)	37
7.1.3 Методы	38
7.2 Шаблон структуры SPML::Sparse::CMatrixCOO< T >	38
7.2.1 Подробное описание	38
7.2.2 Данные класса	39
7.3 Шаблон структуры SPML::Sparse::CMatrixCSC< T >	39
7.3.1 Подробное описание	39
7.3.2 Данные класса	39
7.4 Шаблон структуры SPML::Sparse::CMatrixCSR< T >	40

7.4.1 Подробное описание	40
7.4.2 Данные класса	40
Предметный указатель	43

Раздел 1

LAP - Linear Assignment Problem / Линейная дискретная оптимизационная задача (задача о назначениях)

1.1 1. Brief / Обзор

Solving linear assignment problem using / Решение задачи о назначениях методами:

- Jonker-Volgenant-Castanon method (JVC) for dense and sparse (CSR - compressed sparse row) matrices; / Метод Джонкера-Волгенанта-Кастаньона для плотных и разреженных матриц в CSR формате
- Mack method / Метод Мака;
- Hungarian (Munkres) method / Венгерский алгоритм.

1.2 2. Dependencies / Зависимости

Armadillo for matrices, Boost for testing / Armadillo для работы с матрицами, Boost для тестирования.

1.3 3. Tests / Тесты

- Comparison of calculation speed on dense and sparse matrices / Сравнение скорости работы на плотных и разреженных матрицах;
- Simple assignment problem matrices are provided / Дополнительные тесты на простых матрицах;
- test JVC algorithm for looping / Тест алгоритма JVCdense на заикливание.

Results for time measuring / Результаты замеров скорости работы:

<? <?

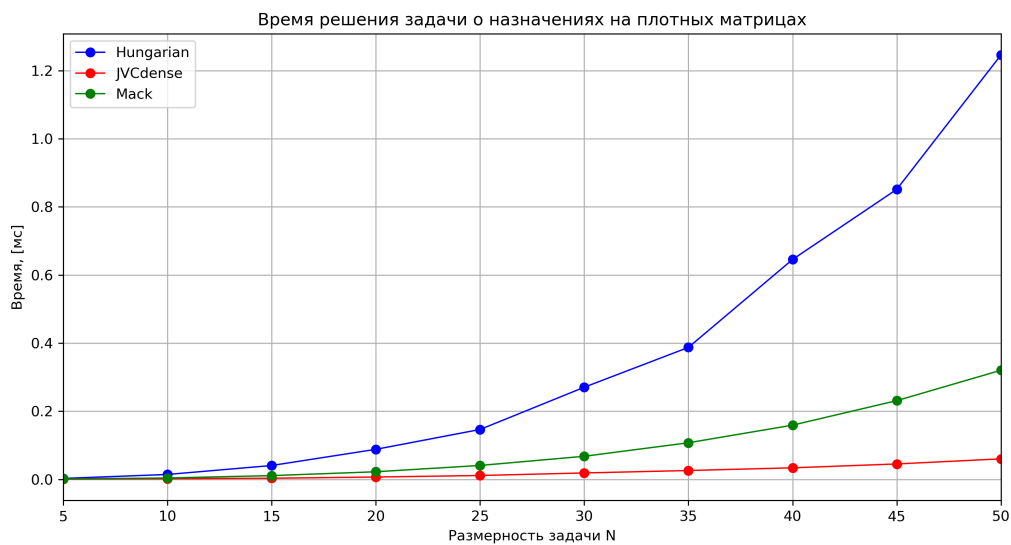


Рис. 1.1 ?>

Fig.1 - Execution time for dense matrices (small dimensions)

Рис.1 - Время выполнения на плотных матрицах (малые размерности)

<? <?



Рис. 1.2 ?>

Fig.2 - Execution time for dense matrices (large dimensions)

Рис.2 - Время выполнения на плотных матрицах (большие размерности)

<? <?

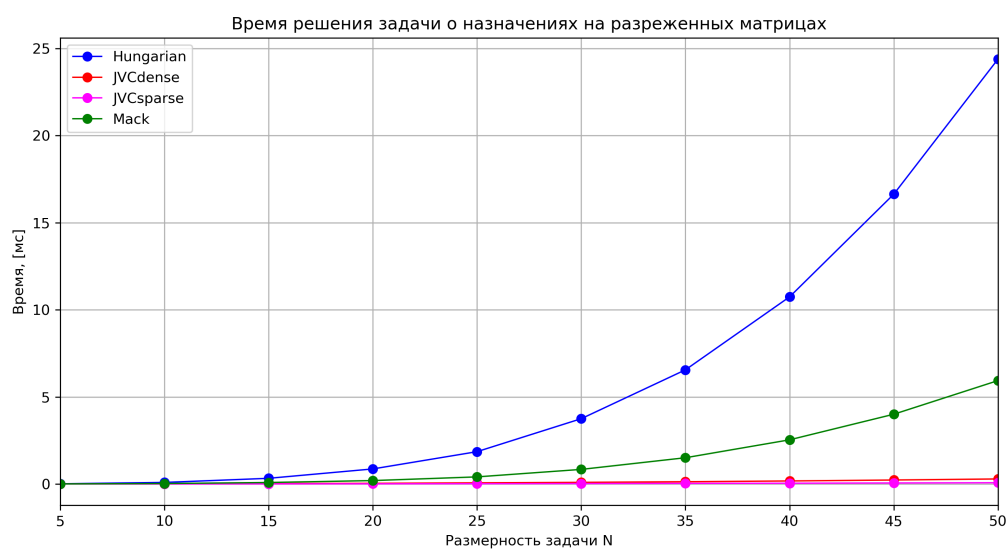


Рис. 1.3 ?>

Fig.3 - Execution time for sparse matrices (small dimensions)

Рис.3 - Время выполнения на разреженных матрицах (малые размерности)

<? <?



