LocalSolver

Solving routing and scheduling problems using LocalSolver

Set-Based Modeling in LocalSolver 6.0

www.localsolver.com

Thierry BENOIST

Who we are

Bouygues, one of the French largest corporation, €33 bn in revenues
http://www.bouygues.com

Operations Research subsidiary of Bouygues
20 years of practice and research
http://www.innovation24.fr

Mathematical optimization solver developed by Innovation 24
http://www.localsolver.com

LocalSolver

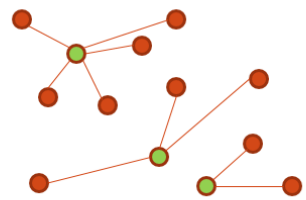
All-terrain optimization solver

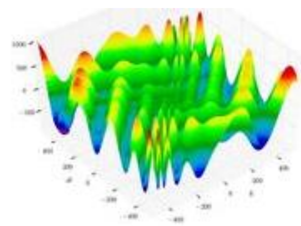
For combinatorial, numerical,
or mixed-variable optimization

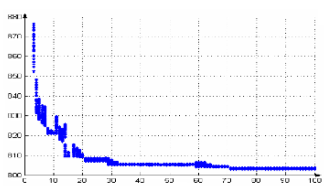
Suited for tackling
large-scale problems

Quality solutions in minutes
without tuning

The « Swiss Army Knife » of mathematical optimization







free trial with support – free for academics – rental licenses from 590 €/month – perpetual licenses from 9,900 €

www.localsolver.com

Clients

- Construction
- Medias & Advertising
- Telco & Retail
- Large Industry
- Energy
- Banking & Finance
- Transportation
- Logistics
- Food & Agribusiness
- Aerospace & Defense
- IT Services

Outline

1. LocalSolver
2. Set-based features for routing
3. Beyond routing

LocalSolver

Quick tour

Features

Better solutions faster

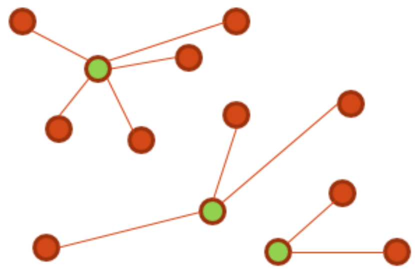
- Provides high-quality solutions quickly (minutes)
- Scalable: able to tackle problems with millions of decisions

Easy to use

- « Model & Run »
    - Rich but simple mathematical modeling formalism
    - Direct resolution: no need of complex tuning

- Innovative modeling language for fast prototyping
- Object-oriented C++, Java, .NET, Python APIs for tight integration
- Fully portable: Windows, Linux, Mac OS (x86, x64)

P-median



Select a subset P among N points minimizing the sum of distances from each point in N to the nearest point in P

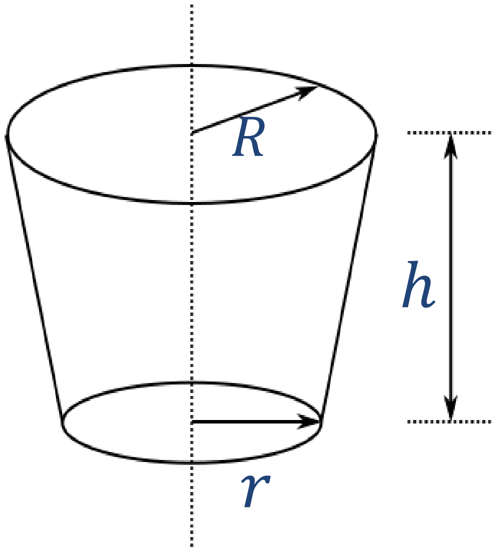
```
function model() {
  x[1..N] <- bool() ; // decisions: point i belongs to P if x[i] = 1
  constraint sum[i in 1..N]( x[i] ) == P ; // constraint: P points selected among N
  minDist[i in 1..N] <- min[j in 1..N]( x[j] ? Dist[i][j] : InfiniteDist ) ; // expressions: distance to the nearest point in P
  minimize sum[i in 1..N]( minDist[i] ) ; // objective: to minimize the sum of distances
}
```

Nothing else to write: “model & run” approach

- Straightforward, natural mathematical model
- Direct resolution: no tuning

Parametric optimization

Maximize the volume of a bucket with a given surface of metal




```
function model() {

  R <- float(0,1);
  r <- float(0,1);
  h <- float(0,1);

  V <- PI * h / 3.0 * (R*R + R*r + r*r);
  S <- PI * r * r + PI*(R+r) * sqrt(pow(R-r,2) + h*h);

  constraint S <= 1;
  maximize V;
}
```

$$V = \frac{\pi h}{3}\left(R^2 + Rr + r^2\right)$$

$$S = \pi r^2 + \pi(R + r)\sqrt{(R - r)^2 + h^2}$$

Mathematical operators

Decisional	Arithmetical			Logical	Relational	Set-related
bool	sum	sub	prod	not	eq	count
float	min	max	abs	and	neq	at
int	div	mod	sqrt	or	geq	indexof
list	log	exp	pow	xor	leq	partition
	cos	sin	tan	iif	gt	disjoint
	floor	ceil	round	array + at	lt	
	dist	scalar		piecewise		

