Vers une programmation par recherche locale : LocalSolver 1.1

Thierry Benoist **Frédéric Gardi** **Romain Megel**
Bouygues e-lab, Paris
{ tbenoist, fgardi, rmegel }@bouygues.com

Bertrand Estellon **Karim Nouioua**
Laboratoire d'Informatique Fondamentale - CNRS UMR 6166,
Faculté des Sciences de Luminy - Université Aix-Marseille, Marseille
{ bertrand.estellon, karim.nouioua }@lif.univ-mrs.fr

Deux constats

1) Quel est l'outil le plus puissant de la RO ?

Programmation Linéaire en Nombres Entiers (PLNE) :

- Formalisme simple et générique
- Solveurs simples d'utilisation : approche « *model & run* »

Outil incontournable pour les praticiens.
Programmation par Contraintes (PPC) suit cette voie.

2) Que font les praticiens lorsque PLNE et PPC s'avèrent inefficaces (problèmes très combinatoires et/ou de grande taille) ?

Recherche Locale : permet d'obtenir en quelques minutes des solutions de qualité. Mais cela induit un surcoût (développement, maintenance).

Une conviction

Les approches par « pure » recherche arborescente resteront limitées face aux problèmes combinatoires de grande taille. Pourquoi ?

1) Souvent, la relaxation n'apporte rien mais coûte beaucoup en temps. Alors pourquoi perdre du temps à énumérer des solutions partielles ?

2) Pourquoi une recherche arborescente *incomplète* serait-elle meilleure qu'une recherche locale ? Il est même difficile d'explorer de façon randomisée (non biaisée) par recherche arborescente.

Une preuve : les solveurs récents intègrent de plus en plus des éléments de « recherche locale » dans le parcours de l'arbre de *Branch & Bound*.

Une idée

Si seule la Recherche Locale permet de passer à l'échelle, alors ne faut-il pas envisager un solveur fondé sur la Recherche Locale ?

Quelques travaux dans ce sens :

- Côté SAT : les meilleurs prouveurs SAT et Pseudo-Booléen sont basés sur des techniques de recherche locale (Walksat).

- Côté PPC : Comet CBLS et iOpt sont des frameworks visant à faciliter l'implémentation d'heuristiques de recherche locale (avec une évaluation incrémentale automatique).

Mais à ce jour, qui connaît (et utilise) un solveur « boite noire » à base de recherche locale pour l'optimisation combinatoire ?

Un projet

2007 : Démarrage du projet LocalSolver

Objectifs à long terme :

1) Définir un formalisme déclaratif simple et générique permettant une « programmation par recherche locale » (*model*)

2) Développer un solveur efficace exploitant ce formalisme avec comme principe fondamental « faire ce qu'un praticien expert ferait » (*run*)

2009 : Première concrétisation : le logiciel LocalSolver 1.0

- Adapté à la programmation 0-1, en particulier pour les problème de type *assignment*, *partitioning*, *packing*, *covering*.

- Binaires distribués gratuitement (BSD) pour les systèmes Windows, Linux, Mac OS X sur architecture x86.

LocalSolver 1.x : formalisme

Programmation 0-1 généralisée

1) Opérateurs mathématiques pour définir contraintes et objectifs :

- arithmétiques : *sum*, *min*, *max, product, divide, modulo, abs, distance*
- logiques : *and*, *or*, *xor*, *not*, *if-then-else*
- relationnels : ≤, <, =, >, ≥, ≠

→ Permet de traiter des programmes 0-1 fortement non linéaires

2) Objectifs multiples à optimiser dans l'ordre lexicographique.

→ Facilite le *goal programming* : Minimize *x* ; Maximize *y* ; Minimize *z* ;

Attention : modélisation = définition de l'espace de recherche

La Recherche Locale n'aime les espaces trop contraints

LocalSolver 1.1 : nouveautés

1) Opérateurs mathématiques enrichis : divide, modulo, abs, distance

2) Mouvements autonomes enrichis :

- Pool élargi : k-Flips, k-Paths, k-Cycles, k-Moves, k-Balances
- Mouvements randomisés (diversification) ou ciblés (intensification)

3) Stratégies de recherche enrichis :

- Métaheuristique du recuit simulé en sus de la descente standard
- Multithreading de la recherche

→ Côté modeleur : plus d'expressivité

→ Côté solveur : convergence plus rapide et plus robuste

LocalSolver 1.1 : benchmarks

Steel mill slab design (CSPLIB) : **6 millions de mouvements par minute**

60 sec	2-0	3-0	4-0	5-0	6-0	7-0	8-0	9-0	10-0
Etat de l'art	22	5	32	0	0	0	0	0	0
LocalSolver 1.0	46	52	35	4	8	2	0	0	0
LocalSolver 1.1	**37**	**8**	**35**	**1**	**4**	**1**	**0**	**0**	**0**
CPLEX 12.2	136	288	X	126	X	232	226	163	133
CPO 2.3	90	65	58	50	54	46	28	29	20
Comet CBLS 2.1	136	135	69	65	42	30	26	21	20

600 sec	2-0	3-0	4-0	5-0	6-0	7-0	8-0	9-0	10-0
Etat de l'art	22	5	32	0	0	0	0	0	0
LocalSolver 1.1	**31**	**7**	**34**	**0**	**4**	**0**	**0**	**0**	**0**
CPLEX 12.2	94	65	X	63	X	189	226	97	64

