
www.localsolver.com

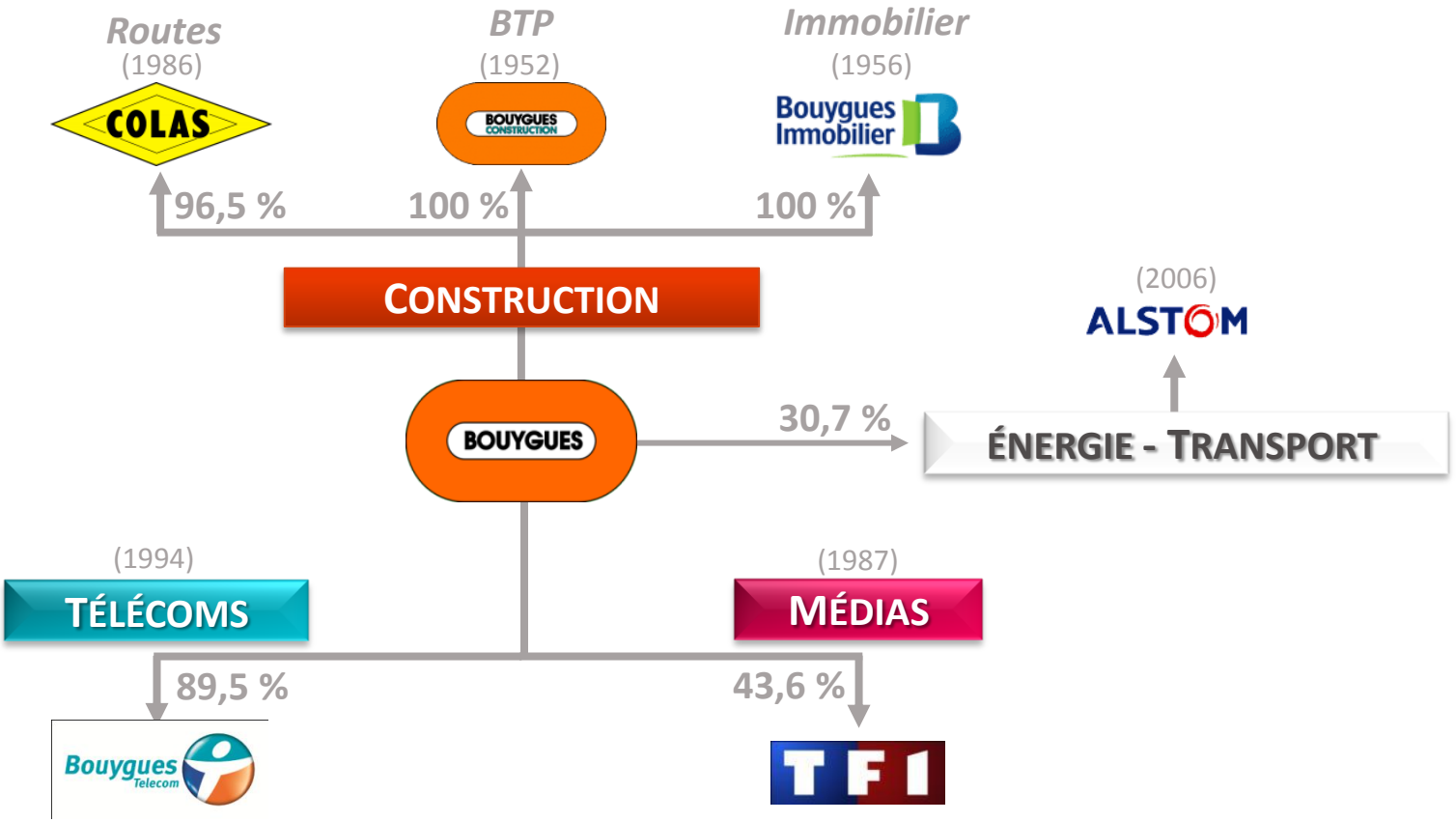
# Bouygues

Une culture de l'innovation

# Le Groupe Bouygues

## 5 métiers

CA : 32,7 Mds €




# Innovation, recherche & développement

## Une forte culture d'entreprise

- Chantiers = culture projet, culture entrepreneur
- Valeurs = responsabilité, confiance, respect
- Recherche permanente de valeur ajoutée pour le client
- Innovation : vitale dans tous nos métiers

## Une équipe de 12 personnes (ingénieurs et docteurs) au service des projets d'innovation et d'optimisation de tous les métiers du Groupe

- Réalisation de projets pour les différentes filiales (planning de terrassement, optimisation de revenu, etc.)
- Animation de l'innovation (filière Innovation, réseau)
- Recherche et partenariats (CEA, startups, universités)

# LocalSolver

Une vision de l'optimisation

# Les besoins de l'entreprise

## Retour d'expérience après 15 ans de RO

- Les entreprises veulent des logiciels de RO :
  - Pour améliorer leur prise de décision
  - Pour accélérer leur prise de décision

→ Les métiers veulent de meilleures solutions plus vite

## Le phénomène du « big data »

- Les entreprises font face à des problèmes d'optimisation :
  - Toujours plus grands : plus de dimensions
  - Toujours plus complexes : plus de contraintes, plus d'objectifs

→ La taille des problèmes croît plus vite que la puissance de calcul disponible

# Les besoins de l'ingénieur

## L'ingénieur optimisation veut satisfaire ses clients

- Produire des études ou des logiciels fournissant :
  - de très bonnes solutions
  - en des temps très courts

## L'ingénieur optimisation veut accroître sa productivité

- Accélérer les phases d'étude et de prototypage
- Réduire les temps de développement
- Simplifier évolutions et maintenance

→ Il utilise des solveurs de programmation mathématique

# LocalSolver 2.0

## Un solveur aligné sur les besoins de l'entreprise

- Fournit de très bonnes solutions en quelques secondes
- Passe à l'échelle : traite jusqu'à 10 millions de décisions
- Prouve l'optimalité dans certains cas

## Un solveur aligné sur les besoins de l'ingénieur

- « Model & Run » :
  - Modélisation mathématique simple
  - Résolution directe : ne requiert aucun paramétrage
- Un prix simple et transparent

# LocalSolver 2.0

## Un solveur de nouvelle génération

- **Recherche de bonnes solutions par recherche locale**
- Calcule une borne inférieure (relaxation, inférence, coupes)

