Package
data

data
Class DataBase

```
java.lang.Object
   |
   +-data.DataBase
```

Direct Known Subclasses:
DataProbabiliste, DataRecours

---

public abstract class **DataBase**
extends java.lang.Object

Cette classe contient les informations de base concernant une instance du problème.

Field Summary

private	apports Les apports en eau pour chaque période
protected	couts Le vecteur de coûts.
protected	productionsMax La production maximale de chaque centrale
protected	turbinage Le turbinage
protected	volumeInitial Le volume initial d'eau dans le réservoir
protected	volumeMax Le volume maximum d'eau dans le réservoir
protected	volumeMin Le volume minimum d'eau dans le réservoir

Constructor Summary

public	DataBase()

Method Summary

double[]	getApports() Retourne les apports
double	getApportsPeriode(int periode) Retourne les apports d'une période
double	getCoutPeriodeCentrale(int periode, int centrale) Retourne le cout de production d'une centrale pour une période

double[][]	getCouts() Retourne les coûts
double	getProductionMaxCentrale(int centrale) Retourne la production maximale d'une centrale
double[]	getProductionsMax() Retourne les productions maximales
double	getTurbinage() Retourne le turbinage
double	getVolumeInitial() Retourne le volume initial d'eau dans le réservoir
double	getVolumeMax() Retourne le volume maximum d'eau dans le réservoir
double	getVolumeMin() Retourne le volume minimum d'eau dans le réservoir

Methods inherited from class `java.lang.Object`

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, registerNatives, toString, wait, wait, wait`

Fields

couts

`protected double` **`couts`**

Le vecteur de coûts. Pour chaque période et chaque centrale thermique il y a un cout de production en /MW associé et pour la centrale hydrolique un cout d'utilisation de l'eau

productionsMax

`protected double` **`productionsMax`**

La production maximale de chaque centrale

volumeInitial

`protected double` **`volumeInitial`**

Le volume initial d'eau dans le réservoir

volumeMin

`protected double` **`volumeMin`**

Le volume minimum d'eau dans le réservoir

volumeMax

`protected double` **`volumeMax`**

(continued on next page)

(continued from last page)

Le volume maximum d'eau dans le réservoir

turbinage

```
protected double turbinage
```

Le turbinage

apports

```
private double apports
```

Les apports en eau pour chaque période

Constructors

DataBase

```
public DataBase()
```

Methods

getCouts

```
public double[][] getCouts()
```

Retourne les coûts

Returns:
les couts

getProductionsMax

```
public double[] getProductionsMax()
```

Retourne les productions maximales

Returns:
les productions maximales

getVolumeInitial

```
public double getVolumeInitial()
```

Retourne le volume initial d'eau dans le réservoir

Returns:
le volume initial d'eau dans le réservoir

getVolumeMin

```
public double getVolumeMin()
```

Retourne le volume minimum d'eau dans le réservoir

(continued on next page)

(continued from last page)

Returns:
le volume minimum d'eau dans le réservoir

getVolumeMax

```
public double getVolumeMax()
```

Retourne le volume maximum d'eau dans le réservoir

Returns:
le volume maximum d'eau dans le réservoir

getTurbinage

```
public double getTurbinage()
```

Retourne le turbinage

Returns:
le turbinage

getCoutPeriodeCentrale

```
public double getCoutPeriodeCentrale(int periode,
         int centrale)
```

Retourne le cout de production d'une centrale pour une période

Parameters:
`periode` - la période
`centrale` - la centrale

Returns:
le cout de production d'une centrale pour une période

getProductionMaxCentrale

```
public double getProductionMaxCentrale(int centrale)
```

Retourne la production maximale d'une centrale

Parameters:
`centrale` - la centrale

Returns:
les production maximale d'une centrale

getApports

```
public double[] getApports()
```

Retourne les apports

Returns:
les apports

(continued from last page)

getApportsPeriode

```
public double getApportsPeriode(int periode)
```

Retourne les apports d'une période

Returns:
les apports d'une période

data
Class DataBinaire

```
java.lang.Object
   |
   +-data.DataBinaire
```

public class **DataBinaire**
extends java.lang.Object

Cette classe contient les informations concernant une instance du problème sous la forme binaire et sa relaxation.

Field Summary

private	couts Le vecteur de coûts.
	probabilite La probabilité voulue que les scénarios se déroulent
private	scenarios Tableau contenant les différents scénarios

Constructor Summary

public	DataBinaire(java.lang.String fileName) Construit une donnée en fonction d'un fichier

Method Summary

double	getCoutPeriodeCentrale(int periode, int centrale) Retourne le cout de production d'une centrale pour une période
double[][]	getCouts() Retourne les coûts
ScenarioBinaire	getScénario(int scenario) Retourne le scénario voulu
ScenarioBinaire[]	getScenarios() Retourne les scénarios

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, registerNatives, toString, wait, wait, wait

Fields

couts

private double **couts**

(continued from last page)