LocalSolver 1.1 : applications

TF1 Publicité : affectation des emplacements préférentiels

- Problème d'affectation avec contraintes non linéaires (max)
- 5 objectifs lexicographiques
- 10^4 variables binaires

- Gurobi : 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 (hebdomadaire) depuis 6 mois

LocalSolver 1.1 : applications

Autres applications chez Bouygues :

1) TF1 (1001 Mariages) : composition des tables de mariage
2) By SA (IMB) : planification des séminaires de formation

En cours :

3) By Construction (ETDE) : planification de maintenance des luminaires
4) Colas : planification de maintenance des routes
5) By Construction (Habitat Social) : optimisation des stocks de banches

PLNE : impuissante sur les problèmes 1, 2, 4

LocalSolver : solutions quasi optimales en moins de 60 secondes

LocalSolver : feuille de route

LocalSolver 2.0 (2012) : un **modeleur** pour LocalSolver

```
// data

nbObjets = ...; nbBoites = ...;
tailleObjets[1..nbObjets] = ...; tailleBoites[1..nbBoites] = ...;

// model

x[1..nbObjets][1..nbBoites] <- bool(); // variables de décision

for[i in 1..nbObjets] // contraintes
   constraint boolsum[j in 1..nbBoites]( x[i][j] ) == 1;

for[j in 1..nbBoites] { // variables intermédiaires
   rempliBoites[j] <- sum[i in 1..nbObjets]( tailleObjets[i] x[i][j] );
   constraint rempliBoites[j] <= tailleBoites[j];
   residuBoites[j] <- if( rempliBoites[j] > 0,
                          sum( tailleBoites[j], -rempliBoites[j] ), 0 );
}

minimize sum[j in 1..nbBoites]( residuBoites[j] ); // objectif
```

LocalSolver : feuille de route

Moyen terme : **programmation en nombres entiers**

Modéliser en nombres entiers en sus du 0-1.

Moyen terme : **opérateurs ensemblistes + structures discrètes**

Modéliser à l'aide de structures discrètes : modélisation de plus haut niveau, plus proche du problème métier ; résolution plus performante.

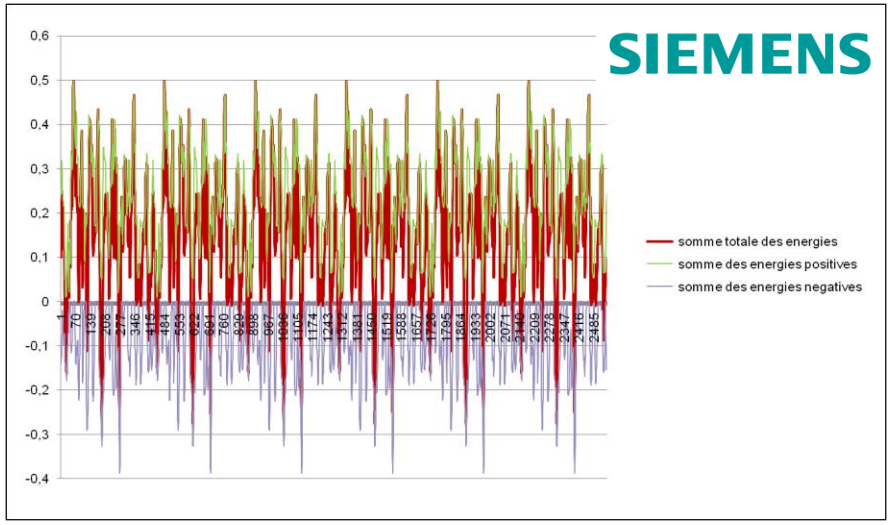
Attaquer les problèmes d'ordonnancement et de routage.

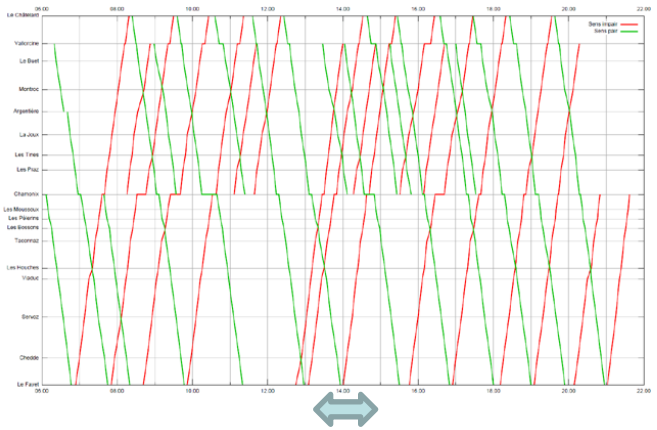
Long terme : **intégration d'un solveur linéaire « incrémental »**

Attaquer des programmes en variables mixtes.

→ Elargir le champ de la modélisation (bas/haut niveau, discret/continu) tout en maintenant l'efficacité du solveur

Optimisation énergétique d'une ligne de métro





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 CPLEX vs LocalSolver sur ce modèle (distance à l'optimum en %)

	LocalSolver 1.1	CPLEX 12.2
10 secondes	5 %	10 %
1 minute	1 %	4 %
20 minutes	0 %	1 %
2 heures	0 %	0 %

EURODECISION
OPERATIONAL RESEARCH