## Un produit complet, de qualité

- Un langage de modélisation innovant pour le prototypage rapide
- Des APIs légères et orientées objet : quelques classes seulement
- Fiable et robuste : méthodologie d'intégration continue
- Portable : Windows, Linux, Mac OS (x86, x64)
- Un support réactif, réalisé par les concepteurs eux-mêmes

# LocalSolver's story

## 2007 : Démarrage du projet LocalSolver

- Définir un formalisme mathématique déclaratif simple et adapté à la recherche locale (*model*)
- Développer un solveur efficace fondé sur la recherche locale, avec comme principe fondamental : « faire ce qu'un expert ferait » (*run*)
- 4 experts : T. Benoist, B. Estellon, F. Gardi, K. Nouioua

→ **Prix Robert Faure 2012**

## 2010-2011 : LocalSolver 1.x, première concrétisation

- Résout des modèles fortement non linéaires en variables 0-1
- Capable d'attaquer des problèmes à 100 000 décisions binaires
- L'équipe se renforce : arrivée de R. Megel et J. Darlay

# Pourquoi la recherche locale ?

## Les défauts d'une recherche arborescente

- Peu adaptée à la recherche de solutions « entières »
- Conçue pour prouver l'optimalité
- Temps exponentiel : ne passe pas à l'échelle (les meilleurs solveurs de PLNE butent toujours sur des instances à 10 000 décisions binaires)

## Les avantages de la recherche locale

- Fournit de bonnes solutions en des temps très limités
- Passe à l'échelle (chaque itération se fait en temps sous-linéaire)
- Adaptée aux espaces non convexes et non lisses

# Pourquoi la recherche locale ?

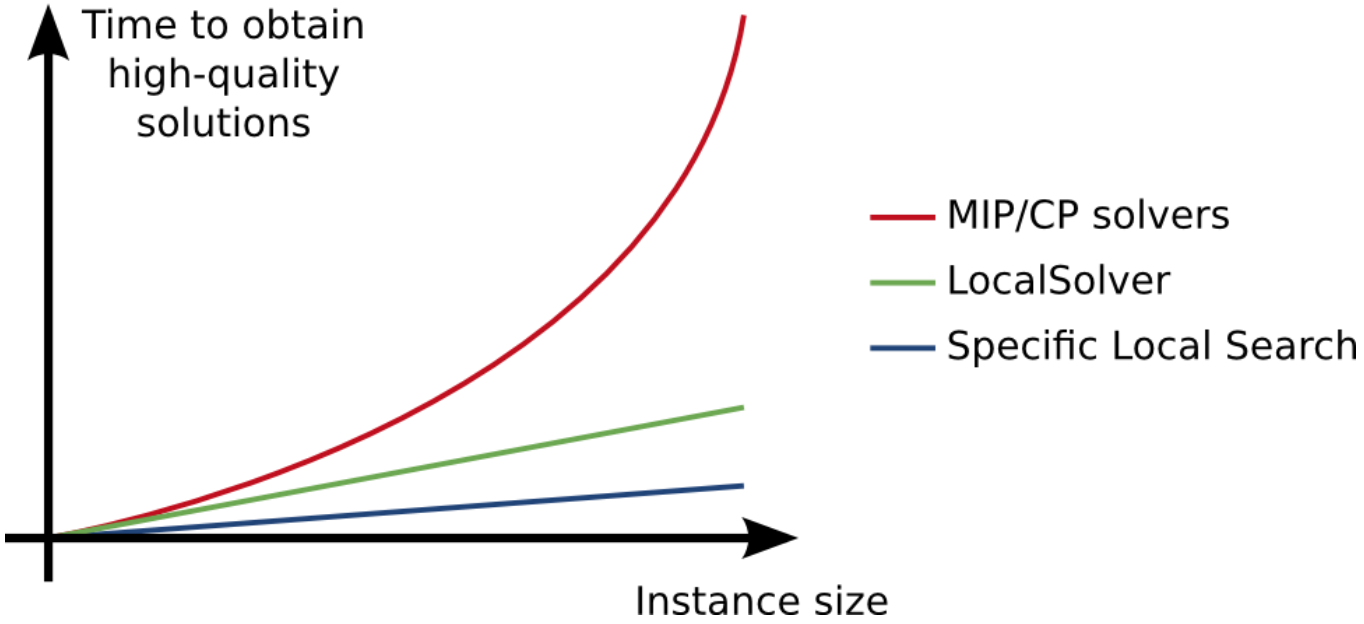
## Notre vision

- Les méthodes de recherche arborescente resteront limitées face aux problèmes combinatoires de très grande taille (1 million de décisions)
- Il faut distinguer recherche de solution et recherche de borne inférieure : trop souvent la relaxation n'apporte rien à la recherche
- Pourquoi une recherche arborescente incomplète serait-elle meilleure qu'une recherche locale ?

## Des faits !

- Lançons le meilleur solveur de PLNE sur le problème du *Car Sequencing Renault* (Challenge ROADEF 2005) et observons…

# Pourquoi la recherche locale ?




# LocalSolver : feuille de route

## À moyen terme

- Une ambition : devenir la référence en programmation non linéaire en variables mixtes (*MINLP)*
  - Décisions discrètes et continues
  - Contraintes et objectifs non convexes
  - Pour l'optimisation à grande échelle : millions de décisions
  - Centré sur la résolution de problématiques réelles

## À long terme

- Modélisation/résolution des problèmes d'ordonnancement et de routage
  - Programmation à base d'objets et d'ensembles, munis d'opérateurs relationnels

# LocalSolver

Un tour du produit en 5 min

# L'exemple du sac-à-dos

8 objets à placer dans un sac, sans excéder un poids de 102 kg et en maximisant la valeur des objets.

```
function model() {
  // 0-1 decisions
  x_0 <- bool(); x_1 <- bool(); x_2 <- bool(); x_3 <- bool();
  x_4 <- bool(); x_5 <- bool(); x_6 <- bool(); x_7 <- bool();

  // weight constraint
  knapsackWeight <- 10*x_0+ 60*x_1+ 30*x_2+ 40*x_3+ 30*x_4+ 20*x_5+ 20*x_6+ 2*x_7;
  constraint knapsackWeight <= 102;

  // maximize value
  knapsackValue <- 1*x_0+ 10*x_1+ 15*x_2+ 40*x_3+ 60*x_4+ 90*x_5+ 100*x_6+ 15*x_7;
  maximize knapsackValue;
}
```

Variables de décision binaires

Variables intermédiaires entières ou binaires

L'utilisateur n'a rien d'autre à faire que d'écrire ce modèle

(approche déclarative = model & run)