Le vecteur de coûts. Pour chaque période et chaque centrale thermique il y a un cout de production en /MW associé et pour la centrale hydrolique un cout d'utilisation de l'eau

scenarios

```
private data.ScenarioBinaire scenarios
```

Tableau contenant les différents scénarios

probabilite

```
double probabilite
```

La probabilité voulue que les scénarios se déroulent

Constructors

DataBinaire

```
public DataBinaire(java.lang.String fileName)
```

Construit une donnée en fonction d'un fichier

Parameters:
`fileName` - le chemin vers le fichier de données

Methods

getCouts

```
public double[][] getCouts()
```

Retourne les coûts

Returns:
les couts

getScenarios

```
public ScenarioBinaire[] getScenarios()
```

Retourne les scénarios

Returns:
les scenarios

getScénario

```
public ScenarioBinaire getScénario(int scenario)
```

Retourne le scénario voulu

Parameters:
`scenario` - l'indice du scénario

Returns:
le scénario voulu

getCoutPeriodeCentrale

```
public double getCoutPeriodeCentrale(int periode,
         int centrale)
```

Retourne le cout de production d'une centrale pour une période

Parameters:

`periode` - la période
`centrale` - la centrale

Returns:

le cout de production d'une centrale pour une période

data
Class DataProbabiliste

```
java.lang.Object
   |-data.DataBase
      |-data.DataProbabiliste
```

public class **DataProbabiliste**
extends DataBase

Cette classe contient les informations concernant une instance du problème sous la forme probabiliste.

Field Summary

private	EDemande L'espérance de la demande par période
private	EFacteurDisponibilite L'espérance du facteur de disponibilité par période
private	probabilites Pour chaque période il y a une probabilité minimale souhaitée
private	variance La variance de la loi de distribution voulue

Fields inherited from class data.DataBase

apports, couts, productionsMax, turbinage, volumeInitial, volumeMax, volumeMin

Constructor Summary

public	DataProbabiliste(java.lang.String fileName) Construit une donnée en fonction d'un fichier

Method Summary

double[]	getEDemande() Retourne les espérances des demandes
double	getEDemandePeriode(int periode)
double[]	getEFacteurDisponibilite() Retourne les espérances des facteur de disponibilité
double	getEFacteurDisponibilitePeriode(int periode)
double[]	getProbabilites() Retourne les probabilités
double	getVariance()

Methods inherited from class data.DataBase
getApports, getApportsPeriode, getCoutPeriodeCentrale, getCouts, getProductionMaxCentrale, getProductionsMax, getTurbinage, getVolumeInitial, getVolumeMax, getVolumeMin

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, registerNatives, toString, wait, wait, wait

Fields

probabilites

`private double probabilites`

Pour chaque période il y a une probabilité minimale souhaitée

EFacteurDisponibilite

`private double EFacteurDisponibilite`

L'espérance du facteur de disponibilité par période

EDemande

`private double EDemande`

L'espérance de la demande par période

variance

`private double variance`

La variance de la loi de distribution voulue

Constructors

DataProbabiliste

`public DataProbabiliste(java.lang.String fileName)`

Construit une donnée en fonction d'un fichier

Parameters:

`fileName` - le chemin vers le fichier de données

Methods

getProbabilites

`public double[] getProbabilites()`

Retourne les probabilités

(continued from last page)

Returns:
les probabilités

getEFacteurDisponibilite

`public double[] getEFacteurDisponibilite()`

Retourne les espérances des facteur de disponibilité

Returns:
les espérances des facteur de disponibilité

getEDemande

`public double[] getEDemande()`

Retourne les espérances des demandes

Returns:
les espérances des demandes

getVariance

`public double getVariance()`

Returns:
la variance

getEFacteurDisponibilitePeriode

`public double getEFacteurDisponibilitePeriode(int periode)`

Returns:
the eFacteurDisponibilite

getEDemandePeriode

`public double getEDemandePeriode(int periode)`

Returns:
the eDemande

data
Class DataRecours

```
java.lang.Object
   |-data.DataBase
      |-data.DataRecours
```

public class **DataRecours**
extends DataBase

Cette classe contient les informations concernant une instance du problème sous la forme d'un recours avec scénarios.

Field Summary

private	prixAchat Le prix d'achat de l'énergie par période
private	prixVente Le prix de vente de l'énergie par période
private	scenarios La liste des scénarios

Fields inherited from class data.DataBase

apports, couts, productionsMax, turbinage, volumeInitial, volumeMax, volumeMin

Constructor Summary

public	DataRecours(java.lang.String fileName) Construit une donnée en fonction d'un fichier

Method Summary

double[]	getPrixAchat() Retourne les prix d'achat par période
double	getPrixAchatPeriode(int periode) Retourne le prix d'achat de l'énergie pour une période
double[]	getPrixVente() Retourne les prix de vente par période
double	getPrixVentePeriode(int periode) Retourne le prix d'achat de l'énergie pour une période
ScenarioRecours	getScenario(int scenario) Retourne le scénario voulu
ScenarioRecours[]	getScenarios() Retourne les scénarios