Рис. 1.4 ?>

Fig.4 - Execution time for sparse matrices (large dimensions)

Рис.4 - Время выполнения на разреженных матрицах (большие размерности)

Раздел 2

Алфавитный указатель групп

2.1 Группы

Полный список групп.

Spm1 [11](#)

Раздел 3

Алфавитный указатель пространств имен

3.1 Пространства имен

Полный список пространств имен.

| | | |
|-------------------------------|--|----|
| SPML | Специальная библиотека программных модулей (СБ ПМ) | 13 |
| SPML::Compare | Сравнение чисел | 13 |
| SPML::LAP | Решение задачи о назначениях | 16 |
| SPML::Sparse | Разреженные матрицы | 29 |

Раздел 4

Алфавитный указатель классов

4.1 Классы

Классы с их кратким описанием.

| | |
|---|----|
| SPML::Sparse::CKeyCOO | |
| Ключ элемента A_{ij} матрицы A в COO формате (Coordinate list) | 37 |
| SPML::Sparse::CMatrixCOO< T > | |
| Структура хранения матрицы в COO формате (Coordinate list) | 38 |
| SPML::Sparse::CMatrixCSC< T > | |
| Структура хранения матрицы в CSC формате (Compressed Sparse Column Yale format) | 39 |
| SPML::Sparse::CMatrixCSR< T > | |
| Структура хранения матрицы в CSR формате (Compressed Sparse Row Yale format) | 40 |

Раздел 5

Группы

5.1 Spml

Пространства имен

- namespace [SPML](#)

Специальная библиотека программных модулей (СБ ПМ)

5.1.1 Подробное описание

Раздел 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](#)
Разреженные матрицы

6.1.1 Подробное описание

Специальная библиотека программных модулей (СБ ПМ)

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: $\text{abs}(\text{first} - \text{second}) < \text{eps}$

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.h](#) 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: $\text{abs}(\text{first} - \text{second}) < \text{eps}$

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.h](#) 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)

Возвращает результат: $(\text{abs}((\text{first} - \text{second}) / \text{first}) < \text{eps}) \&\& (\text{abs}((\text{first} - \text{second}) / \text{second}) < \text{eps})$

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.h](#) 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)

Возвращает результат: $(\text{abs}(\text{first} - \text{second}) / \text{first}) < \text{eps}) \&\& (\text{abs}(\text{first} - \text{second}) / \text{second}) < \text{eps})$

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.h](#) 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: $\text{abs}(\text{value}) < \text{eps}$

Аргументы

| | | |
|----|-------|---------------------------------|
| in | value | - проверяемое число |
| in | eps | - абсолютная точность сравнения |

Возвращает

true - если разница меньше точности, иначе false

См. определение в файле [compare.h](#) 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.h](#) строка 90

6.3 Пространство имен SPML::LAP

Решение задачи о назначениях

Определения типов

- `typedef int(* fp_function_t) (const unsigned, const std::vector< double > &, const std::vector< int > &, const std::vector< int > &, const int, int *, double *, int *, double, double)`

Перечисления

- `enum TSearchParam { SP_Min , SP_Max }`
Критерий поиска - минимум/максимум для задачи о назначениях
- `enum TFindPath { FP_1 , FP_2 , FP_DYNAMIC }`
Способ нахождения пути в алгоритма JVC для ленточных матриц

Функции

- `void JVCdense (const arma::mat &assigncost, int dim, TSearchParam sp, double maxcost, double resolution, arma::ivec &rowsol, double &lapcost)`
Метод Джонкера-Волгенанта-Кастаньона (Jonker-Volgenant-Castanon) решения задачи о назначениях для плотных матриц
- `void JVCsparse (const std::vector< double > &cc, const std::vector< int > &kk, const std::vector< int > &ii, unsigned dim, TSearchParam sp, double maxcost, double resolution, arma::ivec &rowsol, double &lapcost, TFindPath fp=TFindPath::FP_1)`
Метод Джонкера-Волгенанта-Кастаньона (Jonker-Volgenant-Castanon) решения задачи о назначениях для разреженных матриц
- `int JVCsparseNEW (const std::vector< double > &cc, const std::vector< int > &kk, const std::vector< int > &first, TSearchParam sp, double maxcost, double resolution, arma::ivec &rowsol, double &lapcost)`
- `void JVCsparse (const SPML::Sparse::CMatrixCSR< double > &csr, unsigned dim, TSearchParam sp, double maxcost, double resolution, arma::ivec &rowsol, double &lapcost, TFindPath fp=TFindPath::FP_DYNAMIC)`
Метод Джонкера-Волгенанта-Кастаньона (Jonker-Volgenant-Castanon) решения задачи о назначениях для разреженных матриц
- `void Mack (const arma::mat &assigncost, int dim, TSearchParam sp, double maxcost, double resolution, arma::ivec &rowsol, double &lapcost)`
Метод Мака решения задачи о назначениях

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

Венгерский метод решения задачи о назначениях (Метод Мункреса)