+ operator call : to call an external native function
which can be used to implement your own (black-box) operator

Smart APIs

C++ ISO

Java 5.0

.NET C# 2.0

Python 2.7, 3.2, 3.4

```
########## optimal_bucket.py ##########

import localsolver
import sys

with localsolver.LocalSolver() as ls:

    PI = 3.14159265359

    #
    # Declares the optimization model
    #
    m = ls.model

    R = m.float(0,1)
    r = m.float(0,1)
    h = m.float(0,1)

    # Surface constraint
    # surface = PI * r^2 + PI*(R+r) * sqrt ((R-r)^2 + h^2)
    surface = PI*r*r + PI * m.sqrt((R-r)**2 + h**2) * (R+r)
    m.constraint(surface <= PI)

    # Maximize volume
    # volume = PI * h/3 * (R^2 + R*r + r^2)
    volume = PI * h/3 * (R**2+ R*r + r**2)
    m.maximize(volume)

    m.close()

    #
    # Param
    #
    ls.param.nb_threads = 2
    if len(sys.argv) >= 3: ls.create_phase().time_limit = int(sys.argv[2])
    else: ls.create_phase().time_limit = 6

    ls.solve()
```

Motivations

Modeling approaches for

the *Traveling Salesman Problem*

Mixed-Integer Programming

With an **exponential number** of constraints

Minimise $\sum_{\substack{i,j \\ i\neq j}} c_{ij} x_{ij}$

Conventional Formulation (C) (Dantzig, Fulkerson and Johnson (1954))

$$\sum_{\substack{j \\ j\neq i}} x_{ij} = 1 \qquad \forall\ i \in N$$

$$\sum_{\substack{i \\ i\neq j}} x_{ij} = 1 \qquad \forall\ j \in N$$

$$\sum_{\substack{i,j\in M \\ i\neq j}} x_{ij} \leq |M| - 1 \qquad \forall\ M \subset N \text{ such that } \{1\} \notin M, |M| \geq 2$$

→ Iterative procedure to add subtour elimination constraints

Variant with $O(n^2)$ variables and constraints

SINGLE COMMODITY FLOW (**F1**) (Gavish and Graves (1978))

Both constraints are retained but we also introduce (continuous) variables:

y_{ij} = 'Flow' in an arc (i,j) $i \neq j$

and constraints:

$$y_{ij} \leq (n-1) x_{ij} \qquad \forall\ i,j \in N,\ i \neq j$$

$$\sum_{\substack{j \\ j\neq 1}} y_{1j} = n-1$$

$$\sum_{\substack{i \\ i\neq j}} y_{ij} - \sum_{\substack{k \\ i\neq k}} y_{jk} = 1 \qquad \forall\ j \in N - \{1\}$$

In Orman & Williams : A survey of different integer programming formulations of the TSP

Natural Modeling

As a permutation

The Traveling Salesman Problem (TSP)

TSP: Given a list of cities and their pairwise distances, find a shortest possible tour that visits each city exactly once.

Objective: find a permutation $a_1,\ldots,a_n$ of the cities that minimizes

$$d(a_1, a_2) + d(a_2, a_3) + \ldots + d(a_{n-1}, a_n) + d(a_n, a_1)$$

where $d(i, j)$ is the distance between cities i and j

An optimal TSP tour through Germany's 15 largest cities

In Kenneth R. Rosen: Permutations and Combinations.

Reference modeling

Garey & Johnson

[ND22] **TRAVELING SALESMAN**

INSTANCE: Set C of m cities, distance $d(c_i, c_j) \in Z^+$ for each pair of cities $c_i, c_j \in C$, positive integer B.

QUESTION: Is there a tour of C having length B or less, i.e., a permutation $< c_{\pi(1)}, c_{\pi(2)}, \ldots, c_{\pi(m)} >$ of C such that

$$\left(\sum_{i=1}^{m-1} d(c_{\pi(i)}, c_{\pi(i+1)}) \right) + d(c_{\pi(m)}, c_{\pi(1)}) \leqslant B \ \ ?$$

Set-based modeling

Innovative modeling concepts
for routing & scheduling problems

List Variables

Structured decisional operator list(n)

- Order a subset of values in domain {0, …, n-1}
- Each value is unique in the list

Classical operators to interact with "list"

- count(u): number of values selected in the list
- at(u,i) or u[i]: value at index i in the list
- indexOf(u,v): index of value v in the list
- contains(u,v): equivalent to "indexOf(u,v) != -1"
- disjoint(u1, u2, …, uk): true if u1, u2, …, uk are pairwise disjoint
- partition(u1, u2, …, uk): true if u1, u2, …, uk induce a partition of {0, …, n-1}

Traveling salesman

```
function model() {
  x <- list(N) ; // order n cities {0, ..., n-1} to visit
  constraint count(x) == N; // exactly n cities to visit
  minimize sum[i in 1..N-1]( Dist[ x[i-1] ][ x[i] ] )
                  + Dist[ x[N-1] ][ x[0] ] ; // minimize sum of traveled distances
}
```

TSP: Given a list of cities and their pairwise distances, find a shortest possible tour that visits each city exactly once.

