BOUYGUES INNOVATION & OPTIMISATION

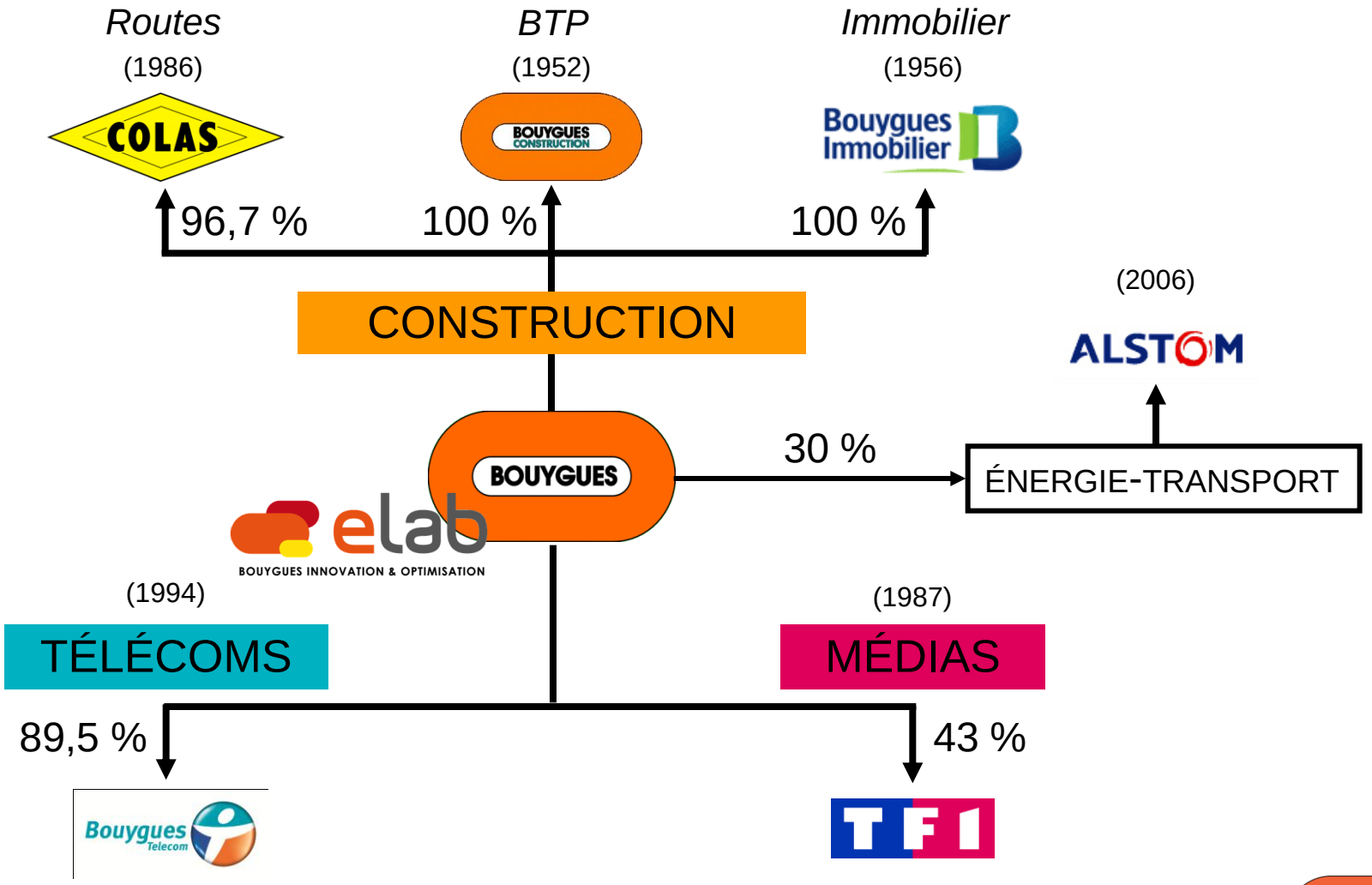
BOUYGUES

La recherche opérationnelle chez Bouygues

Frédéric GARDI

fgardi@bouygues.com

Groupe Bouygues



Bouygues e-lab

- R&D en informatique avancée (RO + NTIC)
- Basé à Bouygues SA au service des métiers
- 11 ingénieurs permanents (dont 4 docteurs)
- 3 missions :
  - Projets pour les filiales
  - Expertise & Recherche
  - Animation de l'innovation

Plan général

- Notre vision de la RO en entreprise,
  illustrée chez TF1 Publicité
- Recherche locale : méthodologie et applications
- La recherche au Bouygues e-lab : LocalSolver,
  solveur de « programmation par recherche locale »

BOUYGUES

Notre vision de la recherche opérationnelle en entreprise

Etudes

- Prévisions d'audiences : comportement des téléspectateurs
- Prévisions d'achat d'espace publicitaire : comportements des annonceurs (réservation, annulation)
- Tarification des écrans publicitaires de TF1 (économétrie, théorie des jeux)
- Etude scientifique au sens large. Exemple : évaluer l'assiduité = probabilité qu'un téléspectateur voir un spot publicitaire entièrement.