Methods inherited from class data.DataBase

getApports, getApportsPeriode, getCoutPeriodeCentrale, getCouts, getProductionMaxCentrale, getProductionsMax, getTurbinage, getVolumeInitial, getVolumeMax, getVolumeMin

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, registerNatives, toString, wait, wait, wait

Fields

prixAchat

private double **prixAchat**

Le prix d'achat de l'énergie par période

prixVente

private double **prixVente**

Le prix de vente de l'énergie par période

scenarios

private data.ScenarioRecours **scenarios**

La liste des scénarios

Constructors

DataRecours

public **DataRecours**(java.lang.String fileName)

Construit une donnée en fonction d'un fichier

Parameters:
fileName - le chemin vers le fichier de données

Methods

getScenarios

public ScenarioRecours[] **getScenarios**()

Retourne les scénarios

Returns:
les scénarios

getPrixAchat

public double[] **getPrixAchat**()

(continued from last page)

Retourne les prix d'achat par période

Returns:
les prix d'achat par période

getPrixVente

`public double[] getPrixVente()`

Retourne les prix de vente par période

Returns:
les prix de vente par période

getPrixAchatPeriode

`public double getPrixAchatPeriode(int periode)`

Retourne le prix d'achat de l'énergie pour une période

Parameters:
`periode` - la période

Returns:
le prix d'achat de l'énergie pour une période

getPrixVentePeriode

`public double getPrixVentePeriode(int periode)`

Retourne le prix d'achat de l'énergie pour une période

Parameters:
`periode` - la période

Returns:
le prix d'achat de l'énergie pour une période

getScenario

`public ScenarioRecours getScenario(int scenario)`

Retourne le scénario voulu

Parameters:
`scenario` - l'indice du scénario

Returns:
le scénario voulu

data
Class ScenarioBinaire

```
java.lang.Object
   |
   +-data.ScenarioBinaire
```

public class **ScenarioBinaire**
extends java.lang.Object

Cette classe représente un scénario pour le problème binaire

Field Summary

private	demandes Le vecteur de demandes.
private	probabilite La probabilité que le scénario se déroule
private	productions La matrice de productions.

Constructor Summary

public	ScenarioBinaire(double[][][] productions, double[] demandes, double probabilite) Crée un scénario pour le problème binaire à partir des productions et des demandes

Method Summary

double	getDemandePeriode(int periode) Returne la demande de la période souhaitée
double[]	getDemandes() Retourne le vecteur de demandes
double[]	getPaliersPeriodeCentrale(int periode, int centrale) Retourne les paliers de production d'une centrale pour une période
double	getProbabilite()
double[][][]	getProductions() Retourne la matrice de productions

Methods inherited from class `java.lang.Object`

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, registerNatives, toString, wait, wait, wait`

Fields

(continued from last page)

productions

```
private double productions
```

La matrice de productions. Pour chaque période, pour chaque centrale, il y a un vecteur de paliers de production réels affectés par l'aléa

---

demandes

```
private double demandes
```

Le vecteur de demandes. Pour chaque période il y a une demande

---

probabilite

```
private double probabilite
```

La probabilité que le scénario se déroule

Constructors

ScenarioBinaire

```
public ScenarioBinaire(double[][][] productions,
                       double[] demandes,
                       double probabilite)
```

Crée un scénario pour le problème binaire à partir des productions et des demandes

Parameters:

`productions` - les paliers de productions par période par centrale
`demandes` - les demandes par période
`probabilite` - la probabilité que le scénario se déroule

Methods

getProductions

```
public double[][][] getProductions()
```

Retourne la matrice de productions

Returns:

la matrice de productions

---

getDemandes

```
public double[] getDemandes()
```

Retourne le vecteur de demandes

Returns:

le vecteur de demandes

---

getProbabilite

```
public double getProbabilite()
```

(continued from last page)

getPaliersPeriodeCentrale

```
public double[] getPaliersPeriodeCentrale(int periode,
         int centrale)
```

Retourne les paliers de production d'une centrale pour une période

Parameters:

`periode` - la periode
`centrale` - la centrale

Returns:

les paliers de production d'une centrale pour une période

getDemandePeriode

```
public double getDemandePeriode(int periode)
```

Returne la demande de la période souhaitée

Parameters:

`periode` - la période

Returns:

la demande de la période souhaitée

data
Class ScenarioRecours

```
java.lang.Object
   |
   +-data.ScenarioRecours
```

public class **ScenarioRecours**
extends java.lang.Object

Cette classe représente un scénario pour le problème avec recours

Field Summary

private	demandes Le vecteur de demandes.
private	facteursDisponibilite Pour chaque période et chaque centrale thermique il y a un facteur de disponiblité
private	probabilite La probabilité que le scénario se déroule

Constructor Summary

public	ScenarioRecours(double[][] facteurDisponibilite, double[] demandes, double[] apports, double probabilite) Crée un scénario pour le problème binaire à partir des productions et des demandes