# L'exemple du sac-à-dos avec 2 objectifs

```
function model() {
  // 0-1 decisions
  x[0..7] <- bool();

 // weight constraint
  knapsackWeight <- 10*x[0]+ 60*x[1]+ 30*x[2]+ 40*x[3]+ 30*x[4]+ 20*x[5]+ 20*x[6]+ 2*x[7];
  constraint knapsackWeight <= 102;

  // maximize value
  knapsackValue <- 1*x[0]+ 10*x[1]+ 15*x[2]+ 40*x[3]+ 60*x[4]+ 90*x[5]+ 100*x[6]+ 15*x[7];
  maximize knapsackValue;

  // secondary objective: minimize product of minimum and maximum values
  knapsackMinValue <- min[i in 0..7](x[i] ? values[i]  : 1000);
  knapsackMaxValue <- max[i in 0..7](x[i] ? values[i]  : 0);
  knapsackProduct <- knapsackMinValue * knapsackMaxValue;
  minimize knapsackProduct;
}
```

Opérateurs fortement non linéaires : sum, product, <, >, min, and, or, if, …

Objectifs lexicographiques

# Opérateurs mathématiques

| Arithmétiques | | Logiques | Relationnels |
|---|---|---|---|
| sum | mod | not | != |
| prod | abs | and | >= |
| min | dist | or | <= |
| max | sqrt | xor | > |
| div | | if | < |

# Langage de modélisation

```
function model() {
  // 0-1 decisions
  x[i in 1..nbItems] <- bool();

  // weight constraint
  knapsackWeight <- sum[i in 1..nbItems](weights[i] * x[i]);
  constraint knapsackWeight <= knapsackBound;

  // maximize value
  knapsackValue <- sum[i in 1..nbItems](values[i] * x[i]);
  maximize knapsackValue;
}
```

**A[3] = 8;**
➔ A est une table et contient 8 en position 3

**X * Y**
➔ Est une variable si X ou Y sont des variables, un entier sinon

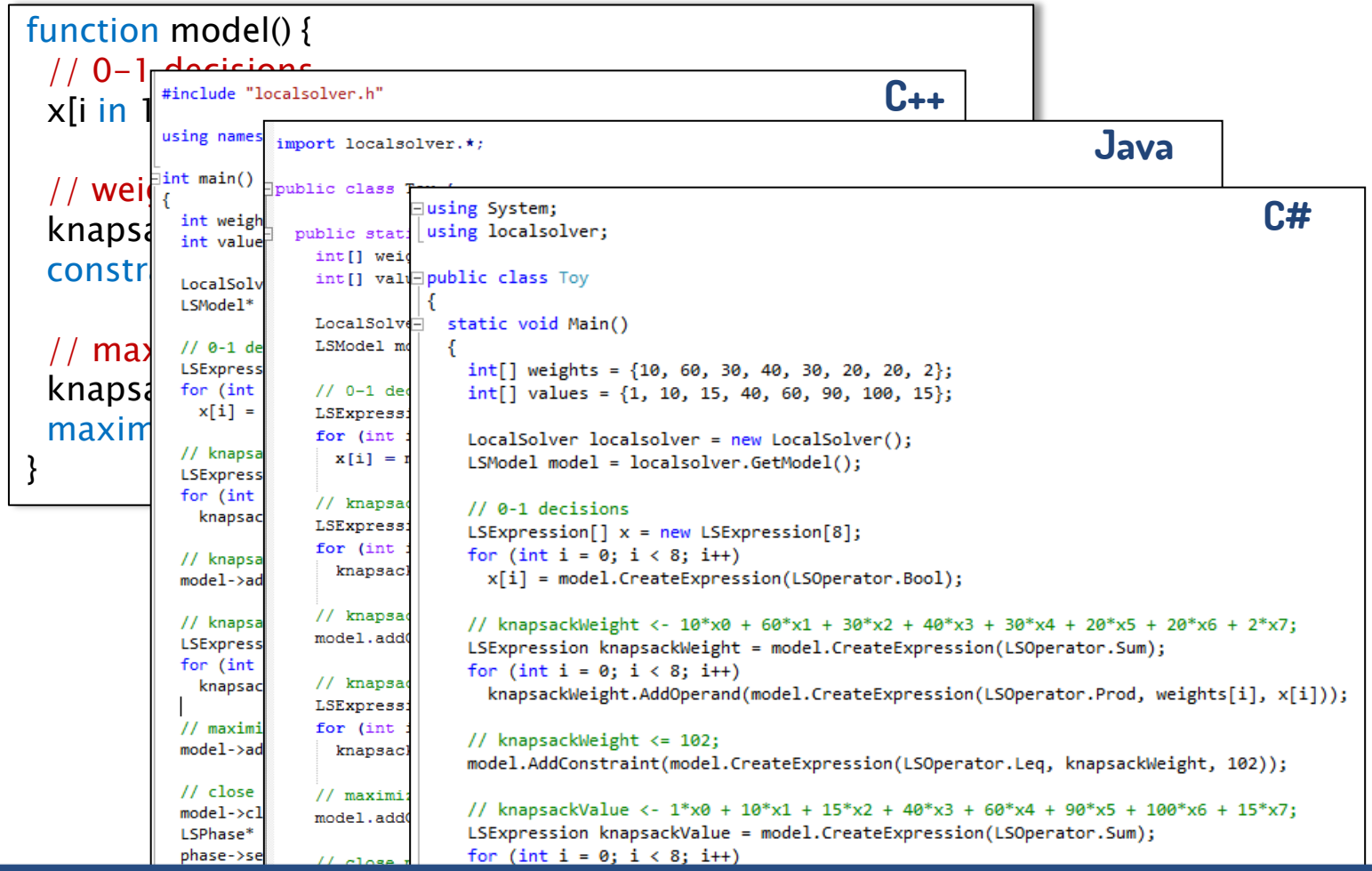
Boucles, fonctions, tables de variables,

Lecture de données dans un fichier, etc.

Adapté à la recherche locale et moins verbeux que les modeleurs existants (OPL, Mosel, AMPL)

- Interprété
- Typage fort et dynamique (comme Python)
- Déclaration implicite des variables (comme PHP, Lua)
- Même opérateurs pour la modélisation et la programmation

# Langage de modélisation




Le modeler permet de prototyper très rapidement.
Pour un déploiement en production LocalSolver offre des API
C++, .NET, Java

# Exemples

Car Sequencing, Challenge Roadef/Google

# Car Sequencing

## Ordonnancement de véhicules sur une chaîne de production

## Objectif = espacer les options

- ex: pas plus de 2 toits ouvrants toutes les 5 voitures («P/Q»)
- mesure: dans chaque fenêtre de 5, pénalité basée sur les dépassements = max(n-2,0) avec *n* le nombre de toits ouvrants.

