CPMpy

`from cpmpy import *`

Modèles

m = `Model(`*contraintes*`, minimize = None, maximize = None)`
m `+=` *contrainte* | *liste_de_contraintes*
m`.minimize(`*expr*`)`
m`.maximize(`*expr*`)`
m`.solve(solver=None, time_limit=None)` → bool
`time_limit` en secondes
m`.solveAll(solver=None, display=None, time_limit=None, solution_limit=None)`
`display` = *callback_function* | *exprs_list*
`solution_limit` = *nombre_maximum_de_solutions_à_traiter*
m`.to_file(`*fname*`)`
`Model.from_file(`*fname*`)`

Variables

Les tableaux de variables ont les attributs des tableaux numpy.
bv `= boolvar(shape = 1, name = None)` → une (ou un tableau de) variable(s) booléenne(s)
`shape =` *int* | `(`*int*`,...)` tableau de variables booléennnes
`name =` *s* `:` `str`, *bv*`[...].name` = *'s[...]'* ; sinon *'BVi'*, *i* entier unique
iv `= intvar(`*min_value*`,`*max_value*`, shape = 1, name = None)` → variable(s) entière(s) (idem)
`cpm_array(`*liste*|*array_numpy*`)` → `cpmpy` array indiçable par une variable

Expressions

Les variables booléennes sont traitées comme des entiers

Comparaisons	*x*`==`*y* (*reification* si booléennes), *x*`!=`*y*, *x*`<`*y*, *x*`<=`*y*, *x*`>`*y*, *x*`>=`*y*
Opérateurs mathématiques	`-`*x*, `abs(`*x*`)`, *x*`+`*y*, *x*`-`*y*, *x*`*`*y*, *x*`//`*y*, *x*`%`*y*, *x*`**`*y*, `sum`, `min`, `max`
Opérateurs logiques	*p*`&`*q*, *p*`|`*q*, *p*`^`*q* (xor), `~`*p*, *p*`.implies(`*q*`)`, `any`, `all`
Autres	*x*`.isbool()` (si `True`, utilisable comme contrainte), *x*`.value()`

Contraintes globales

`AllDifferent(*`*args*`)`
`AllDifferentExcept0(*`*args*`)`
`AllEqual(*`*args*`)`
`Circuit(*`*args*`)` si *t*`.shape` = *n*, `Circuit(`*t*`)` ⇔ *t* représente un cycle de longueur *n* de $\{0, \ldots, n-1\}$
`Count(`*t*`,`*val*`)` le nombre d'occurrences de *val* dans *t*
`GlobalCardinalityCount(`*t*`,`*vals*`,`*occ*`)` *occ*`[`*i*`]` doit être = au nb d'occurrences de *vals*`[`*i*`]` dans la liste *t*
`Cumulative(`*start*`,`*duration*`,`*end*`,`*demand*`,`*capacity*`)` planification de tâches
`Element(`*t*`,`*i*`)` *t*`[`*i*`]`
`Maximum(*`*args*`)`
`Minimum(*`*args*`)`
`Table(`*t*`,`*table*`)` les valeurs de *t* forment une des lignes de *table*
`Xor(`*arg_list*`)`
`Inverse(`*fwd*`,`*rev*`)` *fwd*`[`*i*`] ==` *x* ⇔ *rev*`[`*x*`] ==` *i*