Method Summary

double	getDemandePeriode(int periode) Returne la demande de la période souhaitée
double[]	getDemandes() Retourne le vecteur de demandes
double[][]	getFacteursDisponibilite() Retourne la matrice des facteurs de disponibilité
double	getPaliersPeriodeCentrale(int periode, int centrale) Retourne le facteur de disponibilité d'une centrale pour une période
double	getProbabilite()

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, registerNatives, toString, wait, wait, wait

Fields

(continued from last page)

facteursDisponibilite

```
private double facteursDisponibilite
```

Pour chaque période et chaque centrale thermique il y a un facteur de disponiblité

---

demandes

```
private double demandes
```

Le vecteur de demandes. Pour chaque période il y a une demande

---

probabilite

```
private double probabilite
```

La probabilité que le scénario se déroule

Constructors

ScenarioRecours

```
public ScenarioRecours(double[][] facteurDisponibilite,
                       double[] demandes,
                       double[] apports,
                       double probabilite)
```

Crée un scénario pour le problème binaire à partir des productions et des demandes

Parameters:

`productions` - les paliers de productions par période par centrale
`demandes` - les demandes par période
`probabilite` - la probabilité que le scénario se déroule

Methods

getFacteursDisponibilite

```
public double[][] getFacteursDisponibilite()
```

Retourne la matrice des facteurs de disponibilité

Returns:

la matrice des facteurs de disponibilité

---

getDemandes

```
public double[] getDemandes()
```

Retourne le vecteur de demandes

Returns:

le vecteur de demandes

---

getProbabilite

```
public double getProbabilite()
```

(continued from last page)

getPaliersPeriodeCentrale

```
public double getPaliersPeriodeCentrale(int periode,
         int centrale)
```

Retourne le facteur de disponibilité d'une centrale pour une période

Parameters:

`periode` - la periode
`centrale` - la centrale

Returns:

le facteur de disponibilité d'une centrale pour une période

getDemandePeriode

```
public double getDemandePeriode(int periode)
```

Returne la demande de la période souhaitée

Parameters:

`periode` - la période

Returns:

la demande de la période souhaitée

Package
data.solution

data.solution
Class Solution

```
java.lang.Object
   |
   +-data.solution.Solution
```

Direct Known Subclasses:
SolutionEnergieBinaire, SolutionEnergieRecours

---

public abstract class **Solution**
extends java.lang.Object

Classe contenant la solution d'un problème

Constructor Summary

public	Solution()

Method Summary

abstract Solution	clone() Crée une copie de la solution courante.
abstract double	deltaF(Solution solution) Retourne la différence entre deux solutions.
abstract double	fonctionObjectif() Retourne la valeur de la fonction objectif du problème.
abstract void	solutionInitiale() Initialise la solution.

Methods inherited from class `java.lang.Object`

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, registerNatives, toString, wait, wait, wait`

Constructors

Solution

`public` **`Solution`**`()`

Methods

solutionInitiale

`public abstract void` **`solutionInitiale`**`()`

(continued from last page)

Initialise la solution.

deltaF

```
public abstract double deltaF(Solution solution)
```

Retourne la différence entre deux solutions. Une valeur négative si le paramètre est meilleur que la solution courante.

Parameters:
`solution` - la solution à comparer à la solution courante.

Returns:
la différence entre la solution courante et la solution en argument : solution - solution courante

fonctionObjectif

```
public abstract double fonctionObjectif()
```

Retourne la valeur de la fonction objectif du problème.

Returns:
retourne la valeur de la solution avec la fonction objectif du problème.

clone

```
public abstract Solution clone()
```

Crée une copie de la solution courante.

Returns:
une copie de la solution courante

data.solution
Class SolutionEnergieBinaire

```
java.lang.Object
   |-data.solution.Solution
      |-data.solution.SolutionEnergieBinaire
```

public class **SolutionEnergieBinaire**
extends Solution

Une solution du problème de management de la production d'énergie en binaire mais aussi pour sa relaxation.

Field Summary

private	données Les données du problème
private	y Le vecteur de décision
private	z Le vecteur d'activation des scénarios

Constructor Summary

public	SolutionEnergieBinaire(DataBinaire donnees) Crée une nouvelle solution spécifique au problème de management de la production d'énergie sous sa forme binaire et sa relaxation

Method Summary

Solution	clone()
double	deltaF(Solution solution)
double	fonctionObjectif()
void	solutionInitiale()

Methods inherited from class data.solution.Solution

clone, deltaF, fonctionObjectif, solutionInitiale

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, registerNatives, toString, wait, wait, wait

Fields

(continued from last page)

y

`private double y`

Le vecteur de décision

---

z

`private double z`

Le vecteur d'activation des scénarios

---

données

`private data.DataBinaire données`

Les données du problème

Constructors

SolutionEnergieBinaire

`public SolutionEnergieBinaire(DataBinaire donnees)`

Crée une nouvelle solution spécifique au problème de management de la production d'énergie sous sa forme binaire et sa relaxation

Parameters:

`donnees` - les donnees du problème

Methods

clone

`public Solution clone()`

Crée une copie de la solution courante.