Logiciels

- Optimisation du remplissage des écrans publicitaires à l'ouverture de planning sur la chaîne TF1 (150 jours)
- Programmation de campagnes publicitaires sur les chaînes thématiques du Groupe TF1 (70 jours)
- Programmation des emplacements préférentiels sur la chaîne TF1 (30 jours)
- Prévision d'audiences pour les sites internet et les radios du Groupe TF1 (20 + 10 jours)

Programmation des chaînes thématiques

Une chaîne

Un budget à programmer

Une répartition de l'audience sur la semaine

LES PLANS Modulo

Tarifs valables du 1er au 30 septembre 2008
Tarif Brut base 30 secondes

Pour tout budget à partir de 6 200 € brut base 30"	OPTIM DISNEY	SELECT DISNEY
% des GRP programmés en PRIME (1)	78%	100%
CGRP* Brut Enfants 4-10 ans (Cible de Référence)	950 €	1 100 €
Nombre de messages hebdomadaire indicatif pour une campagne de 6 200 € brut base 30"	35-39	25 à 29

(1) PRIME Disney Channel: Semaine [0645;0830] et [1630;2100], Mercredi et Week End

Un coût garanti par point d'audience

LES TARIFS UNITAIRES

Tarifs valables du 1er au 30 septembre 2008. Tarif Brut base 30 secondes

Ecrans	Lundi	Mardi	Mercredi	Jeudi	Vendredi	Samedi	Dimanche
0625	225 €	225 €	225 €	225 €	225 €	225 €	225 €
0645	225 €	225 €	225 €	225 €	225 €	225 €	225 €
0715	225 €	225 €	225 €	225 €	225 €	225 €	225 €
0730	225 €	225 €	225 €	225 €	225 €	225 €	225 €
0800	225 €	225 €	225 €	225 €	225 €	225 €	225 €
0820	225 €	225 €	225 €	225 €	225 €	225 €	225 €
0840	160 €	160 €	640 €	160 €	160 €	390 €	390 €
0905	160 €	160 €	640 €	160 €	160 €	390 €	390 €

Méthodologie

Prestation payante : modèle « SSII interne ». De fait, les budgets alloués sont fonction des enjeux économiques.

Projet en mode agile, par itération, proche du client final :

- Cadrage initial du besoin
- Livraison Alpha : intégration informatique
- Livraison Beta : validation fonctionnelle
- Livraison 1.0 : prêt pour l'exploitation

Cadrage initial

- Cadrage du besoin avec le client final -> spécification du modèle d'optimisation
- Ne pas se faire déborder par le client : tendance naturelle à gonfler les besoins réels
- Bien analyser : échelle du problème, fréquence d'utilisation du logiciel, enjeux économiques sous-jacents → détermine le choix de la technique d'optimisation

Version Alpha

- Spécification des interfaces de programmation avec l'informatique (API)
- Coller à l'architecture du client : éviter les couplages exotiques (natif/managé, .NET/Java)
- Version Alpha = « Hello World », permet de valider les interfaces

Version Beta

- Vérificateurs d'entrée et de sortie. Solution basique (contraintes ok). Fourniture des données par le client.
- Permet de vérifier la viabilité du modèle (contraintes trop fortes, contraintes manquantes). Trouver une solution valide doit être facile, sinon problème mal posé.
- Version Beta = algorithme glouton, permet de valider le modèle et de rassurer le client sur la faisabilité du logiciel

Version 1.0

- Solution optimisée. Qualité (fiabilité, robustesse) : recette interne, benchmarks. Profilage du code (CPU & RAM).
- À éviter : construction logicielle complexe (coopération de solveurs), cuisine algorithmique (hybridation de techniques), généricité injustifiée, optimisation de code prématurée.
- Qualité cruciale : rien ne sert d'optimiser si nos solutions sont fausses !
- Version 1.0 = version mise en production

Optimisation

Pas de religion algorithmique :

- Optimisation du remplissage des écrans publicitaires sur la chaîne TF1 : programmation dynamique
- Programmation de campagnes publicitaires sur les chaînes thématiques du Groupe TF1 : PLNE
- Programmation des emplacements préférentiels sur la chaîne TF1 : recherche locale
- Construction des cycles de travail des techniciens de maintenance : PPC

Optimisation

Choix de la technique de résolution :

- Problèmes de petite taille : PLNE ou PPC. Nous évitons les solveurs commerciaux (surcoûts).
- Problèmes de grande taille (y compris en variables mixtes) : recherche locale.
- Si enjeux économiques : bornes inférieures.