## Une classe est un ensemble de véhicules identiques

- Ici avec 3 options A, B et C: AB est la classes des véhicules ayant les options A et B

| AB | A | B | AB | A | ABC | C | A | B | C |
|---|---|---|---|---|---|---|---|---|---|

**Lecture des données**

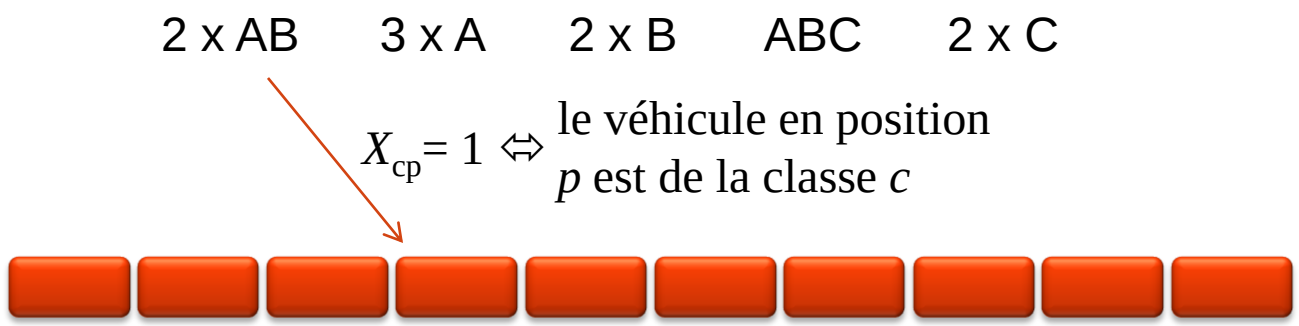
```
datafile = openRead(datafilename);
nbPositions = readInt(datafile);
nbOptions = readInt (datafile);
nbClasses = readInt (datafile);
P[o in 1..nbOptions] = readInt (datafile);
Q[o in 1..nbOptions] = readInt(datafile);
for[c in 1..nbClasses] {
  card[c] = readInt(datafile);
  options[c][1..nbOptions] = (readInt(datafile)==1);
}
```

Typage fort (openRead renvoie un FILE)

Déclaration implicite

Syntaxe compacte

# Modèle




```
X[c in 1..nbClasses][p in 1..nbPositions] <- bool();

for[c in 1..nbClasses]
  constraint sum[p in 1..nbPositions](X[c][p]) == card[c];

for[p in 1..nbPositions]
  constraint sum[c in 1..nbClasses](X[c][p]) == 1;

op[o in 1..nbOptions][p in 1..nbPositions] <-
        or[c in 1..nbClasses : options[c][o]](X[c][p]);

nbVehicles[o in 1..nbOptions][j in 1..nbPositions-Q[o]+1] <-
            sum[k in 1..Q[o]](op[o][j+k-1]);

violations[o in 1..nbOptions][j in 1..nbPositions-Q[o]+1] <- max(nbVehicles[o][j] - P[o], 0 );

obj <- sum[o in 1..nbOptions][p in 1..nbPositions-Q[o]+1](violations[o][p]);
```

Et c'est tout !

# Car Sequencing

Et s'il est interdit d'avoir plus de 10 voitures consécutives de la même couleur ?


```
Color[p in 1..nbPositions] <- sum[c in 1..nbClasses](Color[c] * X[c][p]);
Same[p in 2..nbPositions] <- color[p] == color[p-1];
TooLong[p in 11..nbPositions] <- and[j in p-10..p](Same[j]);
for [p in 11..nbPositions] constraint not(TooLong[p]);
```

Les variables de décision ne changent pas

Et pour le challenge roadef 2005:

```
maximize sum[i in 2..nbPositions](Same[i]);
minimize sum[i in 1..nbPositions](Penalty[i]);
```