Objective: find a permutation $a_1, \ldots, a_n$ of the cities that minimizes

$$d(a_1, a_2) + d(a_2, a_3) + \ldots + d(a_{n-1}, a_n) + d(a_n, a_1)$$

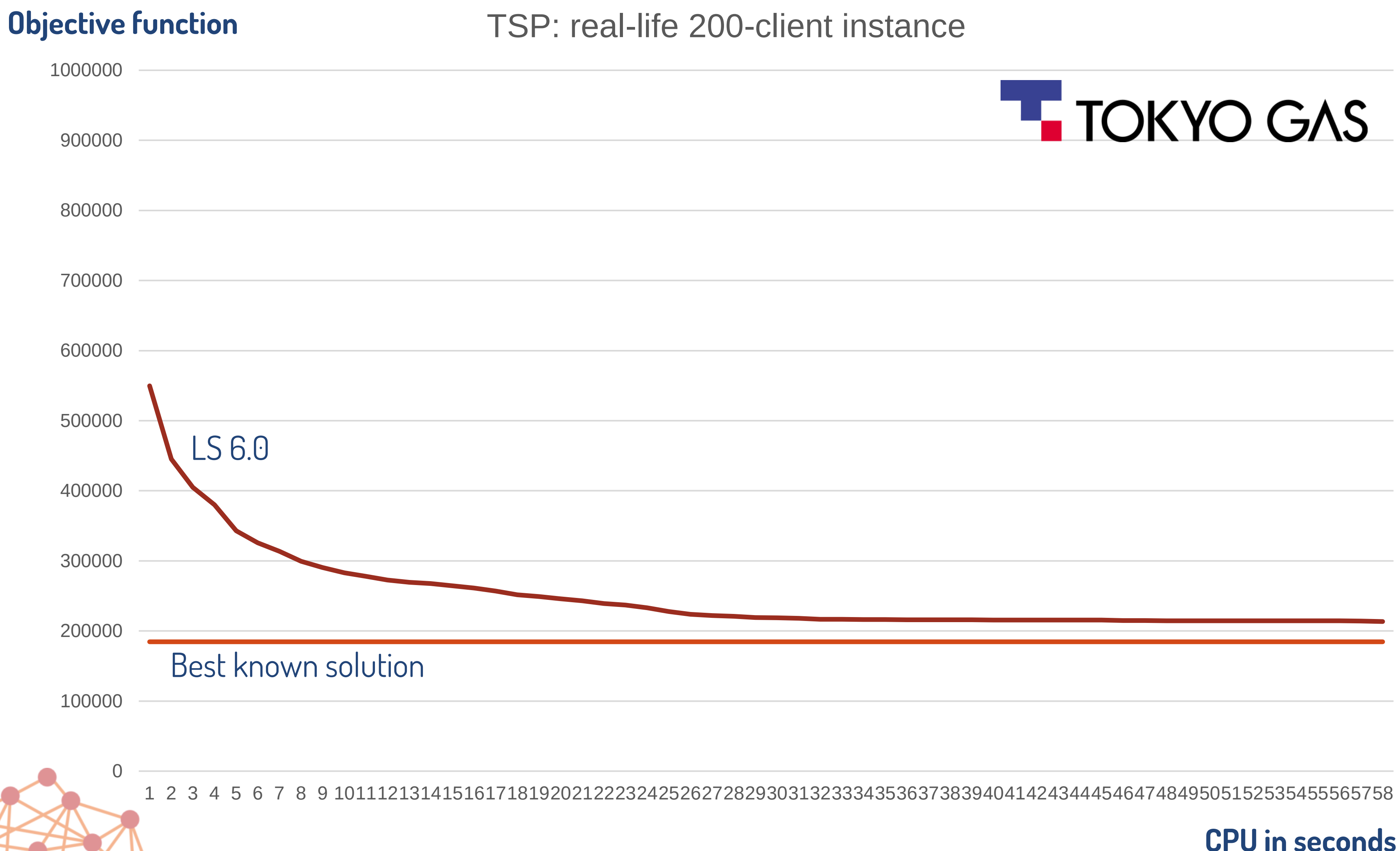
where $d(i, j)$ is the distance between cities i and j