→ Industrialiser notre production logicielle : réduire les risques, réduire les coûts.

Documentation

- Pas de paperasse inutile. Spécifications légères, adaptées aux besoins, aux enjeux.
- Côté technique : petit manuel d'installation. Fournir la liste des composants nécessaires (versions, etc.).
- Côté fonctionnel : petit manuel d'utilisation, tutoriel de démarrage rapide. Idéal : une feuille A4.
- Documentation e-lab : pas de paperasse ou commentaire de code inutile. Le code fait foi : lire le livre « Coder proprement ». Répertoire projet structuré, normé, versionné (SVN).

Cadre juridique

- Lotissement du projet (avec possibilité d'arrêt après chaque lot). Coûts et délais de chaque lot.
- Dates de livraison flottantes : si une phase décalée, alors la suite est décalée. Vigilance : fourniture des données impérative en version Beta.
- Qu'est ce qu'une anomalie ?

- Majeure : pas de solution ou solution non admissible (échec au vérificateur de sortie).

- Mineure : solution pas viable ou sous-optimale (améliorable par le client).

BOUYGUES

Points clés

- Prestations rémunérées et cadre juridique
- Etudes et logiciels opérationnels
- Communication (interne et externe)

- Versions Alpha et Beta cruciales (au plus tôt)
- Intégration & Données -> vigilance !
- Pas de religion algorithmique
- Simplicité = maintenabilité

- Un credo : **« La RO, c'est 80 % d'informatique »**

Recherche Locale

BOUYGUES

Recherche Locale

Les solveurs à base de recherche arborescente (PLNE, PPC) s'avèrent impuissants face aux problèmes difficiles de grande taille.

Recherche locale :

- Exploration incomplète et non déterministe de l'espace des solutions
- Permet de fournir des solutions de qualité en des temps très courts (quelques minutes)
- Solution de choix pour attaquer les problèmes de grande échelle (millions voire milliards de variables booléennes)

Méthodologie

Contrairement à l'idée répandue « recherche locale = métaheuristiques », nous prônons une approche :

- **Pure** et **directe** : pas de décomposition, pas d'hybridation
- **Agressive** : des millions de solutions admissibles visitées

Deux points clés :

- Diversifier l'exploration en concevant une **large variété de mouvements randomisés** dédiés aux instances
- Accélérer l'exploration en exploitant les invariants lors de l'évaluation des mouvements → **algorithmique incrémentale**

Méthodologie

Effort de qualité redoublé → ne pas laisser se propager des anomalies au long de la recherche (infernal à déboguer) :

- Programmation par assertions (en entrée/sortie de chaque fonction, activées en mode *Debug*)
- Vérification exhaustive des structures de données incrémentales à chaque itération en mode *Debug*

Quête de la haute performance :

- Profilage CPU & RAM
- Optimisation des caches L2/L1 (principe de localité)

Projets chez Bouygues

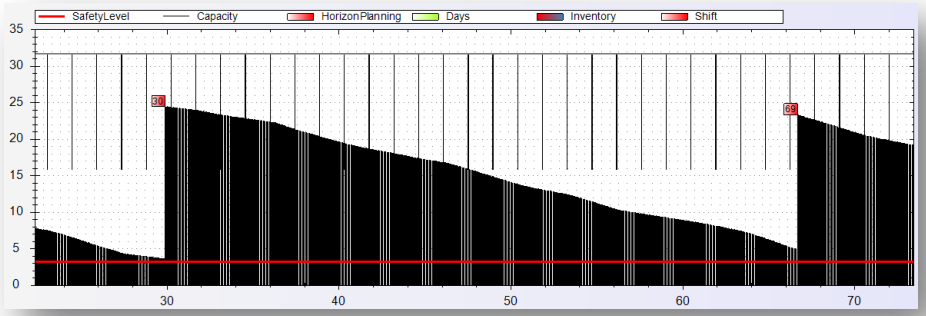
- Programmation des emplacements publicitaires préférentiels (TF1 Publicité)
- Ordonnancement des mouvements de terre sur un chantier linéaire de terrassement (DTP Terrassement)
- Construction des planning des séminaires de formation (Bouygues SA)
- Optimisation de tournées de véhicules avec gestion des stocks (Air Liquide)