- 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)
- int [jvc_sparse_ccrrt_sparse_](#) (const unsigned n, const std::vector< double > &cc, const std::vector< int > &ii, const std::vector< int > &kk, int *free_rows, int *x, int *y, double *v, double maxcost, double resolution)
- int [jvc_sparse_carr_sparse](#) (const unsigned n, const std::vector< double > &cc, const std::vector< int > &ii, const std::vector< int > &kk, const unsigned n_free_rows, int *free_rows, int *x, int *y, double *v, double maxcost, double resolution)
- unsigned [jvc_sparse_find_sparse_1](#) (const unsigned n, unsigned lo, double *d, int *cols, int *y, double resolution)
- int [jvc_sparse_find_sparse_2](#) (double *d, int *scan, const unsigned n_todo, int *todo, bool *done, double maxcost, double resolution)
- int [jvc_sparse_scan_sparse_1](#) (const unsigned n, const std::vector< double > &cc, const std::vector< int > &ii, const std::vector< int > &kk, unsigned *plo, unsigned *phi, double *d, int *cols, int *pred, int *y, double *v, double resolution)
- int [jvc_sparse_scan_sparse_2](#) (const unsigned n, const std::vector< double > &cc, const std::vector< int > &ii, const std::vector< int > &kk, unsigned *plo, unsigned *phi, double *d, int *pred, bool *done, unsigned *pn_ready, int *ready, int *scan, unsigned *pn_todo, int *todo, bool *added, int *y, double *v, double resolution)
- int [jvc_sparse_find_path_sparse_1](#) (const unsigned n, const std::vector< double > &cc, const std::vector< int > &ii, const std::vector< int > &kk, const int start_i, int *y, double *v, int *pred, double maxcost, double resolution)
- int [jvc_sparse_find_path_sparse_2](#) (const unsigned n, const std::vector< double > &cc, const std::vector< int > &ii, const std::vector< int > &kk, const int start_i, int *y, double *v, int *pred, double maxcost, double resolution)
- int [jvc_sparse_find_path_sparse_dynamic](#) (const unsigned n, const std::vector< double > &cc, const std::vector< int > &ii, const std::vector< int > &kk, const int start_i, int *y, double *v, int *pred, double maxcost, double resolution)
- [fp_function_t jvc_sparse_get_better_find_path](#) (const unsigned n, const std::vector< int > &ii)

- `int jvc_sparse_ca_sparse` (const unsigned n, const std::vector< double > &cc, const std::vector< int > &i, const std::vector< int > &kk, const unsigned n_free_rows, int *free_rows, int *x, int *y, double *v, int fp_version, double maxcost, double resolution)
- `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 maxcost, bool &fail)

6.3.1 Подробное описание

Решение задачи о назначениях

6.3.2 Типы

6.3.2.1 fp_function_t

`typedef int(* SPML::LAP::fp_function_t)` (const unsigned, const std::vector< double > &, const std::vector< int > &, const std::vector< int > &, const int, int *, double *, int *, double, double)

См. определение в файле [lap_jvc_sparse.cpp](#) строка 538

6.3.3 Перечисления

6.3.3.1 TFindPath

enum [SPML::LAP::TFindPath](#)

Способ нахождения пути в алгоритме JVC для ленточных матриц

Элементы перечислений

| | |
|------------|--|
| FP_1 | |
| FP_2 | |
| FP_DYNAMIC | |

См. определение в файле [lap.h](#) строка 40

6.3.3.2 TSearchParam

enum [SPML::LAP::TSearchParam](#)

Критерий поиска - минимум/максимум для задачи о назначениях

Элементы перечислений

| | |
|--------|-----------------|
| SP_Min | Поиск минимума |
| SP_Max | Поиск максимума |

См. определение в файле [lap.h](#) строка 31

6.3.4 Функции

6.3.4.1 Hungarian()

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

Венгерский метод решения задачи о назначениях (Метод Мункреса)

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] |
| in | dim | - порядок квадратной матрицы ценности и размерность результата res соответственно |
| in | sp | - критерий поиска (минимум/максимум) |
| out | rowsol | - результат задачи о назначениях, размерность [dim] (индекс макс/мин элемента в i-ой строке) rowsol[i] = j --> в i-ой строке выбран j-ый элемент |

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.

См. определение в файле [lap_hungarian.cpp](#) строка 384

6.3.4.2 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 [lap_hungarian.cpp](#) строка 261

6.3.4.3 hungarian_clear_covers()

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

См. определение в файле [lap_hungarian.cpp](#) строка 273

6.3.4.4 hungarian_erase_primes()

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

См. определение в файле [lap_hungarian.cpp](#) строка 279

6.3.4.5 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 [lap_hungarian.cpp](#) строка 110

6.3.4.6 `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 [lap_hungarian.cpp](#) строка 247

6.3.4.7 `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 [lap_hungarian.cpp](#) строка 342

6.3.4.8 `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 [lap_hungarian.cpp](#) строка 232

6.3.4.9 `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 [lap_hungarian.cpp](#) строка 165

6.3.4.10 `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 [lap_hungarian.cpp](#) строка 153

6.3.4.11 `hungarian_step_1()`

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

См. определение в файле [lap_hungarian.cpp](#) строка 18

6.3.4.12 `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 continuously, 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 [lap_hungarian.cpp](#) строка 39

6.3.4.13 `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 [lap_hungarian.cpp](#) строка 77

6.3.4.14 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 [lap_hungarian.cpp](#) строка 180

6.3.4.15 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 $2 * N \times 2$. In it all vertices between rows and columns are stored row-wise. Now, we can set the complete step 5:

См. определение в файле [lap_hungarian.cpp](#) строка 296

6.3.4.16 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 [lap_hungarian.cpp](#) строка 359

6.3.4.17 jvc_sparse_ca_sparse()

```
int SPML::LAP::jvc_sparse_ca_sparse (
    const unsigned n,
    const std::vector< double > & cc,
```

```

const std::vector< int > & ii,
const std::vector< int > & kk,
const unsigned n_free_rows,
int * free_rows,
int * x,
int * y,
double * v,
int fp_version,
double maxcost,
double resolution )