An optimal TSP tour through Germany's 15 largest cities

Why not a single line model ?

```
constraint TSP(graph);
```

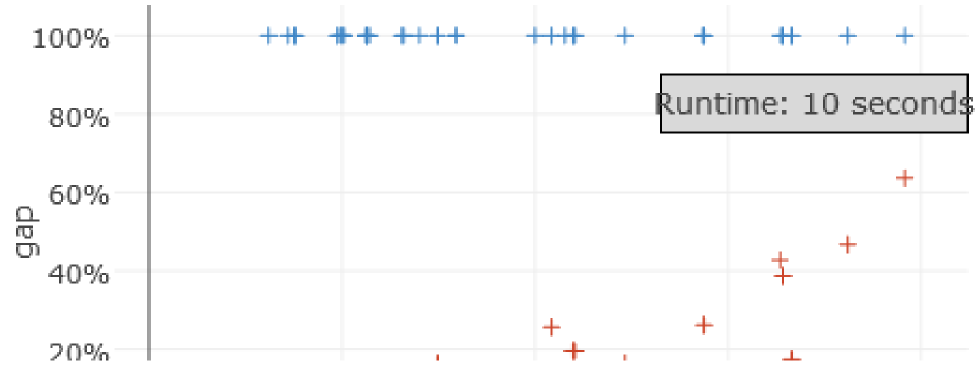
Performance ?

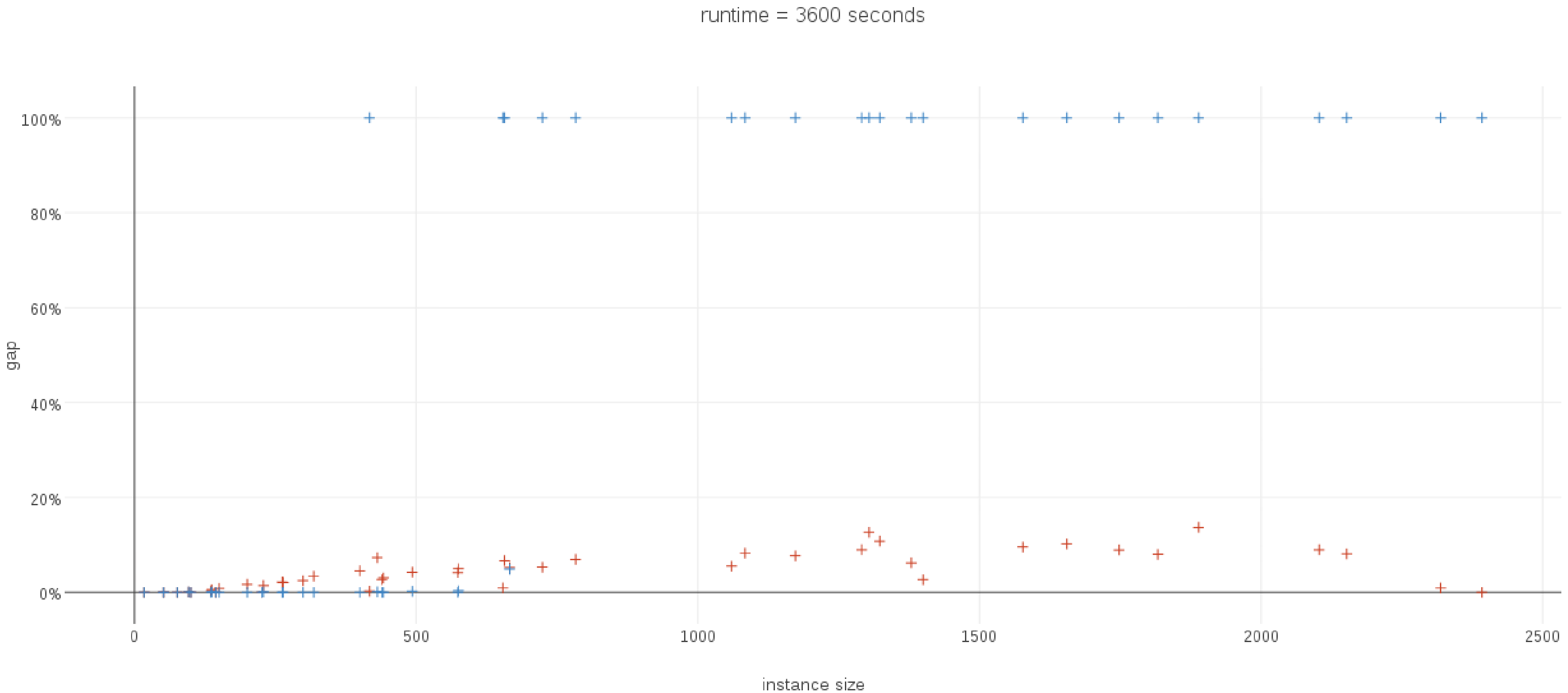


Comparison with TSP MIP approach

TSP Lib instances:

- Symmetric
- Size: 21 to 800 cities





Vehicle routing

```
function model() {
  x[1..K] <- list(N) ; // for each truck, order the clients to visit
  constraint partition( x[1..K] ); // each client is visited once
  distances[k in 1..K] <- sum[i in 1..N-1]( dist( x[k][i-1], x[k][i]) )
              + dist( x[k][N-1], x[k][0] ); // traveled distance for each truck

  minimize sum[k in 1..K]( distances[k] ); // minimize total traveled distance
}
```

	TSP	VRP
Normal	Count(x)=N	partition(x[1..K])
Prize-collecting	maximize sum(…)	disjoint(x[1..K])

CVRP benchmarks

CVRP - instances A

- 32 to 80 clients, 10 trucks max.
- 27 instances
- 5 minutes of running time
- LS binary: almost infeasible
- LS list: 1 % avg. opt. gap

CVRP - instances X100-500

- 100 to 500 clients, 138 trucks max.
- 67 instances
- 5 minutes of running time
- LS binary: almost infeasible
- LS list: 9 % avg. opt. gap

CVRPTW benchmarks

CVRPTW – instances Solomon R100

- 101 to 112 clients, 19 trucks max.
- 13 instances
- 5 minutes of running time
- LS binary: N/A
- LS list: 3 % avg. opt. gap

CVRPTW – instances Solomon R200

- 201 to 208 clients, 4 trucks max.
- 8 instances
- 5 minutes of running time
- LS binary: N/A
- LS list: 8 % avg. opt. gap

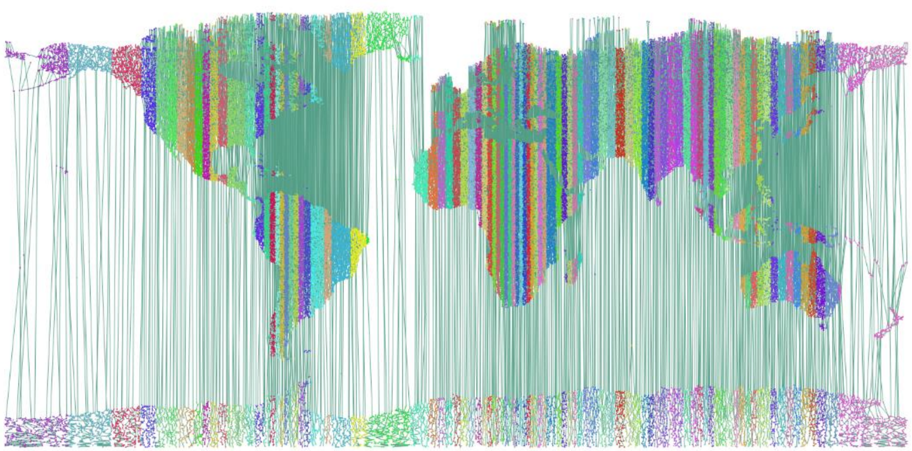
Large-scale VRP challenge (FICO/Kaggle)

100 000 cities

Sleigh capacity

Non linear objective:

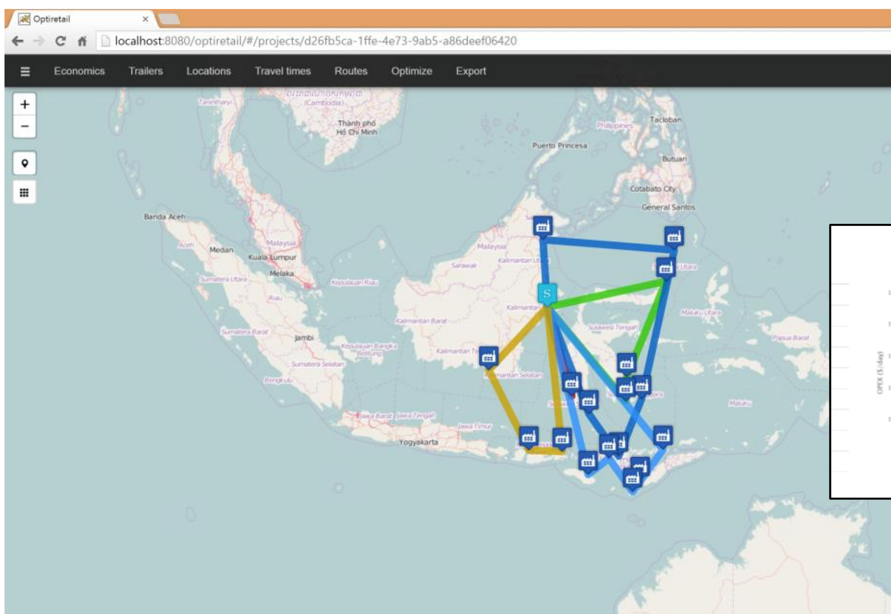
- Distance $= weightCarried \times \overbrace{2r \arcsin\left(\sqrt{\sin^2\left(\frac{\varphi_2 - \varphi_1}{2}\right) + \cos(\varphi_1)\cos(\varphi_2)\sin^2\left(\frac{\lambda_2 - \lambda_1}{2}\right)}\right)}^{haversineDistance}$