---

deltaF

`public double deltaF(Solution solution)`

Retourne la différence entre deux solutions. Une valeur négative si le paramètre est meilleur que la solution courante.

---

fonctionObjectif

`public double fonctionObjectif()`

Retourne la valeur de la fonction objectif du problème.

---

solutionInitiale

`public void solutionInitiale()`

Initialise la solution.

data.solution
Class SolutionEnergieRecours

```
java.lang.Object
   |-data.solution.Solution
      |-data.solution.SolutionEnergieRecours
```

public class **SolutionEnergieRecours**
extends Solution

Une solution du problème de management de la production d'énergie sous forme de recours avec scénarios.

Field Summary

private	x Le vecteur de productions
private	yAchat La matrice contenant l'énergie achetée.
private	yVente La matrice contenant l'énergie vendue.

Constructor Summary

public	SolutionEnergieRecours() Crée une nouvelle solution spécifique au problème de management de la production d'énergie avec recours

Method Summary

Solution	clone()
double	deltaF(Solution solution)
double	fonctionObjectif()
void	solutionInitiale()

Methods inherited from class data.solution.Solution

clone, deltaF, fonctionObjectif, solutionInitiale

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, registerNatives, toString, wait, wait, wait

Fields

(continued from last page)

x

`private double x`

Le vecteur de productions

---

yAchat

`private double yAchat`

La matrice contenant l'énergie achetée. Pour chaque période et chaque scénario il y a de l'énergie achetée

---

yVente

`private double yVente`

La matrice contenant l'énergie vendue. Pour chaque période et chaque scénario il y a de l'énergie vendue

Constructors

SolutionEnergieRecours

`public SolutionEnergieRecours()`

Crée une nouvelle solution spécifique au problème de management de la production d'énergie avec recours

Methods

clone

`public Solution clone()`

Crée une copie de la solution courante.

---

deltaF

`public double deltaF(Solution solution)`

Retourne la différence entre deux solutions. Une valeur négative si le paramètre est meilleur que la solution courante.

---

fonctionObjectif

`public double fonctionObjectif()`

Retourne la valeur de la fonction objectif du problème.

---

solutionInitiale

`public void solutionInitiale()`

Initialise la solution.

Package
manager

manager
Class Manager

```
java.lang.Object
   |
   +-manager.Manager
```

public class **Manager**
extends java.lang.Object

Classe principale gérant la configuration des solveurs et le lien avec l'interface.

Field Summary

private static	solveur Le solveur utilisé pour résourdre le problème

Constructor Summary

public	Manager()

Method Summary

static void	main(java.lang.String[] args) Fonction principale gérant le fonctionnement du programme.

Methods inherited from class `java.lang.Object`

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, registerNatives, toString, wait, wait, wait`

Fields

solveur

`private static manager.solveurs.Solveur` **`solveur`**

Le solveur utilisé pour résourdre le problème

Constructors

Manager

`public` **`Manager`**`()`

Methods

(continued from last page)

main

```
public static void main(java.lang.String[] args)
```

Fonction principale gérant le fonctionnement du programme. Configure un solveur. Lance le solveur. Prend le résultat pour l'afficher.

Parameters:

`args`

manager
Class PreparationMatlab

```
java.lang.Object
   |
   +-manager.PreparationMatlab
```

public class **PreparationMatlab**
extends java.lang.Object

Classe permettant la création d'un fichier exécutable par Matlab pour la résolution du problème déterministe dérivé du problème probabiliste.

Constructor Summary

public	PreparationMatlab()

Method Summary

static void	generer(DataProbabiliste donnees, java.lang.String fichier) Génère un fichier .m utilisable dans Matlab sous la forme d'un programme cvx.

Methods inherited from class `java.lang.Object`

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, registerNatives, toString, wait, wait, wait`

Constructors

PreparationMatlab

```
public PreparationMatlab()
```

Methods

generer

```
public static void generer(DataProbabiliste donnees,
         java.lang.String fichier)
```

Génère un fichier .m utilisable dans Matlab sous la forme d'un programme cvx.

Parameters:

`donnees` - Les données du problème
`fichier` - Le fichier d'écriture

Package
manager.solveurs

manager.solveurs
Interface Solveur

All Known Implementing Classes:
RecuitSimule, PLEnergieBinaireRelaxe, PLEnergieRecours

public interface **Solveur**
extends

Interface correspondant à un algorithme de résolution.