```

См. определение в файле [lap_jvc_sparse.cpp](#) строка 554

6.3.4.18 jvc_sparse_carr_sparse()

```

int SPML::LAP::jvc_sparse_carr_sparse (
    const unsigned n,
    const std::vector< double > & cc,
    const std::vector< int > & ii,
    const std::vector< int > & kk,
    const unsigned n_free_rows,
    int * free_rows,
    int * x,
    int * y,
    double * v,
    double maxcost,
    double resolution )

```

См. определение в файле [lap_jvc_sparse.cpp](#) строка 86

6.3.4.19 jvc_sparse_ccrrt_sparse_()

```

int SPML::LAP::jvc_sparse_ccrrt_sparse_ (
    const unsigned n,
    const std::vector< double > & cc,
    const std::vector< int > & ii,
    const std::vector< int > & kk,
    int * free_rows,
    int * x,
    int * y,
    double * v,
    double maxcost,
    double resolution )

```

См. определение в файле [lap_jvc_sparse.cpp](#) строка 20

6.3.4.20 jvc_sparse_find_path_sparse_1()

```

int SPML::LAP::jvc_sparse_find_path_sparse_1 (
    const unsigned n,
    const std::vector< double > & cc,
    const std::vector< int > & ii,
    const std::vector< int > & kk,
    const int start_i,
    int * y,
    double * v,
    int * pred,
    double maxcost,
    double resolution )

```

См. определение в файле [lap_jvc_sparse.cpp](#) строка 347

6.3.4.21 jvc_sparse_find_path_sparse_2()

```
int SPML::LAP::jvc_sparse_find_path_sparse_2 (
    const unsigned n,
    const std::vector< double > & cc,
    const std::vector< int > & ii,
    const std::vector< int > & kk,
    const int start_i,
    int * y,
    double * v,
    int * pred,
    double maxcost,
    double resolution )
```

См. определение в файле [lap_jvc_sparse.cpp](#) строка 411

6.3.4.22 jvc_sparse_find_path_sparse_dynamic()

```
int SPML::LAP::jvc_sparse_find_path_sparse_dynamic (
    const unsigned n,
    const std::vector< double > & cc,
    const std::vector< int > & ii,
    const std::vector< int > & kk,
    const int start_i,
    int * y,
    double * v,
    int * pred,
    double maxcost,
    double resolution )
```

См. определение в файле [lap_jvc_sparse.cpp](#) строка 526

6.3.4.23 jvc_sparse_find_sparse_1()

```
unsigned SPML::LAP::jvc_sparse_find_sparse_1 (
    const unsigned n,
    unsigned lo,
    double * d,
    int * cols,
    int * y,
    double resolution )
```

См. определение в файле [lap_jvc_sparse.cpp](#) строка 165

6.3.4.24 jvc_sparse_find_sparse_2()

```
int SPML::LAP::jvc_sparse_find_sparse_2 (
    double * d,
    int * scan,
    const unsigned n_todo,
    int * todo,
    bool * done,
    double maxcost,
    double resolution )
```

См. определение в файле [lap_jvc_sparse.cpp](#) строка 185

6.3.4.25 jvc_sparse_get_better_find_path()

```
fp_function_t SPML::LAP::jvc_sparse_get_better_find_path (
    const unsigned n,
    const std::vector< int > & ii )
```

См. определение в файле [lap_jvc_sparse.cpp](#) строка 541

6.3.4.26 jvc_sparse_scan_sparse_1()

```
int SPML::LAP::jvc_sparse_scan_sparse_1 (
    const unsigned n,
    const std::vector< double > & cc,
    const std::vector< int > & ii,
    const std::vector< int > & kk,
    unsigned * plo,
    unsigned * phi,
    double * d,
    int * cols,
    int * pred,
    int * y,
    double * v,
    double resolution )
```

См. определение в файле [lap_jvc_sparse.cpp](#) строка 207

6.3.4.27 jvc_sparse_scan_sparse_2()

```
int SPML::LAP::jvc_sparse_scan_sparse_2 (
    const unsigned n,
    const std::vector< double > & cc,
    const std::vector< int > & ii,
    const std::vector< int > & kk,
    unsigned * plo,
    unsigned * phi,
    double * d,
    int * pred,
    bool * done,
    unsigned * pn_ready,
    int * ready,
    int * scan,
    unsigned * pn_todo,
    int * todo,
    bool * added,
    int * y,
    double * v,
    double resolution )
```

См. определение в файле [lap_jvc_sparse.cpp](#) строка 273

6.3.4.28 JVCdense()

```
void SPML::LAP::JVCdense (
    const arma::mat & assigncost,
    int dim,
    TSearchParam sp,
    double maxcost,
    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 | maxcost | - модуль максимального элемента матрицы assigncost |
| 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 --> в i-ой строке выбран j-ый элемент |
| out | lapcost | - сумма назначенных элементов матрицы ценности assigncost |

См. определение в файле [lap_jvc_dense.cpp](#) строка 18

6.3.4.29 JVCsparse() [1/2]

