

marathon

0.31

Generated by Doxygen 1.8.6

Sat Apr 9 2016 20:18:05

Contents

1	Hierarchical Index	1
1.1	Class Hierarchy	1
2	Class Index	3
2.1	Class List	3
3	Class Documentation	5
3.1	marathon::chain::matching::BipartiteMatching Class Reference	5
3.1.1	Detailed Description	5
3.1.2	Member Function Documentation	6
3.1.2.1	compare_to	6
3.1.2.2	hash_value	6
3.1.2.3	to_string	6
3.2	marathon::chain::matching::Broder86 Class Reference	6
3.2.1	Detailed Description	7
3.2.2	Member Function Documentation	7
3.2.2.1	computeArbitraryState	7
3.2.2.2	computeNeighbours	7
3.2.2.3	parseInstance	7
3.3	marathon::chain::bipgraph::KannanPath::cycle_comparator Struct Reference	7
3.3.1	Detailed Description	8
3.4	marathon::chain::bipgraph::DenseBipartiteGraph Class Reference	8
3.4.1	Detailed Description	8
3.4.2	Member Function Documentation	9
3.4.2.1	compare_to	9
3.4.2.2	hash_value	9
3.4.2.3	to_string	9
3.5	marathon::State::Equal Class Reference	9
3.5.1	Detailed Description	9
3.6	marathon::State::Hash Class Reference	9
3.6.1	Detailed Description	9
3.7	marathon::chain::matching::JerrumSinclairVigoda04 Class Reference	10

3.7.1	Detailed Description	10
3.7.2	Member Function Documentation	10
3.7.2.1	computeNeighbours	10
3.7.2.2	computeWeights	10
3.8	marathon::chain::matching::JS89Path Class Reference	11
3.8.1	Detailed Description	11
3.9	marathon::chain::bipgraph::KannanPath Class Reference	11
3.9.1	Detailed Description	12
3.9.2	Member Function Documentation	12
3.9.2.1	construct	12
3.10	marathon::State::Less Class Reference	12
3.10.1	Detailed Description	12
3.11	marathon::MarkovChain Class Reference	12
3.11.1	Detailed Description	13
3.11.2	Constructor & Destructor Documentation	13
3.11.2.1	MarkovChain	13
3.11.3	Member Function Documentation	13
3.11.3.1	computeArbitraryState	13
3.11.3.2	computeNeighbours	14
3.11.3.3	computeWeights	15
3.11.3.4	getInstance	15
3.11.3.5	getName	15
3.11.3.6	randomize	15
3.11.3.7	randomWalk	15
3.12	marathon::PathConstructionScheme Class Reference	16
3.12.1	Detailed Description	16
3.12.2	Member Function Documentation	16
3.12.2.1	construct	16
3.13	marathon::rational Class Reference	16
3.13.1	Detailed Description	17
3.14	marathon::chain::matching::SparseBipartiteGraph Class Reference	17
3.14.1	Detailed Description	17
3.15	marathon::State Class Reference	18
3.15.1	Detailed Description	18
3.15.2	Member Function Documentation	18
3.15.2.1	compare_to	18
3.15.2.2	hash_value	18
3.15.2.3	to_string	19
3.15.3	Friends And Related Function Documentation	19
3.15.3.1	operator<<	19

3.15.3.2	operator<<	19
3.16	marathon::StateGraph Class Reference	19
3.16.1	Detailed Description	20
3.16.2	Constructor & Destructor Documentation	20
3.16.2.1	StateGraph	20
3.16.2.2	~StateGraph	20
3.16.3	Member Function Documentation	21
3.16.3.1	addArc	21
3.16.3.2	addArc	21
3.16.3.3	addLoopArc	21
3.16.3.4	addState	21
3.16.3.5	addTransitionProbability	21
3.16.3.6	clear	21
3.16.3.7	expand	21
3.16.3.8	expandState	22
3.16.3.9	getArc	22
3.16.3.10	getArc	22
3.16.3.11	getArcs	22
3.16.3.12	getInArcs	22
3.16.3.13	getMarkovChain	22
3.16.3.14	getMinWeight	22
3.16.3.15	getNumOutArcs	23
3.16.3.16	getNumStates	23
3.16.3.17	getNumTransitions	23
3.16.3.18	getOutArcs	23
3.16.3.19	getState	23
3.16.3.20	getStates	23
3.16.3.21	getTransitionProbability	23
3.16.3.22	getWeight	23
3.16.3.23	getWeights	23
3.16.3.24	getZ	23
3.16.3.25	indexOf	23
3.16.3.26	setTransitionProbability	23
3.16.3.27	setWeight	24
3.17	marathon::chain::bipgraph::SwitchChain Class Reference	24
3.17.1	Detailed Description	24
3.17.2	Member Function Documentation	25
3.17.2.1	computeArbitraryState	25
3.17.2.2	computeNeighbours	25
3.17.2.3	parseInstance	25

3.17.2.4	randomize	25
3.18	marathon::chain::bipgraph::SwitchChainBerger Class Reference	25
3.18.1	Detailed Description	26
3.18.2	Member Function Documentation	26
3.18.2.1	computeNeighbours	26
3.19	marathon::Transition Class Reference	26
3.19.1	Detailed Description	26
3.20	marathon::TransitionComparator Struct Reference	27
3.20.1	Detailed Description	27
3.21	marathon::TransitionMatrix< T > Class Template Reference	27
3.21.1	Detailed Description	28
3.21.2	Member Function Documentation	28
3.21.2.1	copy	28
3.21.2.2	generateSubTypeInstance	28
3.21.2.3	getData	28
3.21.2.4	getLeadDimension	28
3.21.2.5	getN	28
3.21.2.6	mult	28
3.21.2.7	pow	29
3.21.2.8	setEye	29
3.21.2.9	setZero	29
3.21.2.10	swap	29
3.21.2.11	to_string	29
3.21.2.12	totalVariationDistance	29
3.21.2.13	variationDistance	30
3.21.3	Friends And Related Function Documentation	30
3.21.3.1	operator<<	30
3.21.3.2	operator<<	30
3.22	marathon::tm::TransitionMatrixCBLAS< T > Class Template Reference	30
3.22.1	Detailed Description	31
3.22.2	Member Function Documentation	31
3.22.2.1	copy	31
3.22.2.2	generateSubTypeInstance	31
3.22.2.3	mult	31
3.22.2.4	setEye	31
3.22.2.5	setZero	32
3.22.2.6	to_string	32
3.22.2.7	totalVariationDistance	32
3.22.2.8	variationDistance	32
3.23	marathon::tm::TransitionMatrixCuBLAS< T > Class Template Reference	32

3.23.1 Detailed Description	33
3.23.2 Member Function Documentation	33
3.23.2.1 copy	33
3.23.2.2 generateSubTypeInstance	33
3.23.2.3 mult	33
3.23.2.4 setEye	34
3.23.2.5 setZero	34
3.23.2.6 to_string	34
3.23.2.7 totalVariationDistance	34
3.23.2.8 variationDistance	34
3.24 marathon::tm::TransitionMatrixCuBLASxt< T > Class Template Reference	35
3.24.1 Detailed Description	35
3.24.2 Member Function Documentation	35
3.24.2.1 mult	35
Index	36

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

marathon::chain::bipgraph::KannanPath::cycle_comparator	7
marathon::State::Equal	9
marathon::State::Hash	9
marathon::State::Less	12
marathon::MarkovChain	12
marathon::chain::bipgraph::SwitchChain	24
marathon::chain::bipgraph::SwitchChainBerger	25
marathon::chain::matching::Broder86	6
marathon::chain::matching::JerrumSinclairVigoda04	10
marathon::PathConstructionScheme	16
marathon::chain::bipgraph::KannanPath	11
marathon::chain::matching::JS89Path	11
marathon::rational	16
marathon::chain::matching::SparseBipartiteGraph	17
marathon::State	18
marathon::chain::bipgraph::DenseBipartiteGraph	8
marathon::chain::matching::BipartiteMatching	5
marathon::StateGraph	19
marathon::Transition	26
marathon::TransitionComparator	27
marathon::TransitionMatrix< T >	27
marathon::tm::TransitionMatrixCBLAS< T >	30
marathon::tm::TransitionMatrixCuBLASxt< T >	35
marathon::tm::TransitionMatrixCuBLAS< T >	32

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

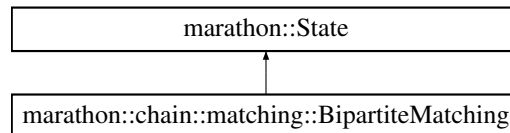
marathon::chain::matching::BipartiteMatching	5
marathon::chain::matching::Broder86	6
marathon::chain::bipgraph::KannanPath::cycle_comparator	7
marathon::chain::bipgraph::DenseBipartiteGraph	8
marathon::State::Equal	9
marathon::State::Hash	9
marathon::chain::matching::JerrumSinclairVigoda04	10
marathon::chain::matching::JS89Path	11
marathon::chain::bipgraph::KannanPath	11
marathon::State::Less	12
marathon::MarkovChain	12
marathon::PathConstructionScheme	16
marathon::rational	16
marathon::chain::matching::SparseBipartiteGraph	17
marathon::State	18
marathon::StateGraph	19
marathon::chain::bipgraph::SwitchChain	24
marathon::chain::bipgraph::SwitchChainBerger	25
marathon::Transition	26
marathon::TransitionComparator	27
marathon::TransitionMatrix< T >	27
marathon::tm::TransitionMatrixCBLAS< T >	30
marathon::tm::TransitionMatrixCuBLAS< T >	32
marathon::tm::TransitionMatrixCuBLASXt< T >	35