# Résolution

## Que va faire LocalSolver pour ce modèle ?

1. Trouver une solution initiale (ici une affectation valide des véhicules)
2. Appliquer les mouvements génériques

Echanges à 2


Echanges à 3 ou plus


Déplacements simples

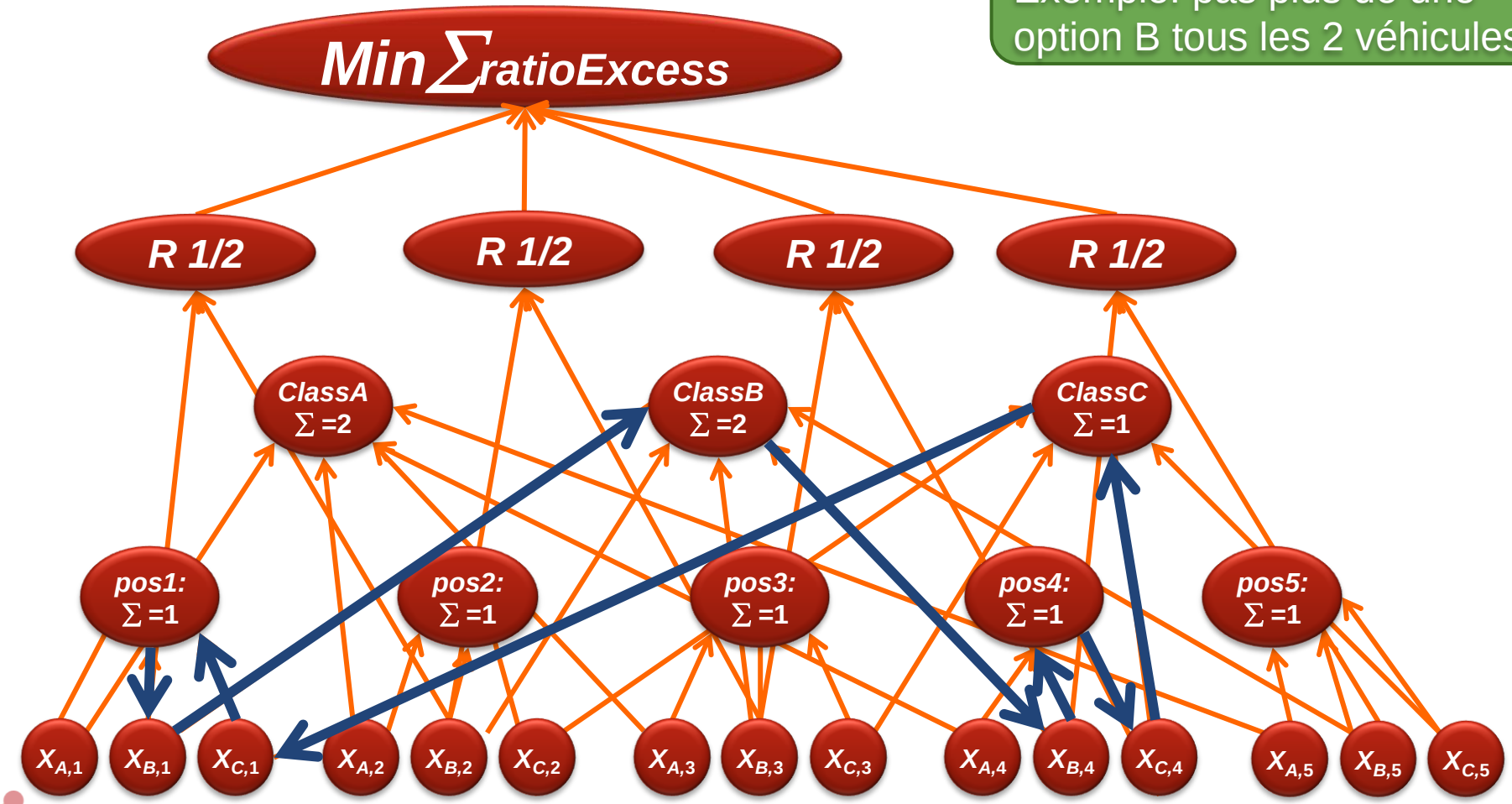

**Contrainte violée !**
**(2 voitures à la même position)**

Etc.

- A noter:
  - Les déplacements simples seront éliminés après quelques secondes car ils échouent systématiquement
  - La stratégie globale est un recuit simulé randomisé (paramétrable)
  - LocalSolver lance en parallèle plusieurs recherches (nombre de threads paramétrable)
  - Certains mouvements seront ciblés sur les fenêtres avec dépassement