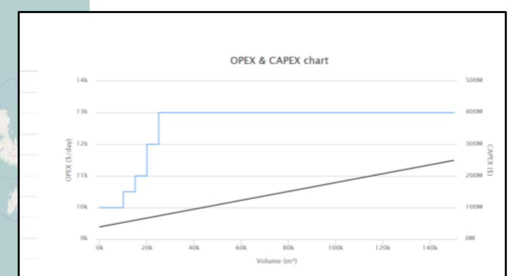


LocalSolver (Julien Darlay) ranked 31 among 1100+ competitors

Real-life VRP and fleet dimensioning

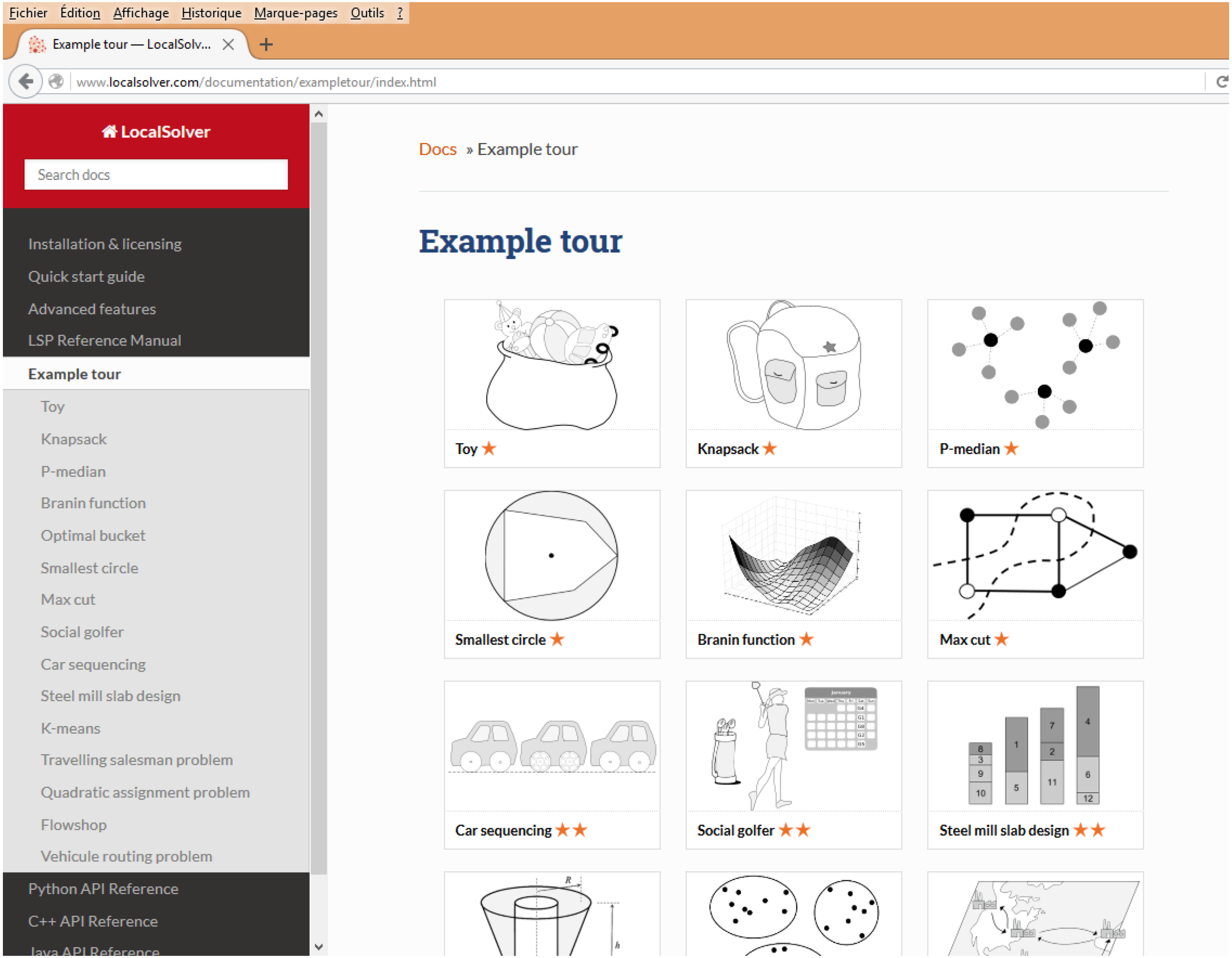
Today in room MC3 at 5:20 pm – Frédéric GARDI