Chapter 3

Class Documentation

3.1 marathon::chain::matching::BipartiteMatching Class Reference

Inheritance diagram for marathon::chain::matching::BipartiteMatching:



Public Member Functions

- **BipartiteMatching** (const [BipartiteMatching](#) &s)
- **BipartiteMatching** (int n, int k, int unmatched[2], int *matching)
- void **addEdge** (int u, int v)
- void **removeEdge** (int u, int v)
- void **operator=** ([BipartiteMatching](#) const &s)
- bool **operator==** (const [BipartiteMatching](#) &s) const
- bool **operator<** (const [BipartiteMatching](#) &s) const
- size_t **hash_value** () const
- int **compare_to** (const [State](#) *) const
- std::string **to_string** () const
- bool **is_perfect** () const
- bool **is_near_perfect** () const

Public Attributes

- int **n**
- int **k**
- int **unmatched** [2]
- int * **mates**

3.1.1 Detailed Description

Definition at line 21 of file `BipartiteMatching.h`.

3.1.2 Member Function Documentation

3.1.2.1 `int marathon::chain::matching::BipartiteMatching::compare_to (const State * s) const` [virtual]

Compare this and s by structural properties.

If this < s : return -1. If this == s: return 0. If this > s : return 1.

Implements [marathon::State](#).

3.1.2.2 `size_t marathon::chain::matching::BipartiteMatching::hash_value () const` [virtual]

Virtual Hash Function for [State](#) Type.

Implements [marathon::State](#).

3.1.2.3 `std::string marathon::chain::matching::BipartiteMatching::to_string () const` [virtual]

Return a string representation of the state.

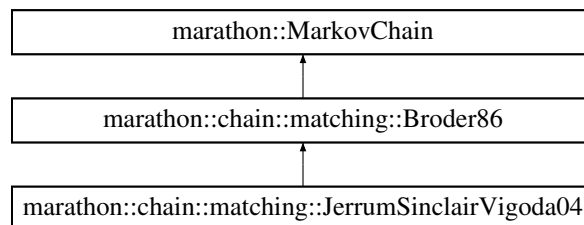
Implements [marathon::State](#).

The documentation for this class was generated from the following file:

- include/marathon/chain/matching/BipartiteMatching.h

3.2 marathon::chain::matching::Broder86 Class Reference

Inheritance diagram for `marathon::chain::matching::Broder86`:



Public Member Functions

- **Broder86** (const std::string &instance)
- virtual [State](#) * [computeArbitraryState](#) ()
- virtual void [computeNeighbours](#) (const [State](#) *s, std::vector< std::pair< [State](#) *, [rational](#) >> &neighbors) const

Protected Member Functions

- void [parseInstance](#) (const std::string &inst)

Protected Attributes

- [SparseBipartiteGraph](#) * **g** = nullptr

Friends

- class **JS89Path**

3.2.1 Detailed Description

Definition at line 21 of file Broder86.h.

3.2.2 Member Function Documentation

3.2.2.1 `virtual State* marathon::chain::matching::Broder86::computeArbitraryState () [virtual]`

Computes an arbitrary state and store it the state object s.

Returns

A pointer to a state object or nullptr if state space is empty.

Implements [marathon::MarkovChain](#).

3.2.2.2 `virtual void marathon::chain::matching::Broder86::computeNeighbours (const State * s, std::vector< std::pair< State *, rational >> & neighbors) const [virtual]`

Compute the set of adjacent states of s with corresponding proposal probability.

Parameters

<i>s</i>	A pointer to the state for which its neighbours are to be computed.
<i>neighbors</i>	A vector with pointers to adjacent state objects that and their proposal probabilities.

Implements [marathon::MarkovChain](#).

Reimplemented in [marathon::chain::matching::JerrumSinclairVigoda04](#).

3.2.2.3 `void marathon::chain::matching::Broder86::parseInstance (const std::string & inst) [protected]`

Instances have the form "110101011". Such a 0-1-String is interpreted as a biadjacency matrix of a bipartite graph, flattened to a single line. Thus, the input string above corresponds to the biadjacency matrix

1 1 0 1 0 1 0 1 1

which is the graph

u1 u2 u3 | \ \ \ | X X | / \ \ | v1 v2 v3

The documentation for this class was generated from the following file:

- include/marathon/chain/matching/Broder86.h

3.3 marathon::chain::bipgraph::KannanPath::cycle_comparator Struct Reference

Public Member Functions

- bool **operator()** (const std::vector< int > &c1, const std::vector< int > &c2)

3.3.1 Detailed Description

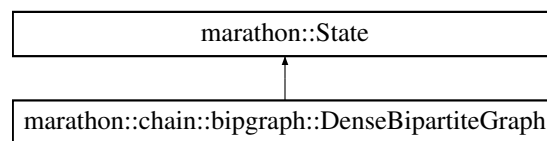
Definition at line 33 of file KannanCanPath.h.

The documentation for this struct was generated from the following file:

- include/marathon/chain/bipgraph/KannanCanPath.h

3.4 marathon::chain::bipgraph::DenseBipartiteGraph Class Reference

Inheritance diagram for marathon::chain::bipgraph::DenseBipartiteGraph:



Public Member Functions

- **DenseBipartiteGraph** (int nrows, int ncols, const bool *bits=nullptr)
- **DenseBipartiteGraph** (int nrows, int ncols, const std::string &str)
- **DenseBipartiteGraph** (const [DenseBipartiteGraph](#) &s)
- int **get_nrows** () const
- int **get_ncols** () const
- bool **has_edge** (int u, int v) const
- void **flip_edge** (int u, int v)
- void **set_edge** (int u, int v, bool)
- bool **is_switchable** (int u1, int u2, int v1, int v2) const
- void **switch_4_cycle** (int u1, int u2, int v1, int v2)
- void **get_row** (int u, boost::dynamic_bitset<> &row) const
- size_t **hash_value** () const
- int **compare_to** (const [State](#) *x) const
- std::string **to_string** () const
- void **operator=** (const [DenseBipartiteGraph](#) &s)
- bool **operator<** (const [DenseBipartiteGraph](#) &rhs) const
- bool **operator==** (const [DenseBipartiteGraph](#) &rhs) const

Static Public Member Functions

- static int **COORD_TRANSFORM** (const int x, const int y, const int ld)

Public Attributes

- int **nrows**
- int **ncols**
- boost::dynamic_bitset **M**

3.4.1 Detailed Description

Definition at line 21 of file DenseBipartiteGraph.h.

3.4.2 Member Function Documentation

3.4.2.1 `int marathon::chain::bipgraph::DenseBipartiteGraph::compare_to (const State * s) const` [virtual]

Compare this and s by structural properties.

If this<s : return -1. If this==s: return 0. If this>s : return 1.

Implements [marathon::State](#).

3.4.2.2 `size_t marathon::chain::bipgraph::DenseBipartiteGraph::hash_value () const` [virtual]

Virtual Hash Function for [State](#) Type.

Implements [marathon::State](#).

3.4.2.3 `std::string marathon::chain::bipgraph::DenseBipartiteGraph::to_string () const` [virtual]

Return a string representation of the state.

Implements [marathon::State](#).

The documentation for this class was generated from the following file:

- include/marathon/chain/bipgraph/DenseBipartiteGraph.h

3.5 marathon::State::Equal Class Reference

Public Member Functions

- `bool operator() (State *x1, State *x2) const`

3.5.1 Detailed Description

Definition at line 77 of file State.h.

The documentation for this class was generated from the following file:

- include/marathon/State.h

3.6 marathon::State::Hash Class Reference

```
#include <State.h>
```

Public Member Functions

- `size_t operator() (State *x) const`

3.6.1 Detailed Description

Wrapper Class for the use in std::unordered_maps.

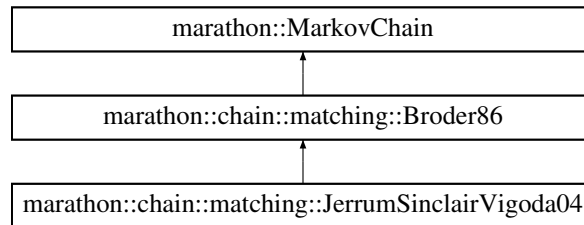
Definition at line 66 of file State.h.

The documentation for this class was generated from the following file:

- `include/marathon/State.h`

3.7 `marathon::chain::matching::JerrumSinclairVigoda04` Class Reference

Inheritance diagram for `marathon::chain::matching::JerrumSinclairVigoda04`:



Public Member Functions

- **JerrumSinclairVigoda04** (`const std::string &input`)
- `void computeNeighbours` (`const State *s`, `std::vector< std::pair< State *, rational >> &neighbors`) `const`
- `void computeWeights` (`const std::vector< const State * > &states`, `std::vector< rational > &weights`)

Protected Member Functions

- `rational getWeight` (`const State *s`) `const`

Protected Attributes

- `uint num_perfect_matching`
- `uint * num_near_perfect_matching`

3.7.1 Detailed Description

Definition at line 19 of file JSV04.h.