Method Summary

Solution	getSolution() Retourne la solution
void	lancer() Lance la résolution du problème

Methods

lancer

`public void lancer()`

Lance la résolution du problème

getSolution

`public Solution getSolution()`

Retourne la solution

Returns:
la solution trouvée

Package
manager.solveurs.PL

manager.solveurs.PL
Class CplexEnergieBinaireRelaxe

```
java.lang.Object
   |-manager.solveurs.PL.PLEnergieBinaireRelaxe
      |-manager.solveurs.PL.CplexEnergieBinaireRelaxe
```

All Implemented Interfaces:
Solveur

---

public class **CplexEnergieBinaireRelaxe**
extends PLEnergieBinaireRelaxe

Cette classe permet de résoudre une instance du problème de management de la production d'énergie à l'aide de CPLEX

Fields inherited from class `manager.solveurs.PL.PLEnergieBinaireRelaxe`
`couts`, `donnees`, `h`, `probabilites`, `productions`, `solution`, `u`

Constructor Summary

`public`	`CplexEnergieBinaireRelaxe(DataBinaire donnees)` Crée un nouveau CPlexEnergie

Method Summary

`void`	`lancer()` Lance la résolution du programme linéaire.

Methods inherited from class `manager.solveurs.PL.PLEnergieBinaireRelaxe`
`genererPL`, `getSolution`, `lancer`

Methods inherited from class `java.lang.Object`
`clone, equals, finalize, getClass, hashCode, notify, notifyAll, registerNatives, toString, wait, wait, wait`

Methods inherited from interface `manager.solveurs.Solveur`
`getSolution`, `lancer`

Constructors

CplexEnergieBinaireRelaxe

`public` **`CplexEnergieBinaireRelaxe`**`(DataBinaire donnees)`

Crée un nouveau CPlexEnergie

Parameters:

(continued from last page)

`donnees` - les données du problème

Methods

lancer

```
public void lancer()
```

Lance la résolution du programme linéaire. Crée le programme à partir des tableaux de coefficients.

manager.solveurs.PL
Class CplexEnergieRecours

```
java.lang.Object
   |-manager.solveurs.PL.PLEnergieRecours
      |-manager.solveurs.PL.CplexEnergieRecours
```

All Implemented Interfaces:
Solveur

---

public class **CplexEnergieRecours**
extends PLEnergieRecours

Cette classe permet de résoudre une instance du problème de management de la production d'énergie à l'aide de CPLEX

Fields inherited from class manager.solveurs.PL.PLEnergieRecours
couts, donnees, facteursDisponibilite, probabilitesPrix, solution

Constructor Summary

public	CplexEnergieRecours(DataRecours donnees) Crée un nouveau CPlexEnergie

Method Summary

void	lancer() Lance la résolution du programme linéaire.

Methods inherited from class manager.solveurs.PL.PLEnergieRecours
genererPL, getSolution, lancer

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, registerNatives, toString, wait, wait, wait

Methods inherited from interface manager.solveurs.Solveur
getSolution, lancer

Constructors

CplexEnergieRecours

```
public CplexEnergieRecours(DataRecours donnees)
```

Crée un nouveau CPlexEnergie

Parameters:

(continued from last page)

`donnees` - les données du problème

Methods

lancer

```
public void lancer()
```

Lance la résolution du programme linéaire. Crée le programme à partir des tableaux de coefficients.

manager.solveurs.PL
Class PLEnergieBinaireRelaxe

```
java.lang.Object
   |
   +-manager.solveurs.PL.PLEnergieBinaireRelaxe
```

All Implemented Interfaces:
Solveur

Direct Known Subclasses:
CplexEnergieBinaireRelaxe

---

public abstract class **PLEnergieBinaireRelaxe**
extends java.lang.Object
implements Solveur

Cette classe permet d'écrire un programme linéaire à partir de données pour le problème de management de la production d'énergie. Ce PL est utilisé par un solveur de programmes linéaires.

Field Summary

protected	couts Le vecteur de coûts
protected	donnees Variable contenant les données du problème
protected	h La matrice pour l'unicité du palier de production thermique par centrale
protected	probabilites Les probabilités des scénarios
protected	productions La matrice qui pour chaque scénario et chaque période associe la liste des paliers de production et les trajectoires
protected	solution Le vecteur de solution du problème
protected	u Le vecteur pour l'unicité de la trajectoire d'utilisation de l'eau

Constructor Summary

public	PLEnergieBinaireRelaxe(DataBinaire donnees) Crée un nouveau PLEnergie.

Method Summary

void	genererPL() Génère le programme linéaire de la p-mediane à partir des données.

SolutionEnergieBinaire	getSolution() Renvoie la solution calculée.
abstract void	lancer()

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, registerNatives, toString, wait, wait, wait

Methods inherited from interface manager.solveurs.Solveur

getSolution, lancer

Fields

donnees

protected data.DataBinaire **donnees**

Variable contenant les données du problème

solution

protected data.solution.SolutionEnergieBinaire **solution**

Le vecteur de solution du problème

couts

protected double **couts**

Le vecteur de coûts

h

protected int **h**

La matrice pour l'unicité du palier de production thermique par centrale

u

protected int **u**

Le vecteur pour l'unicité de la trajectoire d'utilisation de l'eau

probabilites

protected double **probabilites**

Les probabilités des scénarios

(continued from last page)

productions

`protected double productions`

La matrice qui pour chaque scénario et chaque période associe la liste des paliers de production et les trajectoires

Constructors

PLEnergieBinaireRelaxe

`public PLEnergieBinaireRelaxe(DataBinaire donnees)`

Crée un nouveau PLEnergie.