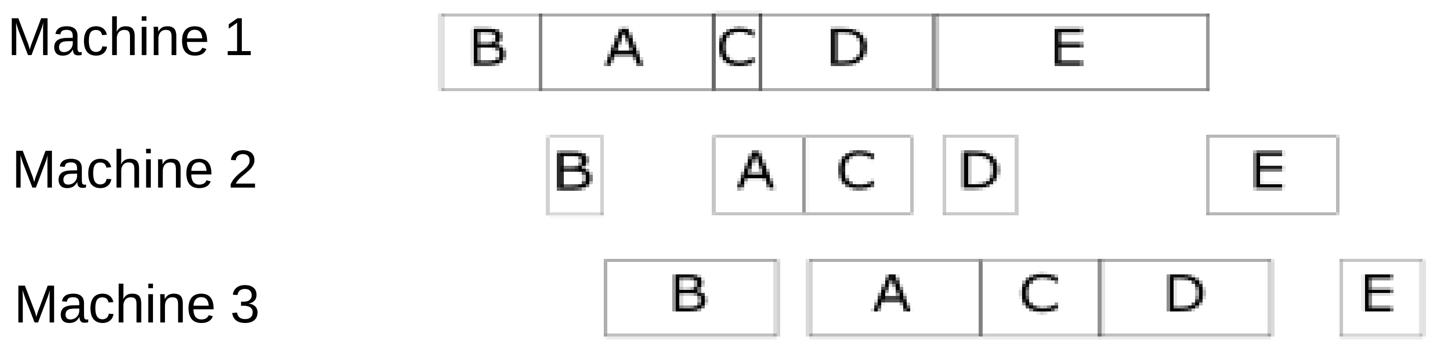


Beyond routing problems

Scheduling, planning, sequencing


Fichier Édition Affichage Historique Marque-pages Outils ?
Example tour — LocalSolv...
www.localsolver.com/documentation/exampletour/index.html
LocalSolver
Search docs
Installation & licensing
Quick start guide
Advanced features
LSP Reference Manual
Example tour
Toy
Knapsack
P-median
Branin function
Optimal bucket
Smallest circle
Max cut
Social golfer
Car sequencing
Steel mill slab design
K-means
Travelling salesman problem
Quadratic assignment problem
Flowshop
Vehicule routing problem
Python API Reference
C++ API Reference
Java API Reference
Docs » Example tour
Example tour
Toy ★
Knapsack ★
P-median ★
Smallest circle ★
Branin function ★
Max cut ★
Car sequencing ★★
Social golfer ★★
Steel mill slab design ★★

Flow-shop scheduling

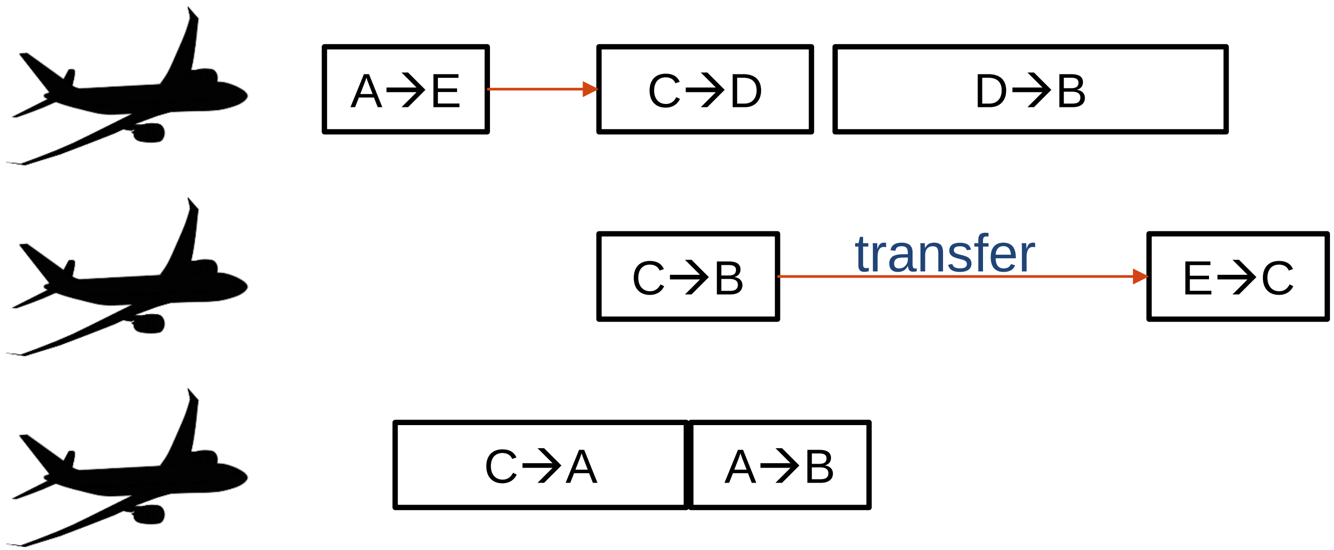


Since we are looking for a permutation of jobs the model is straightforward with a single list variable