3.7.2 Member Function Documentation

3.7.2.1 `void marathon::chain::matching::JerrumSinclairVigoda04::computeNeighbours (const State * s, std::vector< std::pair< State *, rational >> & neighbors) const` `[virtual]`

Compute the set of adjacent states of `s` with corresponding proposal probability.

Parameters

<code>s</code>	A pointer to the state for which its neighbours are to be computed.
<code>neighbors</code>	A vector with pointers to adjacent state objects that and their proposal probabilities.

Reimplemented from `marathon::chain::matching::Broder86`.

3.7.2.2 `void marathon::chain::matching::JerrumSinclairVigoda04::computeWeights (const std::vector< const State * > & states, std::vector< rational > & weights)` `[virtual]`

Computes weights for each state.

Parameters

<i>states</i>	The Vector of states.
<i>weights</i>	The Vector of weights. After calling the method, this vector must have the same size as states and is filled with rationals.

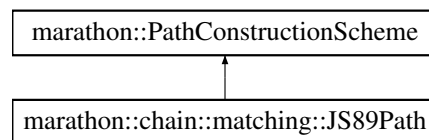
Reimplemented from [marathon::MarkovChain](#).

The documentation for this class was generated from the following file:

- include/marathon/chain/matching/JSV04.h

3.8 marathon::chain::matching::JS89Path Class Reference

Inheritance diagram for marathon::chain::matching::JS89Path:



Additional Inherited Members

3.8.1 Detailed Description

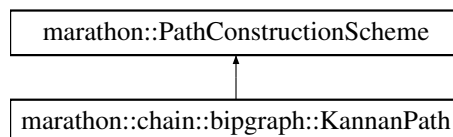
Definition at line 18 of file JS89CanPath.h.

The documentation for this class was generated from the following file:

- include/marathon/chain/matching/JS89CanPath.h

3.9 marathon::chain::bipgraph::KannanPath Class Reference

Inheritance diagram for marathon::chain::bipgraph::KannanPath:



Classes

- struct [cycle_comparator](#)

Public Member Functions

- virtual void [construct](#) (const [StateGraph](#) *sg, const int s, const int t, std::list< int > &path) const

Protected Member Functions

- `int next_red_edge (int col, bool *red_edges, int m, int n) const`
- `int next_blue_edge (int row, bool *blue_edges, int m, int n) const`
- `void trace_cycle (bool *blue_edges, bool *red_edges, int m, int n, int i, int j, std::vector< int > &cycle) const`
- `void splice_cycle (std::vector< int > cycle, std::list< std::vector< int > > &cycles, const int m, const int n) const`
- `void cycle_decomposition (const DenseBipartiteGraph &x, const DenseBipartiteGraph &y, std::list< std::vector< int > > &cycles) const`

3.9.1 Detailed Description

Definition at line 18 of file KannanCanPath.h.

3.9.2 Member Function Documentation

- 3.9.2.1 `virtual void marathon::chain::bipgraph::KannanPath::construct (const StateGraph * sg, const int s, const int t, std::list< int > & path) const` [virtual]

Construct a path between states s and t in Graph sg.

Parameters

<i>sg</i>	A pointer to a state graph object at which the path is embedded.
<i>s</i>	The index of the paths start state.
<i>t</i>	The index of the paths final state.
<i>path</i>	A list of state indices that represent the path.

Implements [marathon::PathConstructionScheme](#).

The documentation for this class was generated from the following file:

- `include/marathon/chain/bipgraph/KannanCanPath.h`

3.10 marathon::State::Less Class Reference

Public Member Functions

- `bool operator() (State *x1, State *x2) const`

3.10.1 Detailed Description

Definition at line 88 of file State.h.

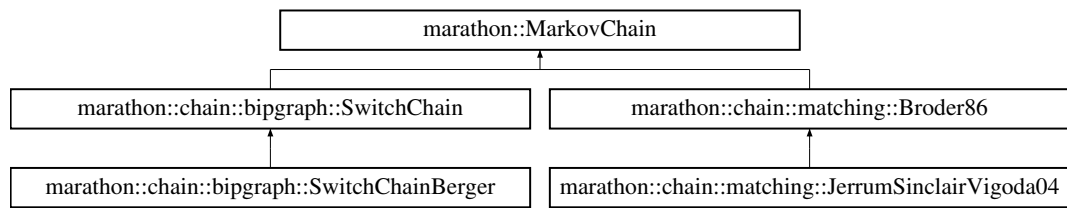
The documentation for this class was generated from the following file:

- `include/marathon/State.h`

3.11 marathon::MarkovChain Class Reference

```
#include <MarkovChain.h>
```

Inheritance diagram for `marathon::MarkovChain`:



Public Member Functions

- [MarkovChain](#) (const std::string &s, int seed=0)
- const std::string & [getInstance](#) () const
- std::string [getName](#) () const
- virtual [State](#) * [computeArbitraryState](#) ()=0
- virtual void [computeNeighbours](#) (const [State](#) *s, std::vector< std::pair< [State](#) *, [rational](#) >> &neighbors) const =0
- virtual void [computeWeights](#) (const std::vector< const [State](#) * > &states, std::vector< [rational](#) > &weights)
- [State](#) * [randomWalk](#) (const int t)
- virtual void [randomize](#) ([State](#) *s) const

Protected Attributes

- std::string **instance**

3.11.1 Detailed Description

Virtual Markov chain base class.

Definition at line 27 of file MarkovChain.h.

3.11.2 Constructor & Destructor Documentation

3.11.2.1 marathon::MarkovChain::MarkovChain (const std::string & s, int seed = 0)

Create A Markov Chain Object for the input s.

Parameters

s	input string of the markov chain. false if is to be constructed by Omega ² procedure.
---	--

3.11.3 Member Function Documentation

3.11.3.1 virtual [State](#)* marathon::MarkovChain::computeArbitraryState () [pure virtual]

Computes an arbitrary state and store it the state object s.

Returns

A pointer to a state object or nullptr if state space is empty.

Implemented in [marathon::chain::matching::Broder86](#), and [marathon::chain::bipgraph::SwitchChain](#).

3.11.3.2 `virtual void marathon::MarkovChain::computeNeighbours (const State * s, std::vector< std::pair< State *, rational >> & neighbors) const [pure virtual]`

Compute the set of adjacent states of *s* with corresponding proposal probability.

Parameters

<i>s</i>	A pointer to the state for which its neighbours are to be computed.
<i>neighbors</i>	A vector with pointers to adjacent state objects that and their proposal probabilities.

Implemented in [marathon::chain::matching::Broder86](#), [marathon::chain::bipgraph::SwitchChain](#), [marathon::chain::matching::JerrumSinclairVigoda04](#), and [marathon::chain::bipgraph::SwitchChainBerger](#).

3.11.3.3 `virtual void marathon::MarkovChain::computeWeights (const std::vector< const State * > & states, std::vector< rational > & weights) [virtual]`

Computes weights for each state.

Parameters

<i>states</i>	The Vector of states.
<i>weights</i>	The Vector of weights. After calling the method, this vector must have the same size as states and is filled with rationals.

Reimplemented in [marathon::chain::matching::JerrumSinclairVigoda04](#).

3.11.3.4 `const std::string& marathon::MarkovChain::getInstance () const`

Returns

A reference to the string instance.

3.11.3.5 `std::string marathon::MarkovChain::getName () const`

Return a human readable name (identifier) of the Markov chain.

3.11.3.6 `virtual void marathon::MarkovChain::randomize (State * s) const [virtual]`

Apply a random transition to the state. Used to simulate a random walk.

Parameters

<i>s</i>	A pointer to a state, which is randomly modified by the method.
----------	---

Reimplemented in [marathon::chain::bipgraph::SwitchChain](#).

3.11.3.7 `State* marathon::MarkovChain::randomWalk (const int t) [inline]`

Apply a random walk and return the current state at the end of the walk.

Parameters

<i>t</i>	The number of steps in the walk.
----------	----------------------------------

Returns

A state, randomly selected from the probability distribution $p^{(t)}_s$, where s is the state that is constructed via the `computeArbitraryState` method.

Definition at line 93 of file `MarkovChain.h`.

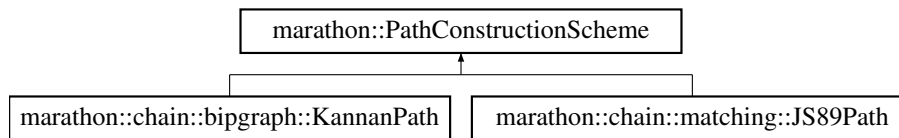
The documentation for this class was generated from the following file:

- `include/marathon/MarkovChain.h`

3.12 marathon::PathConstructionScheme Class Reference

```
#include <PathConstructionScheme.h>
```

Inheritance diagram for marathon::PathConstructionScheme:



Public Member Functions

- virtual void [construct](#) (const [StateGraph](#) *sg, const int s, const int t, std::list< int > &path) const =0

3.12.1 Detailed Description

A virtual base class for construction schemes of Canonical Paths.

