Python Library of toulbar2

Release 1.0.0

INRAE

Apr 21, 2022

CONTENTS

pytoulbar2 software is the Python interface of toulbar2.

class pytoulbar2.CFN(*ubinit=None*, *resolution=0*, *vac=0*, *configuration=False*, *vns=None*, *seed=1*, *verbose=-1*)

pytoulbar2 base class used to manipulate and solve a cost function network.

See pytoulbar2test.py example in src repository.

AddCompactFunction(*scope*, *defcost*, *tuples*, *tcosts*, *incremental=False*)

AddCompactFunction summary line description.. . .

Description text . . . AddCompactFunction . . .

Parameters

- **scope** (*type...*) – Description text. . .
- **tcosts** (*type...*) – Description text. . .
- **incremental** (*type...*) – Description text. . .

Returns

. . .

AddFunction(*scope*, *costs*, *incremental=False*)

AddFunction summary line description.. . .

Description text . . . AddFunction . . .

Parameters

- **scope** (*type...*) – Description text. . .
- **costs** (*type...*) – Description text. . .
- **incremental** (*type...*) – Description text. . .

Returns

. . .

AddGeneralizedLinearConstraint(*tuples*, *operand='=='*, *rightcoef=0*)

AddLinearConstraint(*coefs*, *scope*, *operand='=='*, *rightcoef=0*)

AddVariable(*name*, *values*)

AddVariable summary line description.. . .

Parameters

- **name** (*type...*) – . . .
- **values** (*type...*) – . . .

Returns

. . .

Assign(*varIndex*, *value*)

ClearPropagationQueues()

Decrease(*varIndex*, *value*)

Depth()

`Domain`(*varIndex*)

`Dump`(*problem*)

`GetNbBacktracks`()

`GetNbConstrs`()

`GetNbNodes`()

`GetNbVars`()

`GetSolutions`()

`GetUB`()

`Increase`(*varIndex*, *value*)

`MultipleAssign`(*varIndexes*, *values*)

`NoPreprocessing`()

`Parse`(*certificate*)

`Read`(*problem*)

`Remove`(*varIndex*, *value*)

`Restore`(*depth*)

`SetUB`(*cost*)

`Solve`(*showSolutions=0*, *allSolutions=0*, *diversityBound=0*)

`SolveFirst`()

SolveFirst performs problem preprocessing before doing incremental solving.

Returns Initial upper bound, possibly improved by considering a worst-case situation based on the sum of maximum finite cost for every function plus one. or None if the problem has no solution (a contradiction occurs during preprocessing).

`SolveNext`(*showSolutions=0*)

AddVariable summary line description. . . .

Parameters

- **name** (*type...*) – . . .
- **values** (*type...*) – . . .

Returns

. . .

`Store`()

`UpdateUB`(*cost*)

`static flatten`(*S*)

INDEX