Parameters:
`donnees` - les données du problème

Methods

genererPL

`private void genererPL()`

Génère le programme linéaire de la p-mediane à partir des données. Remplit les tableaux du problème.

---

getSolution

`public SolutionEnergieBinaire getSolution()`

Renvoie la solution calculée.

Returns:
la solution calculée

---

lancer

`public abstract void lancer()`

manager.solveurs.PL
Class PLEnergieRecours

```
java.lang.Object
   |
   +-manager.solveurs.PL.PLEnergieRecours
```

All Implemented Interfaces:
Solveur

Direct Known Subclasses:
CplexEnergieRecours

---

public abstract class **PLEnergieRecours**
extends java.lang.Object
implements Solveur

Cette classe permet d'écrire un programme linéaire à partir de données pour le problème de management de la production d'énergie. Ce PL est utilisé par un solveur de programmes linéaires.

Field Summary

protected	couts Le vecteur de coûts par période et par centrale
protected	donnees Variable contenant les données du problème
protected	facteursDisponibilite La matrice qui pour chaque scénario et chaque période associe la liste des facteurs de disponibilité
protected	probabilitesPrix Le vecteur contenant les prix par période multiplié par la probabilité de chaque scénarios
protected	solution La solution

Constructor Summary

public	PLEnergieRecours(DataRecours donnees) Crée un nouveau PLEnergie.

Method Summary

void	genererPL() Génère le programme linéaire de la p-mediane à partir des données.
SolutionEnergieRecours	getSolution() Renvoie la solution calculée.
abstract void	lancer()

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, registerNatives, toString, wait, wait, wait

Methods inherited from interface manager.solveurs.Solveur
getSolution, lancer

Fields

donnees

protected data.DataRecours **donnees**

Variable contenant les données du problème

solution

protected data.solution.SolutionEnergieRecours **solution**

La solution

probabilitesPrix

protected double **probabilitesPrix**

Le vecteur contenant les prix par période multiplié par la probabilité de chaque scénarios

couts

protected double **couts**

Le vecteur de coûts par période et par centrale

facteursDisponibilite

protected double **facteursDisponibilite**

La matrice qui pour chaque scénario et chaque période associe la liste des facteurs de disponibilité

Constructors

PLEnergieRecours

public **PLEnergieRecours**(DataRecours donnees)

Crée un nouveau PLEnergie.

Parameters:
donnees - les données du problème

Methods

(continued from last page)

genererPL

```
private void genererPL()
```

Génère le programme linéaire de la p-mediane à partir des données. Remplit les tableaux du problème.

---

getSolution

```
public SolutionEnergieRecours getSolution()
```

Renvoie la solution calculée.

Returns:
la solution calculée

---

lancer

```
public abstract void lancer()
```

Package
manager.solveurs.RS

manager.solveurs.RS
Class RecuitSimule

```
java.lang.Object
   |
   +-manager.solveurs.RS.RecuitSimule
```

All Implemented Interfaces:
Solveur

Direct Known Subclasses:
RSEnergie

---

public abstract class **RecuitSimule**
extends java.lang.Object
implements Solveur

Implémentation générale du recuit simulé.

Field Summary

protected	`facteurDecroissance` Le facteur de décroissance de la température du recuit.
protected	`meilleureF` La valeur de la meilleure solution trouvée.
protected	`meilleureSolution` La meilleure solution trouvée pour le moment.
protected	`solutionCourante` La solution courante du recuit simulé.
protected	`tauxAcceptation` Le taux d'acceptation de solutions coûteuses acceptées par le recuit à la température initiale.
protected	`temperature` La température du recuit.

Constructor Summary

protected	`RecuitSimule(double facteurDecroissance, double tauxAcceptation)` Construit un recuit simulé.

Method Summary

void	`decroitreTemperature()` Décroit la température du recuit.
Solution	`getSolution()`
void	`initialiserTemperature()` Initialise la température du recuit.

void	lancer() Démarre le recuit simulé.
abstract boolean	testerCondition1() Teste si le recuit est arrivé à sa température minimale.
abstract boolean	testerCondition2() Teste si le recuit doit continuer à un palier de température.
abstract Solution	voisin() Sélectionne une solution dans le voisinage de la solution courante.

Methods inherited from class `java.lang.Object`

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, registerNatives, toString, wait, wait, wait`

Methods inherited from interface `manager.solveurs.Solveur`

`getSolution, lancer`

Fields

solutionCourante

`protected data.solution.Solution` **`solutionCourante`**

La solution courante du recuit simulé.

meilleureSolution

`protected data.solution.Solution` **`meilleureSolution`**

La meilleure solution trouvée pour le moment.

facteurDecroissance

`protected double` **`facteurDecroissance`**

Le facteur de décroissance de la température du recuit.

temperature

`protected double` **`temperature`**

La température du recuit.

meilleureF

`protected double` **`meilleureF`**

La valeur de la meilleure solution trouvée.