Definition at line 18 of file PathConstructionScheme.h.

3.12.2 Member Function Documentation

3.12.2.1 virtual void marathon::PathConstructionScheme::construct (const [StateGraph](#) * sg, const int s, const int t, std::list< int > & path) const [pure virtual]

Construct a path between states s and t in Graph sg.

Parameters

<i>sg</i>	A pointer to a state graph object at which the path is embedded.
<i>s</i>	The index of the paths start state.
<i>t</i>	The index of the paths final state.
<i>path</i>	A list of state indices that represent the path.

Implemented in [marathon::chain::bipgraph::KannanPath](#).

The documentation for this class was generated from the following file:

- include/marathon/PathConstructionScheme.h

3.13 marathon::rational Class Reference

```
#include <Rational.h>
```

Public Member Functions

- **rational** (const [rational](#) &o)
- **rational** (boost::multiprecision::cpp_rational r)
- **rational** (int n)
- **rational** (int num, int denom)
- void **operator=** (const [rational](#) &o)
- bool **operator==** (const [rational](#) &o) const

- bool **operator!=** (const [rational](#) &o) const
- void **operator+=** (const [rational](#) &o)
- void **operator-=** (const [rational](#) &o)
- void **operator*=** (const [rational](#) &o)
- void **operator/=** (const [rational](#) &o)
- [rational](#) **operator*** (const [rational](#) &o) const
- [rational](#) **operator-** (const [rational](#) &o) const
- [rational](#) **operator+** (const [rational](#) &o) const
- [rational](#) **operator/** (const [rational](#) &o) const
- bool **operator<** (const [rational](#) &o) const
- bool **operator>** (const [rational](#) &o) const
- void **stream_to** (std::ostream &os) const
- template<typename T >
T **convert_to** () const

3.13.1 Detailed Description

just a wrapper around boost rational data type

Definition at line 19 of file Rational.h.

The documentation for this class was generated from the following file:

- include/marathon/Rational.h

3.14 marathon::chain::matching::SparseBipartiteGraph Class Reference

Public Member Functions

- **SparseBipartiteGraph** (const [SparseBipartiteGraph](#) &b)
- **SparseBipartiteGraph** (size_t n)
- **SparseBipartiteGraph** (std::string hash)
- unsigned int **getNumberOfNodes** () const
- unsigned int **getNumberOfEdges** () const
- void **getEdges** (edgelist &edges) const
- void **addEdge** (int u, int v)
- bool **hasEdge** (int u, int v) const
- void **removeEdge** (int u, int v)
- void **getNeighbors** (int v, std::vector< int > &neighbors) const
- void **cardmax_matching** (std::vector< int > &mates) const
- std::string **toString** () const
- void **convert_to_bitset** (boost::dynamic_bitset<> &) const

Friends

- std::ostream & **operator<<** (std::ostream &os, const [SparseBipartiteGraph](#) &bip)

3.14.1 Detailed Description

Definition at line 26 of file SparseBipartiteGraph.h.

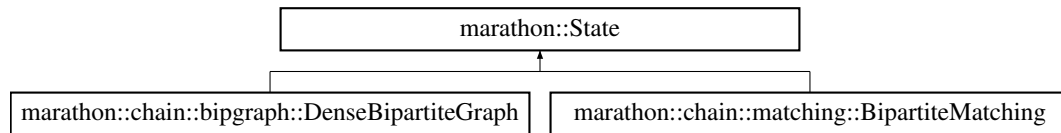
The documentation for this class was generated from the following file:

- include/marathon/chain/matching/SparseBipartiteGraph.h

3.15 marathon::State Class Reference

```
#include <State.h>
```

Inheritance diagram for marathon::State:



Classes

- class [Equal](#)
- class [Hash](#)
- class [Less](#)

Public Member Functions

- virtual size_t [hash_value](#) () const =0
- virtual int [compare_to](#) (const [State](#) *s) const =0
- virtual std::string [to_string](#) () const =0

Friends

- std::ostream & [operator<<](#) (std::ostream &out, const [State](#) &s)
- std::ostream & [operator<<](#) (std::ostream &out, const [State](#) *s)

3.15.1 Detailed Description

Abstract Base Class for States.

Definition at line 20 of file State.h.

3.15.2 Member Function Documentation

3.15.2.1 virtual int marathon::State::compare_to (const [State](#) * s) const [pure virtual]

Compare this and s by structural properties.

If this<s : return -1. If this==s: return 0. If this>s : return 1.

Implemented in [marathon::chain::bipgraph::DenseBipartiteGraph](#), and [marathon::chain::matching::BipartiteMatching](#).

3.15.2.2 virtual size_t marathon::State::hash_value () const [pure virtual]

Virtual [Hash](#) Function for [State](#) Type.

Implemented in [marathon::chain::bipgraph::DenseBipartiteGraph](#), and [marathon::chain::matching::BipartiteMatching](#).

3.15.2.3 `virtual std::string marathon::State::to_string () const [pure virtual]`

Return a string representation of the state.

Implemented in [marathon::chain::bipgraph::DenseBipartiteGraph](#), and [marathon::chain::matching::Bipartite-Matching](#).

3.15.3 Friends And Related Function Documentation

3.15.3.1 `std::ostream& operator<< (std::ostream & out, const State & s) [friend]`

To output into streams.

Definition at line 50 of file State.h.

3.15.3.2 `std::ostream& operator<< (std::ostream & out, const State * s) [friend]`

To output into streams.

Definition at line 58 of file State.h.

The documentation for this class was generated from the following file:

- include/marathon/State.h

3.16 marathon::StateGraph Class Reference

```
#include <StateGraph.h>
```

Public Member Functions

- [StateGraph](#) ([MarkovChain](#) *mc, const int limit=INT_MAX)
- virtual [~StateGraph](#) ()
- void [expand](#) (const int limit=INT_MAX, const bool verbose=false)
- [MarkovChain](#) * [getMarkovChain](#) () const
- int [addLoopArc](#) (const int u, const [rational](#) &p)
- int [addArc](#) (const int u, const int v, const [rational](#) &p)
- int [addArc](#) ([Transition](#) *t)
- [Transition](#) * [getArc](#) (int u, int v) const
- size_t [getNumStates](#) () const
- size_t [getNumTransitions](#) () const
- [rational](#) [getTransitionProbability](#) (int u, int v) const
- void [setTransitionProbability](#) (int u, int v, [rational](#) p)
- void [addTransitionProbability](#) (int u, int v, [rational](#) p)
- void [setWeight](#) (const int i, const [rational](#) p)
- [rational](#) [getWeight](#) (const int i) const
- [rational](#) [getMinWeight](#) () const
- [rational](#) [getZ](#) () const
- const std::vector< [rational](#) > & [getWeights](#) () const
- const std::vector< [Transition](#) * > & [getOutArcs](#) (int v) const
- const std::vector< [Transition](#) * > & [getInArcs](#) (int v) const
- const std::vector< [Transition](#) * > & [getArcs](#) () const
- [Transition](#) * [getArc](#) (const int i) const
- int [getNumOutArcs](#) (int v) const

- virtual void `clear` ()
- int `addState` (State *s)
- const State * `getState` (int i) const
- const std::vector< const State * > & `getStates` () const
- int `indexOf` (State *s) const

Protected Member Functions

- void `expandState` (const int i, const int limit, const int lastStop, const bool verbose)

Protected Attributes

- MarkovChain * `mc`
- std::vector< const State * > `states`
- std::vector< rational > `weights`
- std::unordered_map< State *, int, State::Hash, State::Equal > `indices`
- std::vector< Transition * > `arcs`
- std::vector< std::vector< Transition * > > `outArcs`
- std::vector< std::vector< Transition * > > `inArcs`
- int `nextIndex` = 0
- std::set< int > `reexpand`

3.16.1 Detailed Description

`State` Graph representation. A `State` Graph is a directed, weighted graph that represents a instance of a Markov Chain for a certain input instance.

Definition at line 32 of file `StateGraph.h`.

3.16.2 Constructor & Destructor Documentation

3.16.2.1 `marathon::StateGraph::StateGraph (MarkovChain * mc, const int limit = INT_MAX)`

Standard Constructor. Creates an empty `State` Graph.

Parameters

<i>mc</i>	A pointer to the Markov Chain Object that defines transition rules, etc.
<i>limit</i>	A limit on the number of states of the graph. The graph can later on be expanded by the <code>expand()</code> method.

3.16.2.2 `virtual marathon::StateGraph::~~StateGraph () [virtual]`

Standard Destructor. Remove everything.

3.16.3 Member Function Documentation

3.16.3.1 `int marathon::StateGraph::addArc (const int u, const int v, const rational & p)`

Adds a transition arc to the graph. Precondition: The state graph does not already contain an arc between state *u* and state *v*.

Returns

Returns the index of the new transition.

3.16.3.2 `int marathon::StateGraph::addArc (Transition * t)`

Adds a transition arc to the graph. Precondition: The state graph does not already contain an arc between state *t.u* and state *t.v*.

Returns

Returns the index of the new transition.

3.16.3.3 `int marathon::StateGraph::addLoopArc (const int u, const rational & p)`

Add a new transition to the state graph that represents a loop.

3.16.3.4 `int marathon::StateGraph::addState (State * s)`

Add a new [State](#) to the state graph.