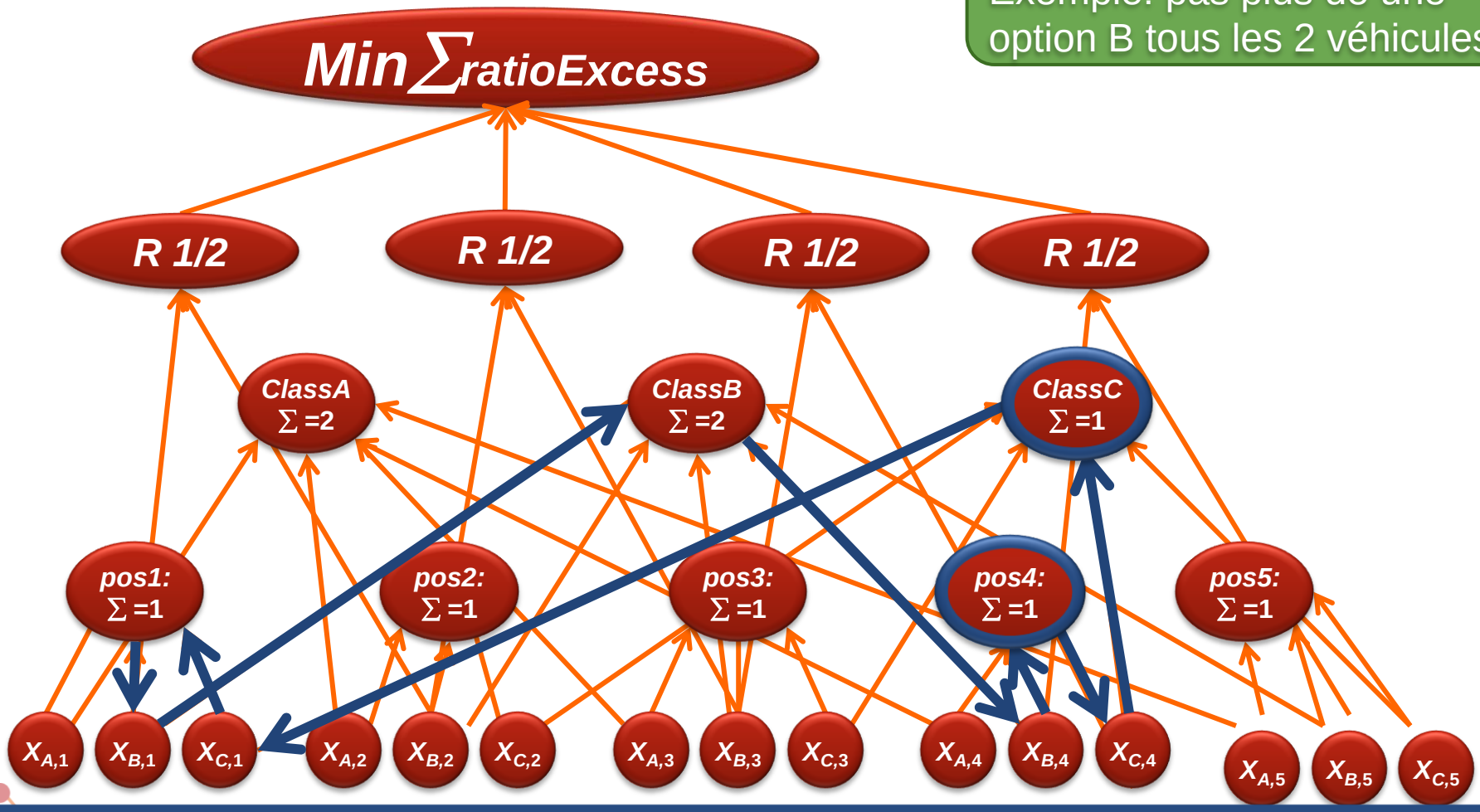
# Mouvements génériques

Exemple: pas plus de une option B tous les 2 véhicules



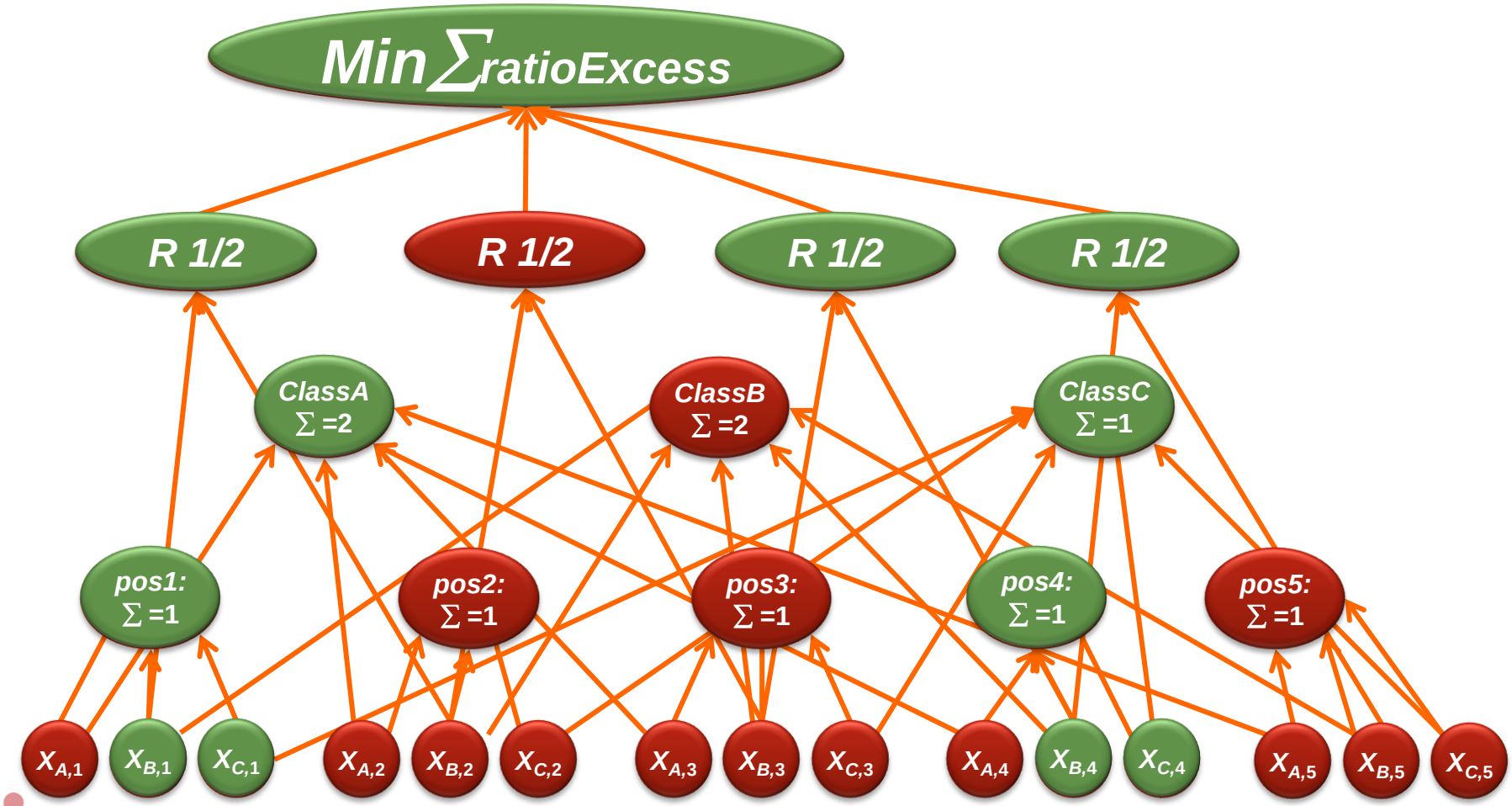
# Mouvements génériques

Exemple: pas plus de une option B tous les 2 véhicules




En utilisant ces structures, LocalSolver effectue des mouvements proches de ce qu'un praticien expert aurait programmé

# Evaluation incrémentale




Grâce à ces mécanismes incrémentaux,

LocalSolver visite des millions de solutions en quelques minutes

# Pour en savoir plus

T. Benoist, B. Estellon, F. Gardi, R. Megel, K. Nouioua.
*LocalSolver 1.x: a black-box local-search solver for 0-1 programming.*
**4OR, A Quarterly Journal of Operations Research**
9(3), pp. 299-316.

# Challenge Roadef 2005

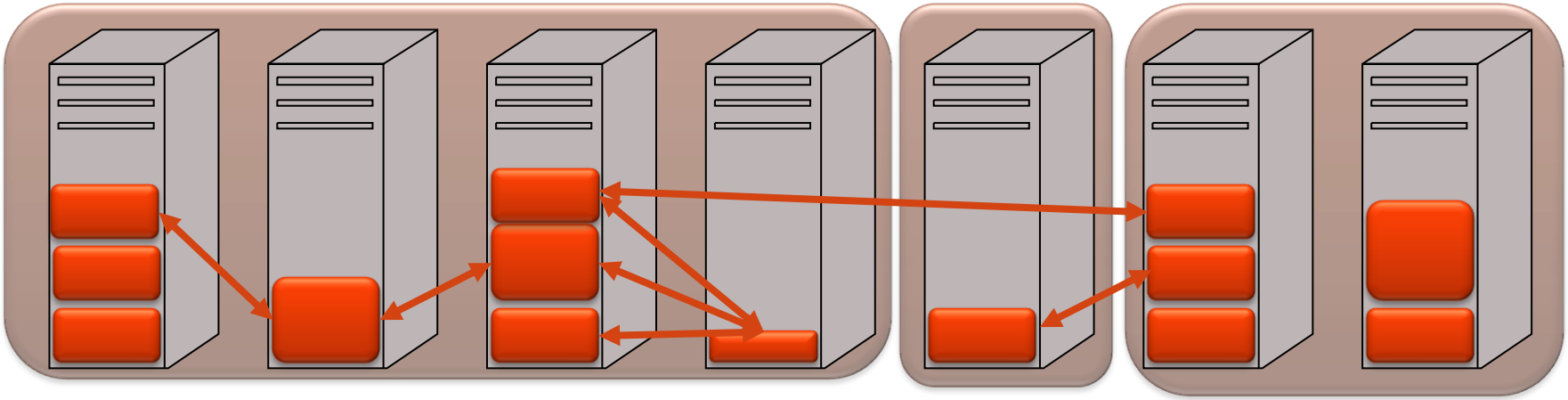
Sur les instances du challenge Roadef 2005 (Renault)

LocalSolver se classe entre la 16eme et la 18eme positions (sur 19 candidats)

Avec un programme de 120 lignes !