Planning

Flights to plane assignments

STELLAR

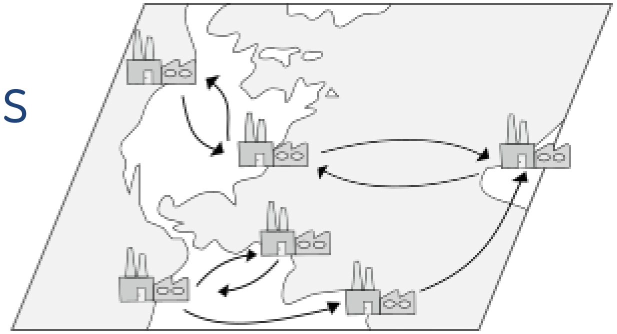


A solution is a partition of flights into K lists (one per plane)

The goal is to minimize the total transfer times

Quadratic Assignment (Facility Location)

Given flows between facilities, position facilities so as to minimize transportation costs



```
function model() {
  p <- list(N) ; // permutation of facilities on locations
  constraint count(x) == N;
  // minimize sum of distance*flow
  minimize sum[i in 1..N-1] [j in 1..N-1]( Distance[i][j] * Flow[p[i]][p[j]] ) ;
}
```

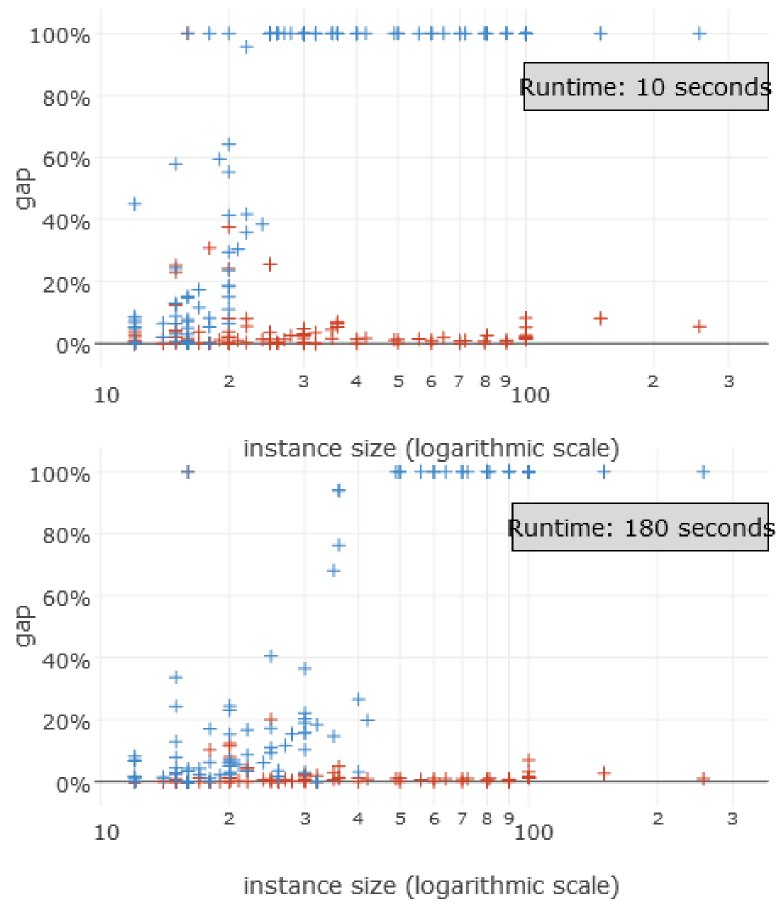
QAP Performance

QAPLib instances:

- 137 instances
- Max size:256 facilities

Gurobi 6.0

LocalSolver 6.0

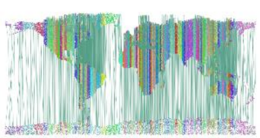


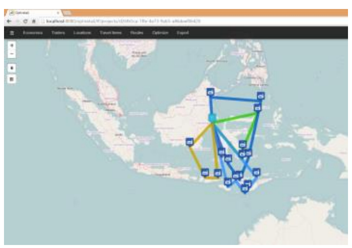
Conclusion

List Variables are a first step towards set-based modeling in LocalSolver

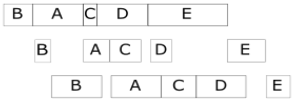
This higher level of modeling yields simple and compact models producing high quality solutions for

Routing





Scheduling



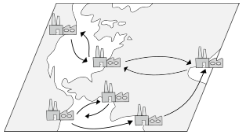
Planning

STELLAR

RCPSP

Leibniz Universität Hannover

Facility Location



Any other sequencing problem