Parameters

<i>s</i>	The State to insert.
----------	--------------------------------------

Returns

The index of the state after insertion.

3.16.3.5 `void marathon::StateGraph::addTransitionProbability (int u, int v, rational p)`

Increases $P(u,v)$ by an amount of *p*.

3.16.3.6 `virtual void marathon::StateGraph::clear ()` [virtual]

Removes all States and Transitions and re-initializes the state graph.

3.16.3.7 `void marathon::StateGraph::expand (const int limit = INT_MAX, const bool verbose = false)`

Expands an existing state graph to a given maximum of states.

Parameters

<i>limit</i>	The maximal number of states after the expansion
<i>verbose</i>	Enables or disables additional debug output

Returns

the number of states that has been added during the expansion

3.16.3.8 `void marathon::StateGraph::expandState (const int i, const int limit, const int lastStop, const bool verbose)`
`[protected]`

This is a private method that is called during state graph expansion. It computes all neighbouring states of state *s* and insert them into the state graph repectively into the leftover structures that store the states and arcs for next `expandStateGraph()`.

Parameters

<i>i</i>	The index of the state that is to be expanded.
<i>limit</i>	The maximal number of states.
<i>lastStop</i>	The size of the state graph when this expansion has been triggered.
<i>verbose</i>	If true, additional debug information is printed.
<i>True,if</i>	all adjacent states could be inserted in the state graph.

3.16.3.9 `Transition* marathon::StateGraph::getArc (int u, int v) const`

Return a pointer to the arc that connects *u* with *v* or nullptr, if no such arc exists.

3.16.3.10 `Transition* marathon::StateGraph::getArc (const int i) const`

Return a pointer to arc with index *i*.

Parameters

<i>i</i>	The index of the arc.
----------	-----------------------

Returns

A pointer to the *i*'th transition.

3.16.3.11 `const std::vector<Transition*>& marathon::StateGraph::getArcs () const`

Returns a reference to the vector of all arcs in the state graph.

3.16.3.12 `const std::vector<Transition*>& marathon::StateGraph::getInArcs (int v) const`

Returns a reference to the ingoing arcs of state *v*.

3.16.3.13 `MarkovChain* marathon::StateGraph::getMarkovChain () const`

Return a pointer to the corresponding Markov Chain Object.

3.16.3.14 `rational marathon::StateGraph::getMinWeight () const`

Return the minimal weight of a state.

3.16.3.15 `int marathon::StateGraph::getNumOutArcs (int v) const`

Returns the number of adjacent states of state[v]

3.16.3.16 `size_t marathon::StateGraph::getNumStates () const`

Returns the number of states of the state graph

3.16.3.17 `size_t marathon::StateGraph::getNumTransitions () const`

Returns the number of Transitions/Arcs of the state graph

3.16.3.18 `const std::vector<Transition*>& marathon::StateGraph::getOutArcs (int v) const`

Returns a reference to the outgoing arcs of state *v*.

3.16.3.19 `const State* marathon::StateGraph::getState (int i) const`

Returns a reference to the [State](#) with index *i*.

3.16.3.20 `const std::vector<const State*>& marathon::StateGraph::getStates () const`

Returns a reference to a vector of States.

3.16.3.21 `rational marathon::StateGraph::getTransitionProbability (int u, int v) const`

Returns the transition probability P_{uv} for going from states[u] to states[v]

3.16.3.22 `rational marathon::StateGraph::getWeight (const int i) const`

Return the weight of state *i*.

3.16.3.23 `const std::vector<rational>& marathon::StateGraph::getWeights () const`

Return a vector of weights for each state.

3.16.3.24 `rational marathon::StateGraph::getZ () const`

Return the sum of all weights.

3.16.3.25 `int marathon::StateGraph::indexOf (State * s) const`

Returns the index of a state or -1 if the state graph does not contain this state.

3.16.3.26 `void marathon::StateGraph::setTransitionProbability (int u, int v, rational p)`

Set $P(u,v)$ to *p*

3.16.3.27 void marathon::StateGraph::setWeight (const int *i*, const rational *p*)

Sets the weight of state[*i*] to *p*.

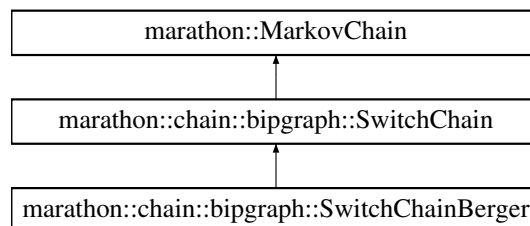
The documentation for this class was generated from the following file:

- include/marathon/StateGraph.h

3.17 marathon::chain::bipgraph::SwitchChain Class Reference

```
#include <SwitchChain.h>
```

Inheritance diagram for marathon::chain::bipgraph::SwitchChain:



Public Member Functions

- **SwitchChain** (const std::string &inst, int seed=0)
- virtual [State](#) * [computeArbitraryState](#) ()
- virtual void [computeNeighbours](#) (const [State](#) *s, std::vector< std::pair< [State](#) *, [rational](#) >> &neighbors) const
- virtual void [randomize](#) ([State](#) *s) const

Protected Member Functions

- virtual void [parseInstance](#) (const std::string &line)

Protected Attributes

- std::vector< int > **u**
- std::vector< int > **v**
- int **sum**

Friends

- class **KannanPath**

3.17.1 Detailed Description

Implements the Markov chain defined by Kannan et al.

Definition at line 24 of file SwitchChain.h.

3.17.2 Member Function Documentation

3.17.2.1 virtual State* marathon::chain::bipgraph::SwitchChain::computeArbitraryState () [virtual]

Computes an arbitrary state and store it the state object s.

Returns

A pointer to a state object or nullptr if state space is empty.

Implements [marathon::MarkovChain](#).

3.17.2.2 virtual void marathon::chain::bipgraph::SwitchChain::computeNeighbours (const State * s, std::vector< std::pair< State *, rational >> & neighbors) const [virtual]

Compute the set of adjacent states of s with corresponding proposal probability.

Parameters

<i>s</i>	A pointer to the state for which its neighbours are to be computed.
<i>neighbors</i>	A vector with pointers to adjacent state objects that and their proposal probabilities.

Implements [marathon::MarkovChain](#).

Reimplemented in [marathon::chain::bipgraph::SwitchChainBerger](#).

3.17.2.3 virtual void marathon::chain::bipgraph::SwitchChain::parseInstance (const std::string & line) [protected], [virtual]

Instances have the form "2,2,2;1,2,1,2". The semicolon separates two degree sequences of both bipartition sets.

3.17.2.4 virtual void marathon::chain::bipgraph::SwitchChain::randomize (State * s) const [virtual]

Randomize the state s by applying a single transition.

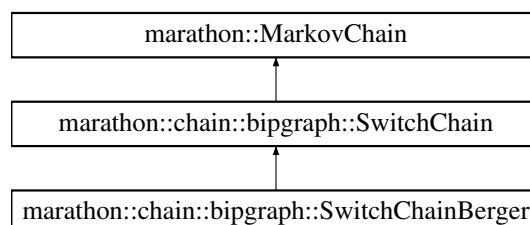
Reimplemented from [marathon::MarkovChain](#).

The documentation for this class was generated from the following file:

- include/marathon/chain/bipgraph/SwitchChain.h

3.18 marathon::chain::bipgraph::SwitchChainBerger Class Reference

Inheritance diagram for marathon::chain::bipgraph::SwitchChainBerger:



Public Member Functions

- **SwitchChainBerger** (const std::string &input)

Protected Member Functions

- virtual void [computeNeighbours](#) (const [State](#) *s, std::vector< std::pair< [State](#) *, [rational](#) >> &neighbors) const

Additional Inherited Members

3.18.1 Detailed Description

Definition at line 17 of file SwitchChainBerger.h.

3.18.2 Member Function Documentation

3.18.2.1 virtual void marathon::chain::bipgraph::SwitchChainBerger::computeNeighbours (const [State](#) * s, std::vector< std::pair< [State](#) *, [rational](#) >> & *neighbors*) const [protected], [virtual]

Compute the set of adjacent states of s with corresponding proposal probability.

Parameters

<i>s</i>	A pointer to the state for which its neighbours are to be computed.
<i>neighbors</i>	A vector with pointers to adjacent state objects that and their proposal probabilities.

Reimplemented from [marathon::chain::bipgraph::SwitchChain](#).

The documentation for this class was generated from the following file:

- include/marathon/chain/bipgraph/SwitchChainBerger.h

3.19 marathon::Transition Class Reference

```
#include <Transition.h>
```

Public Member Functions

- **Transition** (uint u, uint v, [rational](#) p)

Public Attributes

- uint **u**
- uint **v**
- [rational](#) **p**

3.19.1 Detailed Description

[Transition](#) Arc Representation of [State](#) Graph

Definition at line 21 of file Transition.h.

The documentation for this class was generated from the following file:

- include/marathon/Transition.h

3.20 marathon::TransitionComparator Struct Reference

Public Member Functions

- bool **operator()** (const [Transition](#) &a, const [Transition](#) &b)

3.20.1 Detailed Description

Definition at line 33 of file [Transition.h](#).