# Challenge Roadef 2012

Des processus sur des machines, dépendants les uns des autres, à réaffecter pour optimiser une fonction de coût

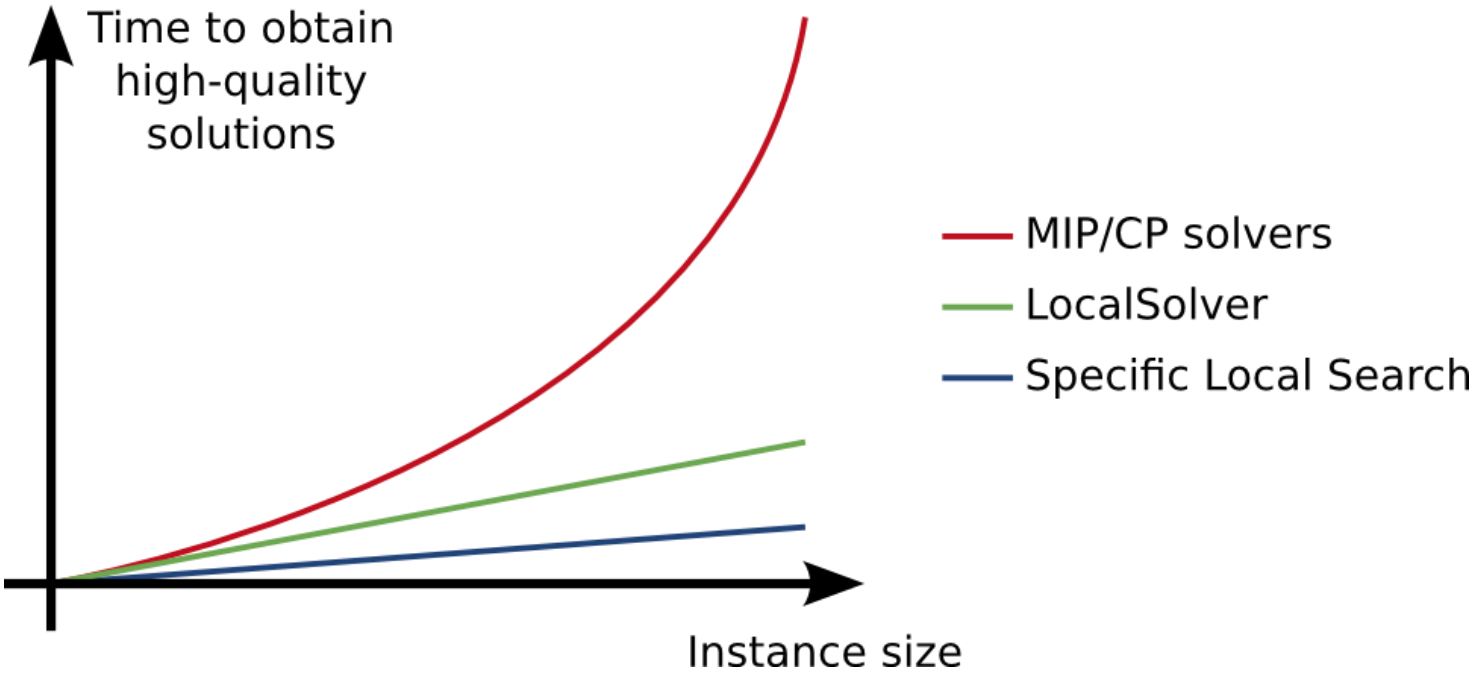


Plus de 100 000 variables binaires

Temps de développement: 4 heures

**LocalSolver qualifié pour la finale (25e sur 80)**

# Challenge Roadef 2012




Plus de 100 000 variables binaires

Temps de développement: 4 heures

**LocalSolver qualifié pour la finale (25e sur 80)**

# Applications




# Affectation des emplacements préférentiels

- Problème d'affectation avec contraintes non linéaires (max)
- 5 objectifs lexicographiques
- 10 000 variables binaires

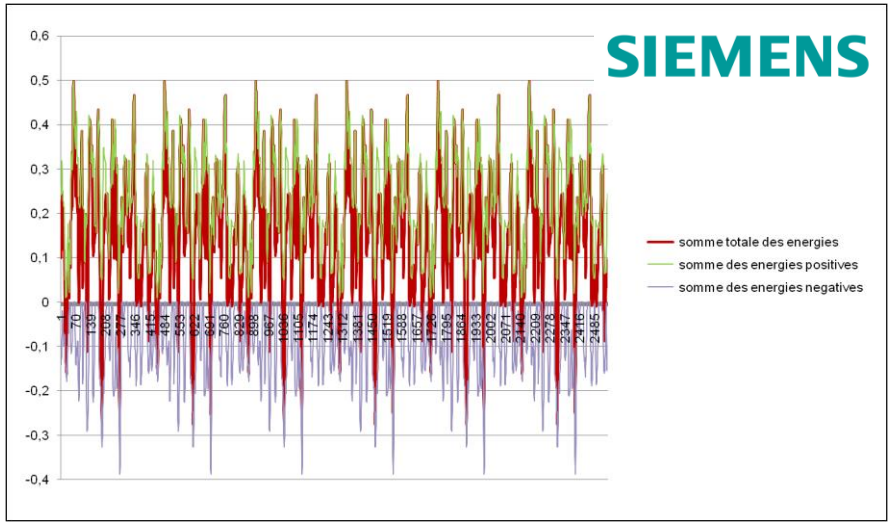

- PLNE: 6 jours de travail (modèle + décomposition objectifs)
- Recherche locale spécifique : 18 jours de travail (algorithme)
- LocalSolver : 3 jours de travail (modèle)

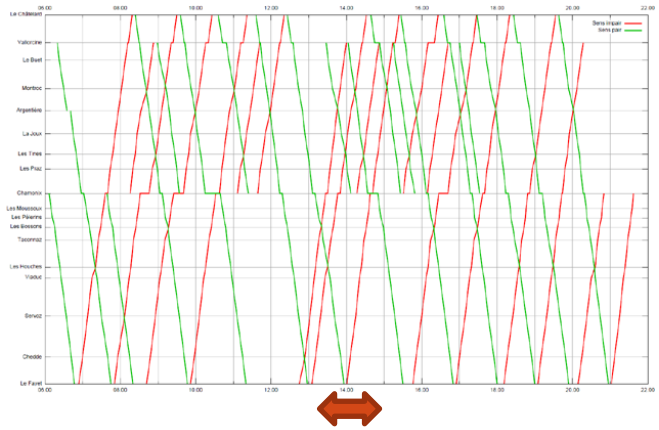
**Solutions quasi optimales en moins de 60 secondes**
**4 millions de mouvements (= solutions visitées) par minute**

LocalSolver en exploitation depuis deux ans

# Optimisation énergétique d'une ligne de métro

EURODECISION
OPERATIONAL RESEARCH



On ajuste les temps d'arrêt pour minimiser l'énergie de freinage électrique perdue.
5000 variables binaires : une par date de départ possible de chaque gare (granularité en secondes).