Projets hors Bouygues

- Ordonnancement de véhicules dans les ateliers peinture et assemblage (Renault, Challenge ROADEF 2005)
- Planification des interventions de techniciens de maintenance (France Telecom, Challenge ROADEF 2007)
- Planification de la maintenance et de la production des centrales thermiques françaises (EDF, Challenge ROADEF/EURO 2010)

Inventory Routing

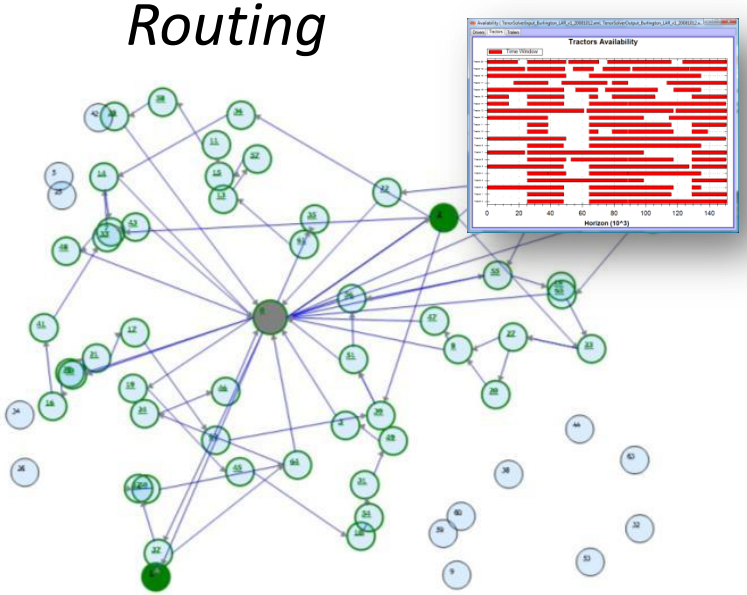
Inventory



Capacité de stockage finie
Consommation non linéaire
Niveaux de sécurité à respecter
Stocks clients + usines
Fenêtres d'ouverture des sites
Accès limités à certains véhicules

SOLUTION = un ensemble de vacations
- **départ et retour à un dépôt**
- **séquence d'opérations (date, site, quantité)**
- **3 ressources : chauffeur, tracteur, remorque**

Routing



Dépôts, usines, clients
Ressources non homogènes
Fenêtres de disponibilité
Matrices de compatibilité
Réglementation du travail
Longues vacations possibles
Matrices de distances et durées

BOUYGUES

Inventory Routing

Vendor Managed Inventory : stocks des clients gérés par le fournisseur
→ minimiser les coûts de réapprovisionnement à long terme

Mais à court terme : Qui livrer ? Quand ? Combien ? Comment ?

Objectifs :

1) Minimiser le nombre de commandes insatisfaites.

2) Minimiser le nombre de violations des seuils de sécurité.

3) Minimiser le ratio logistique : somme des coûts des tournées divisée par somme des quantités livrées.

En pratique, les deux premiers objectifs doivent être (facilement) atteints. En effet, toute commande insatisfaite ou pénurie n'est pas acceptable pour un client.

Inventory Routing

NP-difficile : IRP → VRP → TSP

Échelle des instances en pratique :

- horizon de planification de 15 jours
- des centaines de clients (jusqu'à 1500)
- des dizaines d'usines (jusqu'à 50)
- des dizaines de dépôts (jusqu'à 50)
- des dizaines de ressources (jusqu'à 50 par type)
- temps continue (précision à la minute : 21600 minutes)
- consommations/productions à l'heure (360 pas de temps)

Temps de résolution limité à 5 minutes sur un ordinateur standard

Gain minimum attendu sur solutions métier : 8 % en moyenne

BOUYGUES

Inventory Routing

Heuristique générale :

- solution initiale construite par glouton (basé sur l'urgence des clients)
- *first-improvement descent*
- choix stochastique (mais pas uniforme) des mouvements

Mouvements dérivant de transformations canoniques :