```
void SPML::LAP::JVCsparse (
    const SPML::Sparse::CMatrixCSR< double > & csr,
    unsigned dim,
    TSearchParam sp,
    double maxcost,
    double resolution,
    arma::ivec & rowsol,
    double & lapcost,
    TFindPath fp = TFindPath::FP_DYNAMIC )
```

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 | dim | - порядок квадратной матрицы ценности и размерность результата res соответственно |
| in | sp | - критерий поиска (минимум/максимум) |
| in | maxcost | - модуль максимального элемента матрицы assigncost |

Аргументы

| | | |
|-----|------------|--|
| 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 --> в i-ой строке выбран j-ый элемент |
| out | lapcost | - сумма назначенных элементов матрицы ценности assigncost |
| in | fp | - find path mode (FP_DYNAMIC - default) |

См. определение в файле [lap_jvc_sparse.cpp](#) строка 649

6.3.4.30 JVCsparse() [2/2]

```
void SPML::LAP::JVCsparse (
    const std::vector< double > & cc,
    const std::vector< int > & kk,
    const std::vector< int > & ii,
    unsigned dim,
    TSearchParam sp,
    double maxcost,
    double resolution,
    arma::ivec & rowsol,
    double & lapcost,
    TFindPath fp = TFindPath::FP_1 )
```

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

Аргументы

| | | |
|-----|------------|--|
| in | cc | - |
| in | kk | - |
| in | ii | - |
| in | dim | - порядок квадратной матрицы ценности и размерность результата res соответственно |
| in | sp | - критерий поиска (минимум/максимум) |
| in | maxcost | - модуль максимального элемента матрицы assigncost |
| 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 --> в i-ой строке выбран j-ый элемент |
| out | lapcost | - сумма назначенных элементов матрицы ценности assigncost |
| in | fp | - find path mode (FP_DYNAMIC - default) |

См. определение в файле [lap_jvc_sparse.cpp](#) строка 603

6.3.4.31 JVCsparseNEW()

```
int SPML::LAP::JVCsparseNEW (
    const std::vector< double > & cc,
    const std::vector< int > & kk,
    const std::vector< int > & first,
    TSearchParam sp,
    double maxcost,
    double resolution,
    arma::ivec & rowsol,
```

```
double & lapcost )
```

См. определение в файле [lap_jvc_sparse2.cpp](#) строка 150

6.3.4.32 Mack()

```
void SPML::LAP::Mack (
    const arma::mat & assigncost,
    int dim,
    TSearchParam sp,
    double maxcost,
    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) |
| out | rowsol | - результат задачи о назначениях, размерность [dim] (индекс макс/мин элемента в i-ой строке) rowsol[i] = j --> в i-ой строке выбран j-ый элемент |

См. определение в файле [lap_mack.cpp](#) строка 18

6.3.4.33 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 maxcost,
    bool & fail )
```

См. определение в файле [lap_jvc_sparse2.cpp](#) строка 47

6.3.4.34 updateAssignments()

```
void SPML::LAP::updateAssignments (
    arma::ivec & lab,
    arma::ivec & y,
```



```
arma::ivec & x,
int j,
int i0 )
```

См. определение в файле [lap_jvc_sparse2.cpp](#) строка 27

6.3.4.35 updateDual()

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

См. определение в файле [lap_jvc_sparse2.cpp](#) строка 18

6.4 Пространство имен SPML::Sparse

Разреженные матрицы

Классы

- struct [CKeyCOO](#)
Ключ элемента A_{ij} матрицы A в COO формате (Coordinate list)
- struct [CMatrixCOO](#)
Структура хранения матрицы в COO формате (Coordinate list)
- struct [CMatrixCSC](#)
Структура хранения матрицы в CSC формате (Compressed [Sparse](#) Column Yale format)
- struct [CMatrixCSR](#)
Структура хранения матрицы в CSR формате (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)
- template<typename T >
void [MatrixDenseToCOO](#) (const arma::mat &A, [CMatrixCOO](#)< T > &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
- template<typename T >
void [MatrixCOOtoDense](#) (const [CMatrixCOO](#)< T > &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_first, std::vector< int > &csr_kk)
Преобразование плотной матрицы в CSR формат (Compressed [Sparse](#) Row Yale format)
- template<typename T >
void [MatrixDenseToCSR](#) (const arma::mat &A, [CMatrixCSR](#)< T > &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_first, const std::vector< int > &csr_kk, arma::mat &A)
Преобразование матрицы из CSR формата (Compressed [Sparse](#) Row Yale format) в плотную матрицу

- `template<typename T >`
`void MatrixCSRtoDense (const CMatrixCSR< T > &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_first, std::vector< int > &csc_kk)`
 Преобразование плотной матрицы в CSC формат (Compressed [Sparse](#) Column Yale format)
- `template<typename T >`
`void MatrixDenseToCSC (const arma::mat &A, CMatrixCSC< T > &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_first, const std::vector< int > &csc_kk, arma::mat &A)`
 Преобразование матрицы из CSC формата (Compressed [Sparse](#) Column Yale format) в плотную матрицу
- `template<typename T >`
`void MatrixCSCtoDense (const CMatrixCSC< T > &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_first, std::vector< int > &csr_kk, bool sorted=false)`
 Преобразование матрицы в COO формате (Coordinate list) в CSR формат (Compressed [Sparse](#) Row Yale format)
- `template<typename T >`
`void MatrixCOOtoCSR (const CMatrixCOO< T > &COO, CMatrixCSR< T > &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_first, std::vector< int > &csc_kk, bool sorted=false)`
 Преобразование матрицы в COO формате (Coordinate list) в CSC формат (Compressed [Sparse](#) Column Yale format)
- `template<typename T >`
`void MatrixCOOtoCSC (const CMatrixCOO< T > &COO, CMatrixCSC< T > &CSC, bool sorted=false)`
 Преобразование матрицы в COO формате (Coordinate list) в CSC формат (Compressed [Sparse](#) Column Yale format)

6.4.1 Подробное описание

Разреженные матрицы

Решение задачи о назначениях

6.4.2 Функции

6.4.2.1 `MatrixCOOtoCSC()` [1/2]