(continued from last page)

tauxAcceptation

```
protected double tauxAcceptation
```

Le taux d'acceptation de solutions coûteuses acceptées par le recuit à la température initiale.

Constructors

RecuitSimule

```
protected RecuitSimule(double facteurDecroissance,
                       double tauxAcceptation)
```

Construit un recuit simulé.

Parameters:

`facteurDecroissance` - le facteur de décroissance de la température.

Methods

lancer

```
public void lancer()
```

Démarre le recuit simulé. Implémente le coeur de l'algorithme du recuit simulé commun à tous les problèmes.

---

initialiserTemperature

```
private void initialiserTemperature()
```

Initialise la température du recuit. La température initiale doit accepter un certain nombre de solutions coûteuses. Ce taux est fixé par l'utilisateur.

---

decroitreTemperature

```
private void decroitreTemperature()
```

Décroit la température du recuit. La fonction utilisée est : f(t) = alpha x t avec alpha fixée par l'utilisateur.

---

voisin

```
protected abstract Solution voisin()
```

Sélectionne une solution dans le voisinage de la solution courante.

Returns:

une solution voisine de la solution courante.

---

testerCondition1

```
protected abstract boolean testerCondition1()
```

Teste si le recuit est arrivé à sa température minimale.

Returns:

true si le recuit peut passer au palier de température suivant, false sinon.

(continued from last page)

testerCondition2

```
protected abstract boolean testerCondition2()
```

Teste si le recuit doit continuer à un palier de température.

Returns:
true si le recuit doit continuer à ce palier de température, false sinon.

---

getSolution

```
public Solution getSolution()
```

manager.solveurs.RS
Class RSEnergie

```
java.lang.Object
   |-manager.solveurs.RS.RecuitSimule
      |-manager.solveurs.RS.RSEnergie
```

All Implemented Interfaces:
Solveur

---

public class **RSEnergie**
extends RecuitSimule

Implémentation spécialisée du recuit simulé pour le problème de management de la production d'énergie.

Field Summary

private	donnees Les données du problème
private	iterationCourante Le nombre d'itérations courant pour le palier de température courant.
private	nbIterationsParPalier Le nombre d'itérations par palier de température
private	temperatureFinale La température à atteindre pour arrêter le recuit.

Fields inherited from class manager.solveurs.RS.RecuitSimule

facteurDecroissance, meilleureF, meilleureSolution, solutionCourante, tauxAcceptation, temperature

Constructor Summary

public	RSEnergie(double facteurDecroissance, DataBinaire donnees, double temperatureFinale, int nbIterationsParPalier, double tauxAcceptation) Construit un recuit simulé spécialisé pour le problème de la p-médiane.

Method Summary

boolean	testerCondition1() Teste si le recuit est arrivé la température finale demandée à la création.
boolean	testerCondition2() Teste si le recuit a encore des itérations à faire pour un palier de température.
Solution	voisin() Sélectionne une solution dans le voisinage de la solution courante en modifiant l'activation d'un scénario et une décision.

Methods inherited from class manager.solveurs.RS.RecuitSimule

decroitreTemperature, getSolution, initialiserTemperature, lancer, testerCondition1, testerCondition2, voisin

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, registerNatives, toString, wait, wait, wait

Methods inherited from interface manager.solveurs.Solveur
getSolution, lancer

Fields

donnees

```
private data.DataBinaire donnees
```

Les données du problème

nbIterationsParPalier

```
private int nbIterationsParPalier
```

Le nombre d'itérations par palier de température

iterationCourante

```
private int iterationCourante
```

Le nombre d'itérations courant pour le palier de température courant.

temperatureFinale

```
private double temperatureFinale
```

La température à atteindre pour arrêter le recuit.

Constructors

RSEnergie

```
public RSEnergie(double facteurDecroissance,
                 DataBinaire donnees,
                 double temperatureFinale,
                 int nbIterationsParPalier,
                 double tauxAcceptation)
```

Construit un recuit simulé spécialisé pour le problème de la p-médiane.

Parameters:

`facteurDecroissance` - le facteur de décroissance de la température du recuit.
`temperatureFinale` - la température à atteindre pour arrêter le recuit.

Methods

(continued from last page)

testerCondition1

```
protected boolean testerCondition1()
```

Teste si le recuit est arrivé la température finale demandée à la création.

Returns:
true si le recuit peut passer au palier de température suivant, false sinon.

testerCondition2

```
protected boolean testerCondition2()
```

Teste si le recuit a encore des itérations à faire pour un palier de température.

Returns:
true si le recuit doit continuer à ce palier de température, false sinon.

voisin

```
protected Solution voisin()
```

Sélectionne une solution dans le voisinage de la solution courante en modifiant l'activation d'un scénario et une décision. Un scénario actif devient inactif et inversement. Il faut que la nouvelle configuration ait une probabilité supérieure à celle demandée. Une décision passe de 0 à 1 ou de 1 à 0. Il faut que la nouvelle configuration respecte l'offre et la demande et les contraintes d'unicitées.

Returns:
une solution voisine de la solution courante.