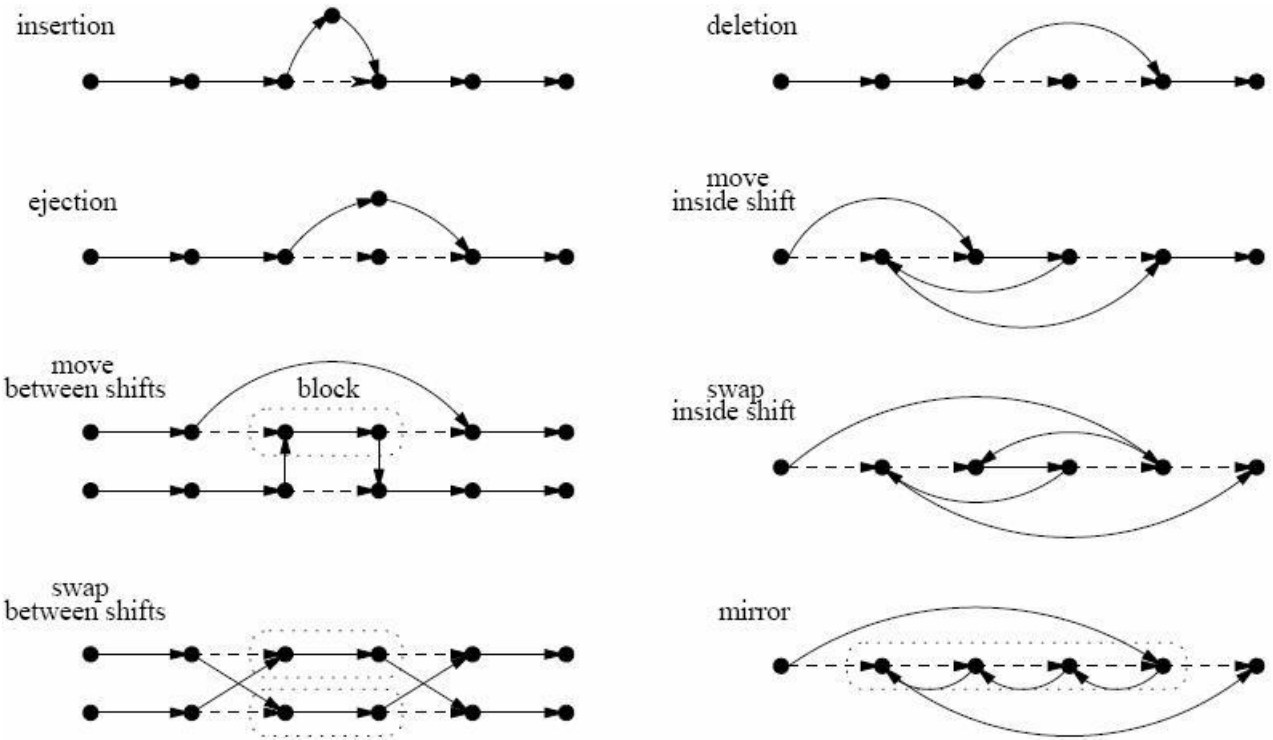
- sur opérations : insertion, suppression, éjection, déplacement intra/inter tournées, échange intra/inter tournées, inversion

- sur tournées : insertion, suppression, décalage temporel, réaffectation, échange, fusion, séparation

→ voisinage de taille $O(n^2)$ avec n le nombre d'opérations dans la solution courante. Mais la constante cachée par le O est large !

Inventory Routing

Transformations sur les opérations :



Inventory Routing

1) (Ré)ordonnancement d'une tournée de k opérations : minimiser le temps improductif sur la tournée → glouton chronologique

- prendre la pause le plus tard possible et convertir les temps d'attente devant les sites fermés en pause

- en temps $O(k)$ si les pauses ne sont pas stockés explicitement

- ordonnancement en avant ou en arrière rendu symétrique (tournée = liste d'opérations doublement chaînées)

- optimal si aucun temps d'attente n'apparaît sur la tournée (cas fréquent en pratique)

Inventory Routing

2) (Ré)affectation des volumes sur n opérations : maximiser le volume total livré → critique pour la performance

- Résolution exacte par flot maximum → temps $O(n^3)$

TROP LENT !

- Glouton poussant un maximum de flot à chaque nœud du DAG suivant un ordre topologique (= chronologique ici) → temps $O(n \log n)$

Propriété (intéressante en pratique) : optimal si chaque client est livré une seule fois sur l'horizon de planification

HUM…

- Même glouton mais avec réaffectation partielle (= locale) des volumes dans réseau → temps $O(n' \log n')$ avec $n' << n$

Inventory Routing

Difficulté : réaffecter la quantité livrée à un client à l'instant *t* alors que d'autres opérations ont lieu après *t* → risque de pénurie ou de dépassement de capacité après *t*.

→ mise en œuvre incrémentale très délicate : les réseaux avant et après transformation sont « superposés » dans la même structure de donnée.

Récompense : le glouton partiel s'exécute en temps quasiment constant en pratique.

T(glouton partiel) = T(glouton complet) / 100
T(glouton partiel) = T(flot maximum) / 2000

Écart moyen à la réaffectation optimale : 2 % (si *MO* = 0 et *SO* = 0)

BANCO !