Comparaison de PLNE vs LocalSolver sur ce modèle (distance à l'optimum en %)

| | LocalSolver 2.0 | PLNE |
|---|---|---|
| 10 secondes | 5 % | 10 % |
| 1 minute | 1 % | 4 % |
| 20 minutes | 0 % | 1 % |
| 2 heures | 0 % | 0 % |

LocalSolver

# Eclairage public: planification de maintenance

## « Eclairer mieux avec moins d'énergie »

- Pour répondre à un appel d'offre il faut  planifier le remplacement des luminaires, la maintenance, l'entretien …
- Il faut tenir compte des coûts, des ressources, de la consommation électrique, des taux de panne

## Des problèmes d'optimisation de grande taille

- Des milliers de rues
- Des dizaines de milliers de points lumineux

Un modèle LocalSolver est désormais utilisé pour chaque réponse à ces appels d'offres

# Intéressés ?

Prenez en main LocalSolver

# Une communauté en pleine croissance













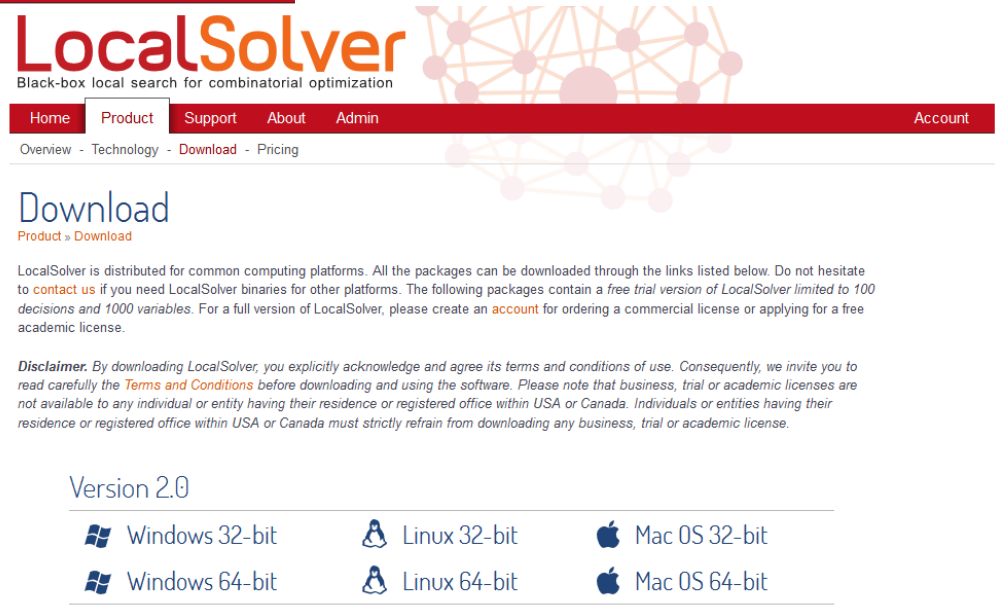






LocalSolver

# Pour démarrer: www.localsolver.com

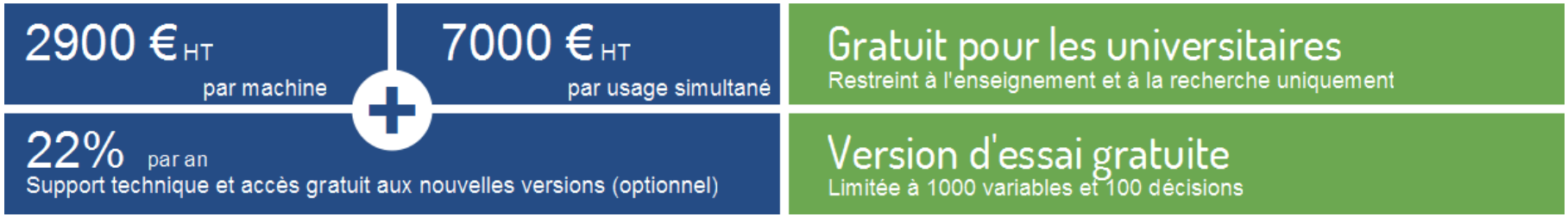




## Version d'essai

(100 décisions, 1000 expressions max)

## Téléchargeable gratuitement

# Offre de licences




## Le prix inclut toutes les fonctionnalités

Solveur, modeleur, APIs, multithreading

## Pas de limitation sur la puissance de l'ordinateur

Simple cœur / multi-cœurs / serveur

## Pas de limitation sur le type d'usage

Développement, déploiement

## La maintenance donne accès gratuitement aux nouvelles versions

# Offre de service

## Starting Pack: 11 000€ HT

- Une licence pour une machine et un usage simultané
- Une journée de conseil d'un expert LocalSolver pour modéliser un problème concret d'optimisation auquel vous êtes confronté
- Nous vous livrons un modèle (fichier LSP)
- Sans engagement: vous pouvez arrêter à l'issue de cette session de modélisation (seule la prestation forfaitaire de 2000€ sera due)

# Offre de service

## Starting Pack: ~~11 000€ HT~~ 9900€ HT jusqu'au 01/09/2012

- Une licence pour une machine et un usage simultané
- Une journée de conseil d'un expert LocalSolver pour modéliser un problème concret d'optimisation auquel vous êtes confronté
- Nous vous livrons un modèle (fichier LSP)
- Sans engagement: vous pouvez arrêter à l'issue de cette session de modélisation (seule la prestation forfaitaire de 2000€ sera due)

## Solution complète

- Vous pouvez nous confier tout ou partie de votre projet: spécifications, modélisation, développement, intégration…
- Contactez nous

# Votre premier modèle LocalSolver

1. Choisissez un de vos problèmes ou un nouveau problème

2. Téléchargez LocalSolver

3. Modélisez, résolvez : vous ne serez pas déçus !
   En cas de besoin, nous sommes à votre service
   → contact@localsolver.com


www.localsolver.com