```
template<typename T >
void SPML::Sparse::MatrixCOOtoCSC (
    const CMatrixCOO< T > & COO,
    CMatrixCSC< T > & 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.h](#) строка 327

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_first,
    std::vector< int > & csc_kk,
    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_first | - вектор индексов строк ненулевых элементов, размер равен количеству ненулевых элементов nnz |
| in | csc_kk | - 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 225

6.4.2.3 MatrixCOOtoCSR() [1/2]

```
template<typename T >
void SPML::Sparse::MatrixCOOtoCSR (
    const CMatrixCOO< T > & COO,
    CMatrixCSR< T > & 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.h](#) строка 292

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_first,
    std::vector< int > & csr_kk,
    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_first | - вектор индексов колонок ненулевых элементов, размер равен количеству ненулевых элементов nnz |
| in | csr_kk | - 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 153

6.4.2.5 MatrixCOOtoDense() [1/2]

```
template<typename T >
void SPML::Sparse::MatrixCOOtoDense (
    const CMatrixCOO< T > & 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.h](#) строка 136

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 43

6.4.2.7 MatrixCSCtoDense() [1/2]

```
template<typename T >
void SPML::Sparse::MatrixCSCtoDense (
    const CMatrixCSC< T > & 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.h](#) строка 256

6.4.2.8 MatrixCSCtoDense() [2/2]

```
void SPML::Sparse::MatrixCSCtoDense (
    const std::vector< double > & csc_val,
    const std::vector< int > & csc_first,
    const std::vector< int > & csc_kk,
    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_first | - вектор индексов строк ненулевых элементов, размер равен количеству ненулевых элементов nnz |
| in | csc_kk | - 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 135

6.4.2.9 MatrixCSRtoDense() [1/2]

```
template<typename T >
void SPML::Sparse::MatrixCSRtoDense (
    const CMatrixCSR< T > & 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.h](#) строка 196

6.4.2.10 MatrixCSRtoDense() [2/2]

```
void SPML::Sparse::MatrixCSRtoDense (
    const std::vector< double > & csr_val,
    const std::vector< int > & csr_first,
    const std::vector< int > & csr_kk,
    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_first | - вектор индексов колонок ненулевых элементов, размер равен количеству ненулевых элементов nnz |
| in | csr_kk | - 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 87

6.4.2.11 MatrixDenseToCOO() [1/2]

```
template<typename T >
void SPML::Sparse::MatrixDenseToCOO (
    const arma::mat & A,
    CMatrixCOO< T > & 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.h](#) строка 107

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]

```
template<typename T >
void SPML::Sparse::MatrixDenseToCSC (
    const arma::mat & A,
    CMatrixCSC< T > & 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.h](#) строка 227

6.4.2.14 MatrixDenseToCSC() [2/2]

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

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_first | - вектор индексов строк ненулевых элементов, размер равен количеству ненулевых элементов nnz |
| out | csc_kk | - вектор начальных смещений в векторе CSR, размер m+1 |

См. определение в файле [sparse.cpp](#) строка 105

6.4.2.15 MatrixDenseToCSR() [1/2]

```
template<typename T >
void SPML::Sparse::MatrixDenseToCSR (
    const arma::mat & A,
    CMatrixCSR< T > & 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.h](#) строка 167

6.4.2.16 MatrixDenseToCSR() [2/2]

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

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_first | - вектор индексов колонок ненулевых элементов, размер равен количеству ненулевых элементов nnz |
| out | csr_kk | - вектор начальных смещений в векторе CSR, размер n+1 |

См. определение в файле [sparse.cpp](#) строка 57

Раздел 7

Классы

7.1 Структура SPML::Sparse::CKeyCOO

Ключ элемента A_{ij} матрицы A в COO формате (Coordinate list)
`#include <sparse.h>`

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`

7.1.1 Подробное описание

Ключ элемента A_{ij} матрицы A в COO формате (Coordinate list)

Внимание

Оператор `<` перегружен для случая построчного хранения

См. определение в файле [sparse.h](#) строка 343

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.h](#) строка 359

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.h](#) строка 366

7.1.3 Методы

7.1.3.1 i()

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

i - индекс строки

См. определение в файле [sparse.h](#) строка 349

7.1.3.2 j()

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

j - индекс столбца

См. определение в файле [sparse.h](#) строка 354

7.1.3.3 operator<()

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

См. определение в файле [sparse.h](#) строка 369

Объявления и описания членов структуры находятся в файле:

- [sparse.h](#)

7.2 Шаблон структуры SPML::Sparse::CMatrixCOO< T >

Структура хранения матрицы в COO формате (Coordinate list)

```
#include <sparse.h>
```

Открытые атрибуты

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

7.2.1 Подробное описание