Inventory Routing

- Programmé en C# 2.0 (pour machine virtuelle .NET 2.0)
- Environ 30 000 lignes de code dont 6 000 (20 %) de *checkers*
- Projet global : 300 jours-homme

- Plus de 10 000 mouvements par seconde
- Près de 10 millions de solutions visitées en 5 minutes
- 30 Mo de mémoire alloué, 300 Mo pour les très grandes instances

- Diversification naturelle à coût constant pour les 3 objectifs
- Taux d'acceptation des mouvements entre 1 et 10 %
- Un millier de mouvements strictement améliorant

- Gain moyen de 21 % sur notre algorithme glouton
- Gain moyen de 25 % sur les experts logistiques

Black-box local search for combinatorial optimization

elab
BOUYGUES INNOVATION & OPTIMISATION

Frédéric GARDI
Thierry BENOIST

Bertrand ESTELLON
Karim NOUIOUA

+ 2 jeunes recrues : Romain MEGEL, Julien DARLAY

BOUYGUES

Introduction

LocalSolver

Comment résoudre de façon générique des problèmes d'optimisation combinatoire ?

1) Techniques de recherche arborescente :

PLNE : un des outils les plus puissants de la RO. Pourquoi ?

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

Logiciels les plus utilisés : CPLEX, Xpress, Gurobi, GLPK, CBC

PPC : suit le même chemin… (ex : ILOG CP Optimizer)

Introduction

LocalSolver

Comment résoudre de façon générique des problèmes d'optimisation combinatoire ?

2) Techniques de recherche locale :

Permet d'obtenir des solutions de qualité en temps limité (minutes). Solution la plus performante face aux problèmes réels de grande taille (voir les challenges ROADEF).

Malheureusement, implémenter une recherche locale n'est pas si facile. Répartition du temps observée sur un projet :

- stratégie de recherche = 10 %
- mouvements = 30 %
- algorithmique incrémentale d'évaluation = 60 %

BOUYGUES

État de l'art

LocalSolver

Plusieurs librairies proposées pour faciliter l'implémentation de la couche « stratégie de recherche » (ex : ParadisEO, EasyLocal++).

Deux logiciels pionniers facilitant l'implémentation des couches basses :

- Comet (Van Hentenryck & Michel) : langage de type PPC
- iOpt (British Telecom) : librairie Java

Aucun solveur d'optimisation combinatoire de type « *model & run* » basé sur la recherche locale, comme on en connaît en PLNE et PPC.

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

Projet LocalSolver

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

- Permet de traiter une classe restreinte mais importante de problèmes d'optimisation combinatoire : *assignment*, *partitioning*, *packing*, *covering*.

- Binaires sous licence BSD disponibles sur demande pour les systèmes Windows, Linux, Mac OS X sur architecture x86.

Mars 2011 : LocalSolver 1.1

Février 2012 : LocalSolver 2.0 → version commerciale !

BOUYGUES

Formalisme

LocalSolver

Modélisation en variables booléennes, proche de la PLNE, mais…

1) Offrant une gamme enrichie d'opérateurs mathématiques pour définir contraintes et objectifs:

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

→ Permet de modéliser des problèmes fortement non linéaires

2) Permet de définir des objectifs multiples à optimiser dans l'ordre lexicographique.

→ Facilite le *goal programming* : Minimize 1000000 x - 1000 y + z

Attention : modéliser = définir l'espace de recherche
Trop contraindre cet espace perturbera la recherche locale

BOUYGUES

Formalisme

LocalSolver

Modélisation en variables booléennes, proche de la PLNE, mais…

1) Offrant une gamme enrichie d'opérateurs mathématiques pour définir contraintes et objectifs:

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

→ Permet de modéliser des problèmes fortement non linéaires

2) Permet de définir des objectifs multiples à optimiser dans l'ordre lexicographique.

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

Attention : modéliser = définir l'espace de recherche
Trop contraindre cet espace perturbera la recherche locale

BOUYGUES

Formalisme

LocalSolver

Un petit problème de *bin-packing* écrit dans le format LSP : 3 objets *x*, *y*, *z* à empiler dans 2 boîtes *A*, *B* de façon à minimiser la hauteur de la pile la plus grande :

```
xA <- bool(); yA <- bool(); zA <- bool();
xB <- bool(); yB <- bool(); zB <- bool();
constraint sum(xA, xB) = 1;
Constraint sum(xA, xB) = 1;
Constraint sum(xA, xB) = 1;
heightA <- sum(2xA, 3yA, 4zA);
heightB <- sum(2xB, 3yB, 4zB, 5);
minimize heightMax <- max(heightA, heightB);
```