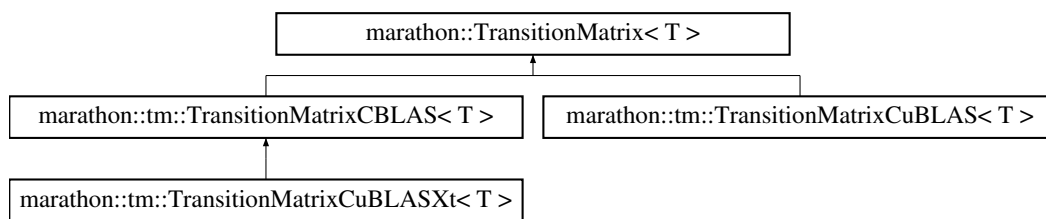
The documentation for this struct was generated from the following file:

- [include/marathon/Transition.h](#)

3.21 marathon::TransitionMatrix< T > Class Template Reference

```
#include <TransitionMatrix.h>
```

Inheritance diagram for `marathon::TransitionMatrix< T >`:



Public Member Functions

- `size_t` [getN](#) () const
- `size_t` [getLeadDimension](#) () const
- `T *` [getData](#) () const
- virtual void [copy](#) (const [TransitionMatrix](#)< T > *P)=0
- virtual void [setEye](#) ()=0
- virtual void [setZero](#) ()=0
- virtual void [mult](#) (const [TransitionMatrix](#)< T > *A, const [TransitionMatrix](#)< T > *B)=0
- void [pow](#) (const [TransitionMatrix](#)< T > *P, const int k)
- virtual std::string [to_string](#) () const =0
- virtual void [variationDistance](#) (const T *pi, T *dist) const =0
- virtual T [totalVariationDistance](#) (const T *pi) const =0
- void [swap](#) ([TransitionMatrix](#)< T > *P)

Protected Member Functions

- virtual [TransitionMatrix](#)< T > * [generateSubTypeInstance](#) (const int n)=0

Protected Attributes

- `size_t` **n**
- `size_t` **ld**
- `T *` **data**

Friends

- `std::ostream & operator<< (std::ostream &out, const TransitionMatrix< T > &s)`
- `std::ostream & operator<< (std::ostream &out, const TransitionMatrix< T > *s)`

3.21.1 Detailed Description

`template<typename T = double>class marathon::TransitionMatrix< T >`

Virtual Base Class for [Transition](#) Matrix.

Definition at line 19 of file TransitionMatrix.h.

3.21.2 Member Function Documentation

3.21.2.1 `template<typename T = double> virtual void marathon::TransitionMatrix< T >::copy (const TransitionMatrix< T > * P) [pure virtual]`

Copy the content of matrix P to this.

Implemented in [marathon::tm::TransitionMatrixCuBLAS< T >](#), and [marathon::tm::TransitionMatrixCBLAS< T >](#).

3.21.2.2 `template<typename T = double> virtual TransitionMatrix<T>* marathon::TransitionMatrix< T >::generateSubTypeInstance (const int n) [protected],[pure virtual]`

Return a pointer to a [Transition](#) matrix of an appropriate subtype.

Implemented in [marathon::tm::TransitionMatrixCuBLAS< T >](#), and [marathon::tm::TransitionMatrixCBLAS< T >](#).

3.21.2.3 `template<typename T = double> T* marathon::TransitionMatrix< T >::getData () const [inline]`

Return a pointer to the data.

Definition at line 55 of file TransitionMatrix.h.

3.21.2.4 `template<typename T = double> size_t marathon::TransitionMatrix< T >::getLeadDimension () const [inline]`

Return lead dimension of the matrix.

Definition at line 48 of file TransitionMatrix.h.

3.21.2.5 `template<typename T = double> size_t marathon::TransitionMatrix< T >::getN () const [inline]`

Return size of the matrix.

Definition at line 41 of file TransitionMatrix.h.

3.21.2.6 `template<typename T = double> virtual void marathon::TransitionMatrix< T >::mult (const TransitionMatrix< T > * A, const TransitionMatrix< T > * B) [pure virtual]`

Multiply A with B and write the result to this.

Parameters

<i>A</i>	A pointer to matrix A. Will not be changed.
<i>B</i>	A pointer to matrix B. Will not be changed.

Implemented in [marathon::tm::TransitionMatrixCuBLAS< T >](#), [marathon::tm::TransitionMatrixCBLAS< T >](#), and [marathon::tm::TransitionMatrixCuBLASxt< T >](#).

3.21.2.7 `template<typename T = double> void marathon::TransitionMatrix< T >::pow (const TransitionMatrix< T > * P, const int k) [inline]`

Compute P^k and write the result to this.

Parameters

<i>P</i>	A pointer to a Transition Matrix.
<i>k</i>	Exponent.

Definition at line 87 of file TransitionMatrix.h.

3.21.2.8 `template<typename T = double> virtual void marathon::TransitionMatrix< T >::setEye () [pure virtual]`

Overwrite the current matrix with unity matrix.

Implemented in [marathon::tm::TransitionMatrixCuBLAS< T >](#), and [marathon::tm::TransitionMatrixCBLAS< T >](#).

3.21.2.9 `template<typename T = double> virtual void marathon::TransitionMatrix< T >::setZero () [pure virtual]`

Overwrite the current matrix with zeroes.

Implemented in [marathon::tm::TransitionMatrixCuBLAS< T >](#), and [marathon::tm::TransitionMatrixCBLAS< T >](#).

3.21.2.10 `template<typename T = double> void marathon::TransitionMatrix< T >::swap (TransitionMatrix< T > * P) [inline]`

Swap the content of the Matrix with another matrix.

Definition at line 162 of file TransitionMatrix.h.

3.21.2.11 `template<typename T = double> virtual std::string marathon::TransitionMatrix< T >::to_string () const [pure virtual]`

Return a string that represents the matrix.

Implemented in [marathon::tm::TransitionMatrixCuBLAS< T >](#), and [marathon::tm::TransitionMatrixCBLAS< T >](#).

3.21.2.12 `template<typename T = double> virtual T marathon::TransitionMatrix< T >::totalVariationDistance (const T * pi) const [pure virtual]`

Compute the total variation distance to the distribution.

Parameters

<i>pi</i>	A probability distribution.
-----------	-----------------------------

Implemented in [marathon::tm::TransitionMatrixCuBLAS< T >](#), and [marathon::tm::TransitionMatrixCBLAS< T >](#).

3.21.2.13 `template<typename T = double> virtual void marathon::TransitionMatrix< T >::variationDistance (const T *
pi, T * dist) const [pure virtual]`

Compute the variation distance of each state to the distribution pi

Parameters

<i>pi</i>	A pointer to a probability distribution.
<i>dist</i>	Out parameter.

Implemented in [marathon::tm::TransitionMatrixCuBLAS< T >](#), and [marathon::tm::TransitionMatrixCBLAS< T >](#).

3.21.3 Friends And Related Function Documentation

3.21.3.1 `template<typename T = double> std::ostream& operator<< (std::ostream & out, const TransitionMatrix< T >
& s) [friend]`

To output into streams.

Definition at line 171 of file TransitionMatrix.h.

3.21.3.2 `template<typename T = double> std::ostream& operator<< (std::ostream & out, const TransitionMatrix< T >
* s) [friend]`

To output into streams.

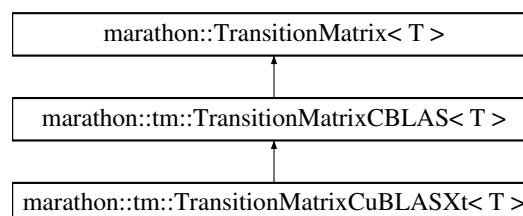
Definition at line 180 of file TransitionMatrix.h.

The documentation for this class was generated from the following file:

- include/marathon/TransitionMatrix.h

3.22 marathon::tm::TransitionMatrixCBLAS< T > Class Template Reference

Inheritance diagram for [marathon::tm::TransitionMatrixCBLAS< T >](#):



Public Member Functions

- **TransitionMatrixCBLAS** (const int n)
- **TransitionMatrixCBLAS** (const [StateGraph](#) *sg)
- virtual void [setEye](#) ()
- virtual void [setZero](#) ()

- virtual std::string [to_string](#) () const
- virtual void [mult](#) (const [TransitionMatrix](#)< T > *A, const [TransitionMatrix](#)< T > *B)
- virtual void [variationDistance](#) (const T *pi, T *dist) const
- virtual T [totalVariationDistance](#) (const T *pi) const

Protected Member Functions

- virtual void [copy](#) (const [TransitionMatrix](#)< T > *P)
- virtual [TransitionMatrix](#)< T > * [generateSubTypeInstance](#) (const int n)

Additional Inherited Members

3.22.1 Detailed Description

template<typename T>class marathon::tm::TransitionMatrixCBLAS< T >

Definition at line 17 of file TransitionMatrixCBLAS.h.

3.22.2 Member Function Documentation

3.22.2.1 template<typename T> virtual void marathon::tm::TransitionMatrixCBLAS< T >::copy (const [TransitionMatrix](#)< T > * P) [inline], [protected], [virtual]

Copy the content of matrix P to this.

Implements [marathon::TransitionMatrix< T >](#).