```
template<typename T>
struct SPML::Sparse::CMatrixCOO< T >
```

Структура хранения матрицы в COO формате (Coordinate list)

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

См. определение в файле [sparse.h](#) строка 33

7.2.2 Данные класса

7.2.2.1 coo_col

template<typename T >
 std::vector<int> [SPML::Sparse::CMatrixCOO< T >::coo_col](#)
 Индексы столбцов ненулевых элементов
 См. определение в файле [sparse.h](#) строка 45

7.2.2.2 coo_row

template<typename T >
 std::vector<int> [SPML::Sparse::CMatrixCOO< T >::coo_row](#)
 Индексы строк ненулевых элементов
 См. определение в файле [sparse.h](#) строка 44

7.2.2.3 coo_val

template<typename T >
 std::vector<T> [SPML::Sparse::CMatrixCOO< T >::coo_val](#)
 Вектор ненулевых элементов матрицы A[n,m] (n - число строк, m - число столбцов), размер nnz.
 См. определение в файле [sparse.h](#) строка 41
 Объявления и описания членов структуры находятся в файле:

- [sparse.h](#)

7.3 Шаблон структуры SPML::Sparse::CMatrixCSC< T >

Структура хранения матрицы в CSC формате (Compressed [Sparse](#) Column Yale format)
 #include <sparse.h>

Открытые атрибуты

- std::vector< T > [csc_val](#)
 Вектор ненулевых элементов матрицы A[n,m] (n - число строк, m - число столбцов), размер nnz.
- std::vector< int > [csc_first](#)
 Вектор индексов колонок ненулевых элементов, размер равен количеству ненулевых элементов nnz.
- std::vector< int > [csc_kk](#)
 вектор начальных смещений в векторе CSC, размер m+1

7.3.1 Подробное описание

template<typename T>
 struct SPML::Sparse::CMatrixCSC< T >

Структура хранения матрицы в CSC формате (Compressed [Sparse](#) Column Yale format)
 Матрица A[n,m] (n - число строк, m - число столбцов)
 См. определение в файле [sparse.h](#) строка 73

7.3.2 Данные класса

7.3.2.1 csc_first

```
template<typename T>
std::vector<int> SPML::Sparse::CMatrixCSC< T >::csc_first
```

Вектор индексов колонок ненулевых элементов, размер равен количеству ненулевых элементов nnz.
См. определение в файле [sparse.h](#) строка 84

7.3.2.2 csc_kk

```
template<typename T>
std::vector<int> SPML::Sparse::CMatrixCSC< T >::csc_kk
```

вектор начальных смещений в векторе CSC, размер m+1
См. определение в файле [sparse.h](#) строка 85

7.3.2.3 csc_val

```
template<typename T>
std::vector<T> SPML::Sparse::CMatrixCSC< T >::csc_val
```

Вектор ненулевых элементов матрицы $A[n,m]$ (n - число строк, m - число столбцов), размер nnz.
См. определение в файле [sparse.h](#) строка 81
Объявления и описания членов структуры находятся в файле:

- [sparse.h](#)

7.4 Шаблон структуры SPML::Sparse::CMatrixCSR< T >

Структура хранения матрицы в CSR формате (Compressed [Sparse](#) Row Yale format)
`#include <sparse.h>`

Открытые атрибуты

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

7.4.1 Подробное описание

```
template<typename T>
struct SPML::Sparse::CMatrixCSR< T >
```

Структура хранения матрицы в CSR формате (Compressed [Sparse](#) Row Yale format)
Матрица $A[n,m]$ (n - число строк, m - число столбцов)
См. определение в файле [sparse.h](#) строка 53

7.4.2 Данные класса

7.4.2.1 csr_first

```
template<typename T>
std::vector<int> SPML::Sparse::CMatrixCSR< T >::csr_first
```

Вектор индексов колонок ненулевых элементов, размер равен количеству ненулевых элементов nnz.
См. определение в файле [sparse.h](#) строка 64

7.4.2.2 csr_kk

```
template<typename T >
```

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

вектор начальных смещений в векторе CSR, размер n+1

См. определение в файле [sparse.h](#) строка 65

7.4.2.3 csr_val