BOUYGUES

Formalisme

LocalSolver

Un petit problème de *bin-packing* écrit dans le format LSP : 3 objets *x*, *y*, *z* à empiler dans 2 boîtes *A*, *B* de façon à minimiser la hauteur de la pile la plus grande :

```
xA <- bool(); yA <- bool(); zA <- bool();
xB <- bool(); yB <- bool(); zB <- bool();
constraint sum(xA, xB) = 1;
Constraint sum(xA, xB) = 1;
Constraint sum(xA, xB) = 1;
heightA <- sum(2xA, 3yA, 4zA);
heightB <- sum(2xB, 3yB, 4zB, 5);
minimize heightMax <- max(heightA, heightB);
```

Si vous souhaitez maximiser la hauteur de la pile la plus petite comme second objectif, ajoutez simplement la ligne suivante :

```
maximize heightMin <- min(heightA, heightB);
```

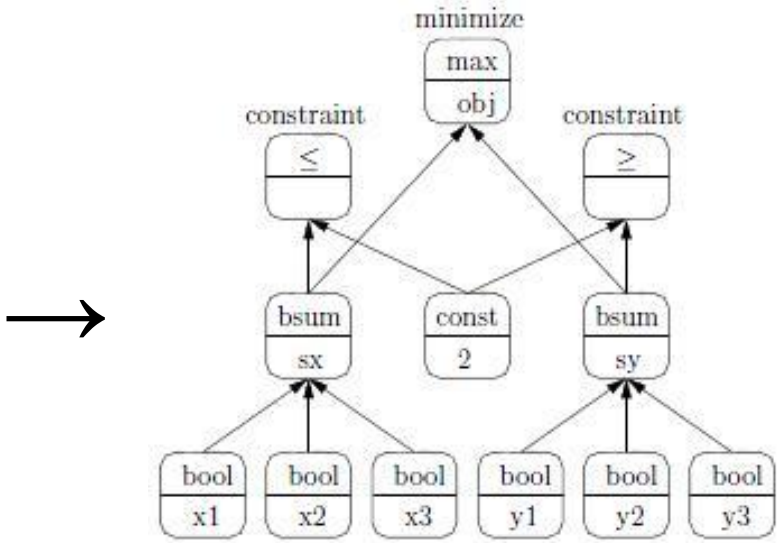
BOUYGUES

Solveur

LocalSolver

Représentation du LSP par un DAG :

```
x1 <- bool(); x2 <- bool(); x3 <- bool();
x1 <- bool(); y2 <- bool(); y3 <- bool();
sx <- sum(x1, x2, x3);
sy <- sum(y1, y2, y3);
constraint sx <= 2;
constraint sy >= 2;
obj <- max(sx, sy);
minimize obj;
```



Solveur

LocalSolver

Comment ça marche ?

BOUYGUES

Solveur

LocalSolver

Comment ça marche ?

1) Une évaluation hautement optimisée :

Propagation des modifications dans le DAG : *Lazy Breadth-First Search*

Chaque nœud est visité au plus une fois. Un nœud est visité seulement si la modification d'un de ses parents rend sa valeur obsolète.

Ex : Un nœud `x <- a < b` de valeur vraie. Si `a` diminue ou `b` augmente, alors `x` n'est pas visité.

Exploitation des invariants induits par les opérateurs mathématiques

Ex : Un nœud $z \leftarrow or(a_1, \ldots, a_k)$ avec T la liste des a_i de valeur vraie et M la liste des a_i modifiés. Si $|T| \neq |M|$, alors z sera vraie. Dans ce cas, évaluation en temps $O(1)$.

BOUYGUES

Solveur

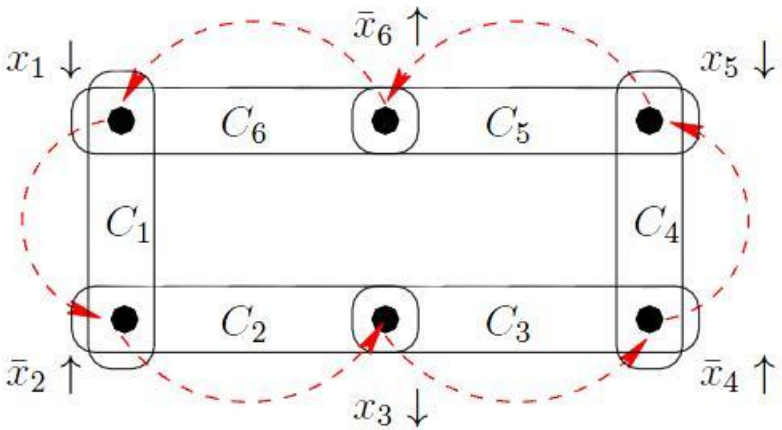
LocalSolver

Comment ça marche ?