Definition at line 24 of file TransitionMatrixCBLAS.h.

3.22.2.2 template<typename T> virtual [TransitionMatrix](#)<T>* marathon::tm::TransitionMatrixCBLAS< T >::generateSubTypeInstance (const int n) [inline], [protected], [virtual]

Return a pointer to a [Transition](#) matrix of subtype instance.

Implements [marathon::TransitionMatrix< T >](#).

Definition at line 34 of file TransitionMatrixCBLAS.h.

3.22.2.3 template<typename T> virtual void marathon::tm::TransitionMatrixCBLAS< T >::mult (const [TransitionMatrix](#)< T > * A, const [TransitionMatrix](#)< T > * B) [virtual]

Multiply A with B and write the result to this.

Parameters

<i>A</i>	A pointer to matrix A. Will not be changed.
<i>B</i>	A pointer to matrix B. Will not be changed.

Implements [marathon::TransitionMatrix< T >](#).

Reimplemented in [marathon::tm::TransitionMatrixCuBLASxt< T >](#).

3.22.2.4 template<typename T> virtual void marathon::tm::TransitionMatrixCBLAS< T >::setEye () [inline], [virtual]

Overwrite the current matrix with unity matrix.

Implements [marathon::TransitionMatrix< T >](#).

Definition at line 64 of file TransitionMatrixCBLAS.h.

3.22.2.5 `template<typename T> virtual void marathon::tm::TransitionMatrixCBLAS< T >::setZero ()`
`[inline],[virtual]`

Overwrite the current matrix with zeroes.

Implements [marathon::TransitionMatrix< T >](#).

Definition at line 74 of file TransitionMatrixCBLAS.h.

3.22.2.6 `template<typename T> virtual std::string marathon::tm::TransitionMatrixCBLAS< T >::to_string () const`
`[inline],[virtual]`

Return a string that represents the matrix.

Implements [marathon::TransitionMatrix< T >](#).

Definition at line 82 of file TransitionMatrixCBLAS.h.

3.22.2.7 `template<typename T> virtual T marathon::tm::TransitionMatrixCBLAS< T >::totalVariationDistance (const`
`T * pi) const [inline],[virtual]`

Compute the total variation distance to the distribution.

Parameters

<i>pi</i>	A probability distribution.
-----------	-----------------------------

Implements [marathon::TransitionMatrix< T >](#).

Definition at line 133 of file TransitionMatrixCBLAS.h.

3.22.2.8 `template<typename T> virtual void marathon::tm::TransitionMatrixCBLAS< T >::variationDistance (const T`
`* pi, T * dist) const [inline],[virtual]`

Compute the variation distance of each state to the distribution *pi*

Parameters

<i>pi</i>	A pointer to a probability distribution.
<i>dist</i>	Out parameter.

Implements [marathon::TransitionMatrix< T >](#).

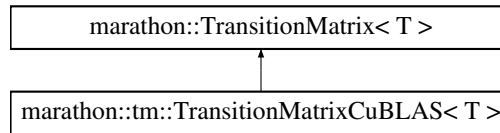
Definition at line 118 of file TransitionMatrixCBLAS.h.

The documentation for this class was generated from the following file:

- include/marathon/TransitionMatrixCBLAS.h

3.23 [marathon::tm::TransitionMatrixCuBLAS< T >](#) Class Template Reference

Inheritance diagram for [marathon::tm::TransitionMatrixCuBLAS< T >](#):



Public Member Functions

- **TransitionMatrixCuBLAS** (const int n)
- **TransitionMatrixCuBLAS** (const [StateGraph](#) *sg)
- virtual void [setEye](#) ()
- virtual void [setZero](#) ()
- virtual std::string [to_string](#) () const
- virtual void [mult](#) (const [TransitionMatrix](#)< T > *A, const [TransitionMatrix](#)< T > *B)
- virtual void [variationDistance](#) (const T *pi, T *dist) const
- virtual T [totalVariationDistance](#) (const T *pi) const

Protected Member Functions

- virtual void [copy](#) (const [TransitionMatrix](#)< T > *P)
- virtual [TransitionMatrix](#)< T > * [generateSubTypeInstance](#) (const int n)

Additional Inherited Members

3.23.1 Detailed Description

template<typename T>class marathon::tm::TransitionMatrixCuBLAS< T >

Definition at line 32 of file TransitionMatrixCuBLAS.h.

3.23.2 Member Function Documentation

3.23.2.1 template<typename T> virtual void marathon::tm::TransitionMatrixCuBLAS< T >::copy (const [TransitionMatrix](#)< T > * P) [inline], [protected], [virtual]

Copy the content of matrix P to this.

Implements [marathon::TransitionMatrix< T >](#).

Definition at line 39 of file TransitionMatrixCuBLAS.h.

3.23.2.2 template<typename T> virtual [TransitionMatrix](#)<T>* marathon::tm::TransitionMatrixCuBLAS< T >::generateSubTypeInstance (const int n) [inline], [protected], [virtual]

Return a pointer to a [Transition](#) matrix of subtype instance.

Implements [marathon::TransitionMatrix< T >](#).

Definition at line 53 of file TransitionMatrixCuBLAS.h.

3.23.2.3 template<typename T> virtual void marathon::tm::TransitionMatrixCuBLAS< T >::mult (const [TransitionMatrix](#)< T > * A, const [TransitionMatrix](#)< T > * B) [virtual]

Multiply A with B and write the result to this.

Parameters

<i>A</i>	A pointer to matrix A. Will not be changed.
<i>B</i>	A pointer to matrix B. Will not be changed.

Implements [marathon::TransitionMatrix< T >](#).

3.23.2.4 `template<typename T> virtual void marathon::tm::TransitionMatrixCuBLAS< T >::setEye ()`
`[inline], [virtual]`

Overwrite the current matrix with unity matrix.

Implements [marathon::TransitionMatrix< T >](#).

Definition at line 86 of file TransitionMatrixCuBLAS.h.

3.23.2.5 `template<typename T> virtual void marathon::tm::TransitionMatrixCuBLAS< T >::setZero ()`
`[inline], [virtual]`

Overwrite the current matrix with zeroes.

Implements [marathon::TransitionMatrix< T >](#).

Definition at line 99 of file TransitionMatrixCuBLAS.h.

3.23.2.6 `template<typename T> virtual std::string marathon::tm::TransitionMatrixCuBLAS< T >::to_string () const`
`[inline], [virtual]`

Return a string that represents the matrix.

Implements [marathon::TransitionMatrix< T >](#).

Definition at line 109 of file TransitionMatrixCuBLAS.h.

3.23.2.7 `template<typename T> virtual T marathon::tm::TransitionMatrixCuBLAS< T >::totalVariationDistance (`
`const T * pi) const [inline], [virtual]`

Compute the total variation distance to the distribution.

Parameters

<i>pi</i>	A probability distribution.
-----------	-----------------------------

Implements [marathon::TransitionMatrix< T >](#).

Definition at line 143 of file TransitionMatrixCuBLAS.h.

3.23.2.8 `template<typename T> virtual void marathon::tm::TransitionMatrixCuBLAS< T >::variationDistance (const`
`T * pi, T * dist) const [inline], [virtual]`

Compute the variation distance of each state to the distribution *pi*

Parameters

<i>pi</i>	A pointer to a probability distribution.
<i>dist</i>	Out parameter.

Implements [marathon::TransitionMatrix< T >](#).

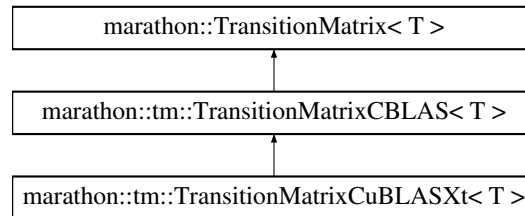
Definition at line 133 of file TransitionMatrixCuBLAS.h.

The documentation for this class was generated from the following file:

- include/marathon/TransitionMatrixCuBLAS.h

3.24 marathon::tm::TransitionMatrixCuBLASXt< T > Class Template Reference

Inheritance diagram for marathon::tm::TransitionMatrixCuBLASXt< T >:



Public Member Functions

- **TransitionMatrixCuBLASXt** (const int n)
- **TransitionMatrixCuBLASXt** (const [StateGraph](#) *sg)
- virtual void **mult** (const [TransitionMatrix](#)< T > *A, const [TransitionMatrix](#)< T > *B)

Additional Inherited Members

3.24.1 Detailed Description

template<typename T>class marathon::tm::TransitionMatrixCuBLASXt< T >

Definition at line 17 of file TransitionMatrixCuBLASXt.h.

3.24.2 Member Function Documentation

3.24.2.1 template<typename T> virtual void marathon::tm::TransitionMatrixCuBLASXt< T >::mult (const [TransitionMatrix](#)< T > * A, const [TransitionMatrix](#)< T > * B) [virtual]

Multiply A with B and write the result to this.

Parameters

<i>A</i>	A pointer to matrix A. Will not be changed.
<i>B</i>	A pointer to matrix B. Will not be changed.

Reimplemented from [marathon::tm::TransitionMatrixCBLAS< T >](#).