```
template<typename T >
```

```
std::vector<T> SPML::Sparse::CMatrixCSR< T >::csr_val
```

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

См. определение в файле [sparse.h](#) строка 61

Объявления и описания членов структуры находятся в файле:

- [sparse.h](#)

Предметный указатель

- AreEqualAbs
 - SPML::Compare, [14](#)
- AreEqualRel
 - SPML::Compare, [14](#), [15](#)
- CKeyCOO
 - SPML::Sparse::CKeyCOO, [37](#)
- coo_col
 - SPML::Sparse::CMatrixCOO< T >, [39](#)
- coo_row
 - SPML::Sparse::CMatrixCOO< T >, [39](#)
- coo_val
 - SPML::Sparse::CMatrixCOO< T >, [39](#)
- csc_first
 - SPML::Sparse::CMatrixCSC< T >, [39](#)
- csc_kk
 - SPML::Sparse::CMatrixCSC< T >, [40](#)
- csc_val
 - SPML::Sparse::CMatrixCSC< T >, [40](#)
- csr_first
 - SPML::Sparse::CMatrixCSR< T >, [40](#)
- csr_kk
 - SPML::Sparse::CMatrixCSR< T >, [40](#)
- csr_val
 - SPML::Sparse::CMatrixCSR< T >, [41](#)
- FP_1
 - SPML::LAP, [18](#)
- FP_2
 - SPML::LAP, [18](#)
- FP_DYNAMIC
 - SPML::LAP, [18](#)
- fp_function_t
 - SPML::LAP, [18](#)
- Hungarian
 - SPML::LAP, [18](#)
- hungarian_augment_path
 - SPML::LAP, [19](#)
- hungarian_clear_covers
 - SPML::LAP, [19](#)
- hungarian_erase_primes
 - SPML::LAP, [19](#)
- hungarian_find_noncovered_zero
 - SPML::LAP, [19](#)
- hungarian_find_prime_in_row
 - SPML::LAP, [20](#)
- hungarian_find_smallest
 - SPML::LAP, [20](#)
- hungarian_find_star_in_col
 - SPML::LAP, [20](#)
- hungarian_find_star_in_row
 - SPML::LAP, [20](#)
- hungarian_star_in_row
 - SPML::LAP, [21](#)
- hungarian_step_1
 - SPML::LAP, [21](#)
- hungarian_step_2
 - SPML::LAP, [21](#)
- hungarian_step_3
 - SPML::LAP, [21](#)
- hungarian_step_4
 - SPML::LAP, [22](#)
- hungarian_step_5
 - SPML::LAP, [22](#)
- hungarian_step_6
 - SPML::LAP, [22](#)
- i
 - SPML::Sparse::CKeyCOO, [38](#)
- IsZeroAbs
 - SPML::Compare, [15](#)
- j
 - SPML::Sparse::CKeyCOO, [38](#)
- jvc_sparse_ca_sparse
 - SPML::LAP, [22](#)
- jvc_sparse_carr_sparse
 - SPML::LAP, [23](#)
- jvc_sparse_ccrrt_sparse_
 - SPML::LAP, [23](#)
- jvc_sparse_find_path_sparse_1
 - SPML::LAP, [23](#)
- jvc_sparse_find_path_sparse_2
 - SPML::LAP, [24](#)
- jvc_sparse_find_path_sparse_dynamic
 - SPML::LAP, [24](#)
- jvc_sparse_find_sparse_1
 - SPML::LAP, [24](#)
- jvc_sparse_find_sparse_2
 - SPML::LAP, [24](#)
- jvc_sparse_get_better_find_path
 - SPML::LAP, [24](#)
- jvc_sparse_scan_sparse_1
 - SPML::LAP, [25](#)
- jvc_sparse_scan_sparse_2
 - SPML::LAP, [25](#)
- JVCdense
 - SPML::LAP, [25](#)

- JVCsparse
 - SPML::LAP, 26, 27
- JVCsparseNEW
 - SPML::LAP, 27
- Mack
 - SPML::LAP, 28
- MatrixCOOtoCSC
 - SPML::Sparse, 30, 31
- MatrixCOOtoCSR
 - SPML::Sparse, 31, 32
- MatrixCOOtoDense
 - SPML::Sparse, 32
- MatrixCSCtoDense
 - SPML::Sparse, 33
- MatrixCSRtoDense
 - SPML::Sparse, 33, 34
- MatrixDenseToCOO
 - SPML::Sparse, 34
- MatrixDenseToCSC
 - SPML::Sparse, 35
- MatrixDenseToCSR
 - SPML::Sparse, 35, 36
- operator<
 - SPML::Sparse::CKeyCOO, 38
- solveForOneL
 - SPML::LAP, 28
- SP_Max
 - SPML::LAP, 18
- SP_Min
 - SPML::LAP, 18
- SPML, 13
- Spml, 11
- SPML::Compare, 13
 - AreEqualAbs, 14
 - AreEqualRel, 14, 15
 - IsZeroAbs, 15
- SPML::LAP, 16
 - FP_1, 18
 - FP_2, 18
 - FP_DYNAMIC, 18
 - fp_function_t, 18
 - Hungarian, 18
 - hungarian_augment_path, 19
 - hungarian_clear_covers, 19
 - hungarian_erase_primes, 19
 - hungarian_find_noncovered_zero, 19
 - hungarian_find_prime_in_row, 20
 - hungarian_find_smallest, 20
 - hungarian_find_star_in_col, 20
 - hungarian_find_star_in_row, 20
 - hungarian_star_in_row, 21
 - hungarian_step_1, 21
 - hungarian_step_2, 21
 - hungarian_step_3, 21
 - hungarian_step_4, 22
 - hungarian_step_5, 22
 - hungarian_step_6, 22
 - jvc_sparse_ca_sparse, 22
 - jvc_sparse_carr_sparse, 23
 - jvc_sparse_ccrrt_sparse_, 23
 - jvc_sparse_find_path_sparse_1, 23
 - jvc_sparse_find_path_sparse_2, 24
 - jvc_sparse_find_path_sparse_dynamic, 24
 - jvc_sparse_find_sparse_1, 24
 - jvc_sparse_find_sparse_2, 24
 - jvc_sparse_get_better_find_path, 24
 - jvc_sparse_scan_sparse_1, 25
 - jvc_sparse_scan_sparse_2, 25
 - JVCdense, 25
 - JVCsparse, 26, 27
 - JVCsparseNEW, 27
 - Mack, 28
 - solveForOneL, 28
 - SP_Max, 18
 - SP_Min, 18
 - TFindPath, 18
 - TSearchParam, 18
 - updateAssignments, 28
 - updateDual, 29
- SPML::Sparse, 29
 - MatrixCOOtoCSC, 30, 31
 - MatrixCOOtoCSR, 31, 32
 - MatrixCOOtoDense, 32
 - MatrixCSCtoDense, 33
 - MatrixCSRtoDense, 33, 34
 - MatrixDenseToCOO, 34
 - MatrixDenseToCSC, 35
 - MatrixDenseToCSR, 35, 36
- SPML::Sparse::CKeyCOO, 37
 - CKeyCOO, 37
 - i, 38
 - j, 38
 - operator<, 38
- SPML::Sparse::CMatrixCOO< T >, 38
 - coo_col, 39
 - coo_row, 39
 - coo_val, 39
- SPML::Sparse::CMatrixCSC< T >, 39
 - csc_first, 39
 - csc_kk, 40
 - csc_val, 40
- SPML::Sparse::CMatrixCSR< T >, 40
 - csr_first, 40
 - csr_kk, 40
 - csr_val, 41
- TFindPath
 - SPML::LAP, 18
- TSearchParam
 - SPML::LAP, 18
- updateAssignments
 - SPML::LAP, 28
- updateDual
 - SPML::LAP, 29