2) Des mouvements visant à préserver la faisabilité :

Généralisation des « chaînes d'éjection » dans l'hypergraphe induit par les variables de décision (booléennes) et les contraintes.

Ces mouvements, appelés *k*-Chaînes et *k*-Cycles, simulent en temps $O(k)$ des *k-Moves* et *k-Exchanges* respectivement dans les modèles de type *packing*/*covering*.



BOUYGUES

Benchmarks

LocalSolver

6 problèmes réalistes (choisis en tout début projet) :

Ordonnancement de véhicules (CSPLIB + Renault)
Constitution de groupes de travail de séminaires (CSPLIB + BySA)
Découpage de tôles en aciérie (CSPLIB)
Ordonnancement des prises de vue du satellite Spot 5 (CNES)
Gestion des stocks de matériel sur les chantiers (ByCons)
Puzzle Eternity 2 (Tomy)

Et depuis…

Bin packing (M. Van Caneghem)
Coloration de graphes (M. Van Caneghem)
Emplois de temps universitaires (M. Van Caneghem)
Art digital : portraits à base de dominos (G. Rochart)
Plans de table de mariage (TF1 - 1001 Listes)
Génération automatique des examens du permis de conduire (Eurodecision)
Optimisation énergétique d'une ligne de transport (Eurodecision)
Placement de spots publicitaires télévisuels (TF1 Publicité)
Génération automatique de résumés de réunions (LIF Marseille - TAL)

BOUYGUES

Benchmarks

LocalSolver

Réalisés sur un ordinateur standard : 3 GHz, RAM 2 Go, L2 6 Mio

Car sequencing (CSPLIB) :

60 sec	10-93	200-01	300-01	400-01	500-08
State-of-the-art LS	3	0	0	1	0
LocalSolver 1.1	8	8	8	13	18
CPLEX 12.2	6	11	27	17	X
Comet CBLS 2.1	7	8	16	18	91

600 sec	10-93	200-01	300-01	400-01	500-08
State-of-the-art LS	3	0	0	1	0
LocalSolver 1.1	6	5	4	6	6
CPLEX 12.2	3	3	11	16	104
Comet CBLS 2.1	7	6	10	18	47

Benchmarks

LocalSolver

Car sequencing (RENAULT, ROADEF 2005 Challenge) :

600 sec	X2	X3	X4
State-of-the-art LS	0, 192, 66 (1/19)	0, 337, 6 (1/19)	0, 160, 407 (1/19)
LocalSolver 1.1	0, 268, 212 (16/19)	36, 544, 187 (16/19)	2, 353, 692 (18/19)

À notre connaissance, aucun solveur (PLNE, PPC, SAT, etc.) n'est capable de trouver une solution admissible à ces instances dans le temps imparti.

X2 : 1260 véhicules,12 options, 13 couleurs
LSP : 516 936 variables, dont 374 596 booléens
LocalSolver : 3 M mouvements par minute, 450 Mo RAM

Benchmarks

LocalSolver

Steel mill slab design (CSPLIB) :

60 sec	2-0	3-0	4-0	5-0	6-0	7-0	8-0	9-0	10-0
Optimum (BCP)	22	5	32	0	0	0	0	0	0
State-of-the-art LS	28	6	34	0	0	0	0	0	0
LocalSolver 1.1	31	5	35	0	3	1	0	0	0
CPLEX 12.2	178	511	X	X	X	275	226	229	201
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

BOUYGUES

Feuille de route

LocalSolver

LocalSolver : black-box local search for combinatorial optimization

www.localsolver.com

LocalSolver 2.0 :

- Sortie début février 2012
- Désormais payant pour toute utilisation commerciale
- Gratuit pour l'enseignement et la recherche à but non lucratif

BOUYGUES

Feuille de route

LocalSolver

LocalSolver : black-box local search for combinatorial optimization

www.localsolver.com

LocalSolver 2.0 :

- Programmation 0-1 généralisée (non linéaire)

- Un modeleur pour LocalSolver (éxecutable)
- API C++, Java, .NET (librairies)
- Compatible Linux, Mac OS, Windows (x86 et x64)

- Recuit simulé auto-adaptatif + *multithreading*
- Mouvements autonomes renforcés
- Performance « décuplée » : 10 millions de variables 0-1 !

BOUYGUES

Feuille de route

LocalSolver

LocalSolver : black-box local search for combinatorial optimization

www.localsolver.com

Moyen terme : programmation en variables entières généralisée

Introduire des concepts “ensemblistes” dans le formalisme

Long terme : programmation en variables mixtes (entières & continues)

Intégrer un solveur “continu” (comme le simplexe pour la PL)…
Mais à base de recherche locale !

BOUYGUES