The documentation for this class was generated from the following file:

- include/marathon/TransitionMatrixCuBLASXt.h

Index

- ~StateGraph
 - marathon::StateGraph, [20](#)
- addArc
 - marathon::StateGraph, [21](#)
- addLoopArc
 - marathon::StateGraph, [21](#)
- addState
 - marathon::StateGraph, [21](#)
- addTransitionProbability
 - marathon::StateGraph, [21](#)
- clear
 - marathon::StateGraph, [21](#)
- compare_to
 - marathon::chain::bipgraph::DenseBipartiteGraph, [9](#)
 - marathon::chain::matching::BipartiteMatching, [6](#)
 - marathon::State, [18](#)
- computeArbitraryState
 - marathon::chain::bipgraph::SwitchChain, [25](#)
 - marathon::chain::matching::Broder86, [7](#)
 - marathon::MarkovChain, [13](#)
- computeNeighbours
 - marathon::chain::bipgraph::SwitchChain, [25](#)
 - marathon::chain::bipgraph::SwitchChainBerger, [26](#)
 - marathon::chain::matching::Broder86, [7](#)
 - marathon::chain::matching::JerrumSinclairVigoda04, [10](#)
 - marathon::MarkovChain, [13](#)
- computeWeights
 - marathon::chain::matching::JerrumSinclairVigoda04, [10](#)
 - marathon::MarkovChain, [15](#)
- construct
 - marathon::chain::bipgraph::KannanPath, [12](#)
 - marathon::PathConstructionScheme, [16](#)
- copy
 - marathon::tm::TransitionMatrixCBLAS, [31](#)
 - marathon::tm::TransitionMatrixCuBLAS, [33](#)
 - marathon::TransitionMatrix, [28](#)
- expand
 - marathon::StateGraph, [21](#)
- expandState
 - marathon::StateGraph, [22](#)
- generateSubTypeInstance
 - marathon::tm::TransitionMatrixCBLAS, [31](#)
 - marathon::tm::TransitionMatrixCuBLAS, [33](#)
- marathon::TransitionMatrix, [28](#)
- getArc
 - marathon::StateGraph, [22](#)
- getArcs
 - marathon::StateGraph, [22](#)
- getData
 - marathon::TransitionMatrix, [28](#)
- getInArcs
 - marathon::StateGraph, [22](#)
- getInstance
 - marathon::MarkovChain, [15](#)
- getLeadDimension
 - marathon::TransitionMatrix, [28](#)
- getMarkovChain
 - marathon::StateGraph, [22](#)
- getMinWeight
 - marathon::StateGraph, [22](#)
- getN
 - marathon::TransitionMatrix, [28](#)
- getName
 - marathon::MarkovChain, [15](#)
- getNumOutArcs
 - marathon::StateGraph, [22](#)
- getNumStates
 - marathon::StateGraph, [23](#)
- getNumTransitions
 - marathon::StateGraph, [23](#)
- getOutArcs
 - marathon::StateGraph, [23](#)
- getState
 - marathon::StateGraph, [23](#)
- getStates
 - marathon::StateGraph, [23](#)
- getTransitionProbability
 - marathon::StateGraph, [23](#)
- getWeight
 - marathon::StateGraph, [23](#)
- getWeights
 - marathon::StateGraph, [23](#)
- getZ
 - marathon::StateGraph, [23](#)
- hash_value
 - marathon::chain::bipgraph::DenseBipartiteGraph, [9](#)
 - marathon::chain::matching::BipartiteMatching, [6](#)
 - marathon::State, [18](#)
- indexOf
 - marathon::StateGraph, [23](#)

- marathon::MarkovChain, 12
 - computeArbitraryState, 13
 - computeNeighbours, 13
 - computeWeights, 15
 - getInstance, 15
 - getName, 15
 - MarkovChain, 13
 - randomWalk, 15
 - randomize, 15
- marathon::PathConstructionScheme, 16
 - construct, 16
- marathon::State, 18
 - compare_to, 18
 - hash_value, 18
 - operator<<, 19
 - to_string, 18
- marathon::State::Equal, 9
- marathon::State::Hash, 9
- marathon::State::Less, 12
- marathon::StateGraph, 19
 - ~StateGraph, 20
 - addArc, 21
 - addLoopArc, 21
 - addState, 21
 - addTransitionProbability, 21
 - clear, 21
 - expand, 21
 - expandState, 22
 - getArc, 22
 - getArcs, 22
 - getInArcs, 22
 - getMarkovChain, 22
 - getMinWeight, 22
 - getNumOutArcs, 22
 - getNumStates, 23
 - getNumTransitions, 23
 - getOutArcs, 23
 - getState, 23
 - getStates, 23
 - getTransitionProbability, 23
 - getWeight, 23
 - getWeights, 23
 - getZ, 23
 - indexOf, 23
 - setTransitionProbability, 23
 - setWeight, 23
 - StateGraph, 20
- marathon::Transition, 26
- marathon::TransitionComparator, 27
- marathon::TransitionMatrix
 - copy, 28
 - generateSubTypeInstance, 28
 - getData, 28
 - getLeadDimension, 28
 - getN, 28
 - mult, 28
 - operator<<, 30
 - pow, 29
 - setEye, 29
 - setZero, 29
 - swap, 29
 - to_string, 29
 - totalVariationDistance, 29
 - variationDistance, 30
- marathon::TransitionMatrix< T >, 27
- marathon::chain::bipgraph::DenseBipartiteGraph, 8
 - compare_to, 9
 - hash_value, 9
 - to_string, 9
- marathon::chain::bipgraph::KannanPath, 11
 - construct, 12
- marathon::chain::bipgraph::KannanPath::cycle_comparator, 7
- marathon::chain::bipgraph::SwitchChain, 24
 - computeArbitraryState, 25
 - computeNeighbours, 25
 - parseInstance, 25
 - randomize, 25
- marathon::chain::bipgraph::SwitchChainBerger, 25
 - computeNeighbours, 26
- marathon::chain::matching::BipartiteMatching, 5
 - compare_to, 6
 - hash_value, 6
 - to_string, 6
- marathon::chain::matching::Broder86, 6
 - computeArbitraryState, 7
 - computeNeighbours, 7
 - parseInstance, 7
- marathon::chain::matching::JS89Path, 11
- marathon::chain::matching::JerrumSinclairVigoda04, 10
 - computeNeighbours, 10
 - computeWeights, 10
- marathon::chain::matching::SparseBipartiteGraph, 17
- marathon::rational, 16
- marathon::tm::TransitionMatrixCBLAS
 - copy, 31
 - generateSubTypeInstance, 31
 - mult, 31
 - setEye, 31
 - setZero, 32
 - to_string, 32
 - totalVariationDistance, 32
 - variationDistance, 32
- marathon::tm::TransitionMatrixCBLAS< T >, 30
- marathon::tm::TransitionMatrixCuBLAS
 - copy, 33
 - generateSubTypeInstance, 33
 - mult, 33
 - setEye, 34
 - setZero, 34
 - to_string, 34
 - totalVariationDistance, 34
 - variationDistance, 34
- marathon::tm::TransitionMatrixCuBLAS< T >, 32
- marathon::tm::TransitionMatrixCuBLASxt
 - mult, 35

- marathon::tm::TransitionMatrixCuBLASXt< T >, 35
- MarkovChain
 - marathon::MarkovChain, 13
- mult
 - marathon::tm::TransitionMatrixCBLAS, 31
 - marathon::tm::TransitionMatrixCuBLAS, 33
 - marathon::tm::TransitionMatrixCuBLASXt, 35
 - marathon::TransitionMatrix, 28
- operator<<
 - marathon::State, 19
 - marathon::TransitionMatrix, 30
- parseInstance
 - marathon::chain::bipgraph::SwitchChain, 25
 - marathon::chain::matching::Broder86, 7
- pow
 - marathon::TransitionMatrix, 29
- randomWalk
 - marathon::MarkovChain, 15
- randomize
 - marathon::chain::bipgraph::SwitchChain, 25
 - marathon::MarkovChain, 15
- setEye
 - marathon::tm::TransitionMatrixCBLAS, 31
 - marathon::tm::TransitionMatrixCuBLAS, 34
 - marathon::TransitionMatrix, 29
- setTransitionProbability
 - marathon::StateGraph, 23
- setWeight
 - marathon::StateGraph, 23
- setZero
 - marathon::tm::TransitionMatrixCBLAS, 32
 - marathon::tm::TransitionMatrixCuBLAS, 34
 - marathon::TransitionMatrix, 29
- StateGraph
 - marathon::StateGraph, 20
- swap
 - marathon::TransitionMatrix, 29
- to_string
 - marathon::chain::bipgraph::DenseBipartiteGraph, 9
 - marathon::chain::matching::BipartiteMatching, 6
 - marathon::State, 18
 - marathon::tm::TransitionMatrixCBLAS, 32
 - marathon::tm::TransitionMatrixCuBLAS, 34
 - marathon::TransitionMatrix, 29
- totalVariationDistance
 - marathon::tm::TransitionMatrixCBLAS, 32
 - marathon::tm::TransitionMatrixCuBLAS, 34
 - marathon::TransitionMatrix, 29
- variationDistance
 - marathon::tm::TransitionMatrixCBLAS, 32
 - marathon::tm::TransitionMatrixCuBLAS, 34
 - marathon::TransitionMatrix, 30