



A mathematical optimization solver based on neighborhood search

Thierry Benoist, Julien Darlay, Bertrand Estellon,
Frédéric Gardi, Romain Megel, Clément Pajean

Innovation 24 & LocalSolver

www.localsolver.com

OR 2015
Vienna, Austria

Who we are



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

Innovation24

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

LocalSolver

Mathematical optimization solver
commercialized by Innovation 24
<http://www.localsolver.com>



LocalSolver

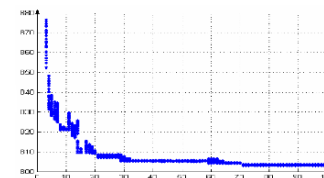
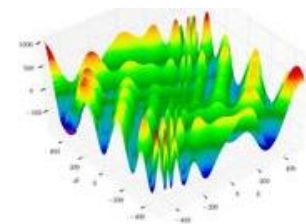
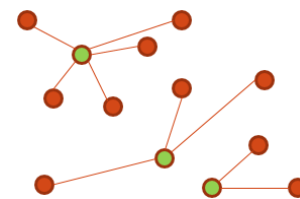
Hybrid optimization solver

For combinatorial, numerical,
or mixed-variable optimization

Suited for large-scale
nonlinear optimization

Quality solutions quickly
without tuning

LocalSolver
=
LS + CP/SAT + LP/MIP + NLP



free trial with support – free for academics
monthly rental for 590 € – perpetual license for 9,900 €

www.localsolver.com

Modeling formalism



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  
}
```

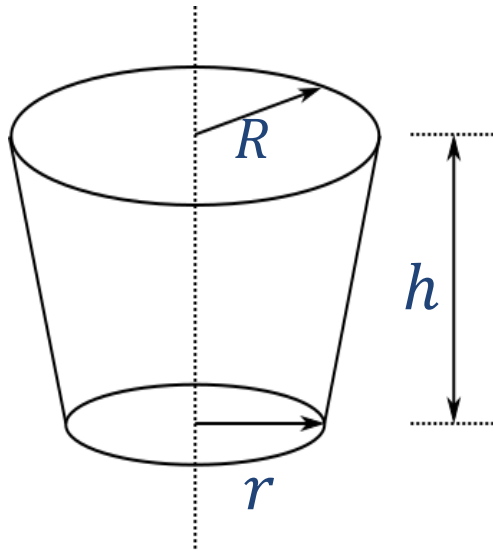
Nothing else to write: “model & run” approach

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



Bucket 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;  
}
```

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

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



Mathematical operators

Decisional	Arithmetic			Logical	Relational
bool	sum	sub	prod	not	==
float	min	max	abs	and	!=
int	div	mod	sqrt	or	<=
list	log	exp	pow	xor	>=
	cos	sin	tan	if-then-else	<
	floor	ceil	round	array + at	>
				piecewise	



Technicalities



Local search

An iterative improvement method

- Explore a neighborhood of the current solution
 - Smaller or larger neighborhoods
- Incomplete exploration of the solution space

Essential in combinatorial optimization

- Hidden behind many textbook algorithms (ex: simplex, max flow)
- In the heart of all metaheuristic approaches
- Proved to be inefficient in the worst case
- Largely used because very effective in practice



Why local search?

When it is hopeless to enumerate

- Large-scale combinatorial problems
- When relaxation or inference brings nothing (ex: linear relaxation is very fractional)
- When computing relaxation or inference is costly

Adapted to client needs

- Good-quality optima satisfy them
 - Fast: each iteration runs in sublinear or even constant time
- Solutions in short running times + ability to scale



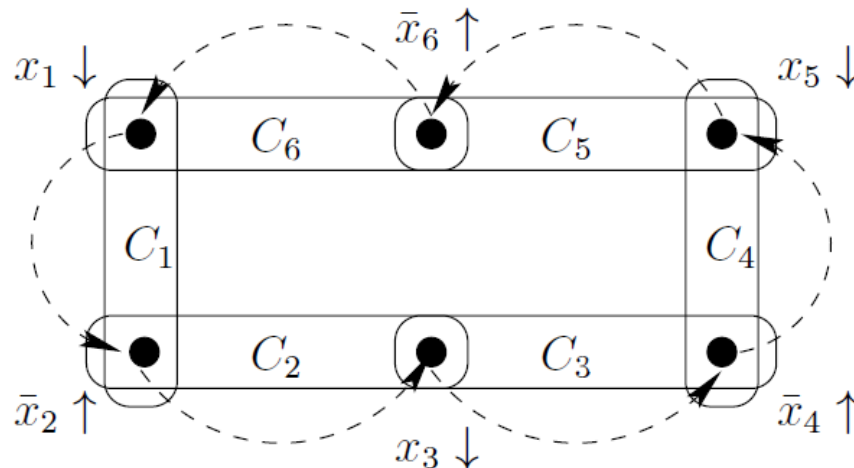
Small, structured neighborhoods

The classic in Boolean Programming: “k-flips”

- Lead to infeasible solutions for structured (= real-life) problems
- Feasibility is hard to recover: slow convergence

LocalSolver moves tend to preserve feasibility

- Destroy & repair approach
- Ejection paths in the constraint hypergraph
- More or less specific to some combinatorial structures



Large neighborhoods

Move Sequence

- Break feasibility with one move
- Recover feasibility through a sequence of other moves

Move Simplex

- Detect and exploit a linear substructure
- Use rounding techniques for integer programming

Gradient

- Compute a finite-difference gradient
- Line search along the gradient



Case studies



Car sequencing

2005 ROADEF Challenge: <http://challenge.roadef.org/2005/en>

Large-scale instances

- Until 1,300 vehicles to sequence: 400,000 binary decisions

Instance with only 540 vehicles

- Small instance: **44,000 binary decisions**
- State of the art: **3,109** (winner of the Challenge)
- Lower bound: 3,103

Minimization

Results

- Gurobi 5.5: **3.027e+06 in 10 min** | **194,161 in 1 hour**
- LocalSolver 5.0: **3,140 in 10 sec** | **3,113 in 10 min**



Supply chain optimization

Pasco



FUTURE
Architect

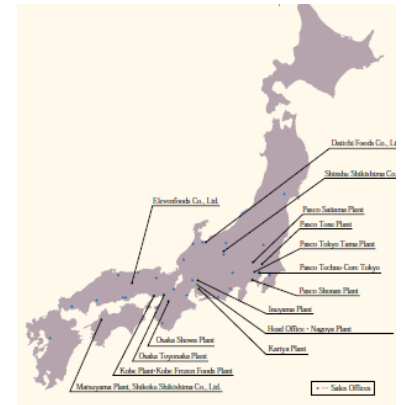


Global supply chain optimization

- Both production and logistics optimization
- 10 factories, each with several production lines
- Large number of stores and distribution centers

A challenging context for LocalSolver

- 20,000,000 expressions including 3 million binaries
- Rich model involving setup costs, delivery times, packaging, etc.
- Vain attempts to solve the problem with MIP solvers
- LocalSolver finds high-quality solutions in 5 minutes



Application panorama



TV media planning



Outdoor & indoor advertising



Logistic clustering and routing



Road maintenance planning



Network deployment planning



Loan assembling optimization



Placement of nuclear fuel assemblies in pools



Airline network management



Weapon resource allocation



Packing and transportation of military equipment



Novelties



Set-based modeling

Structured decisional operator `list(n)`

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

Classical operators to interact with “list”

- **count**(u): number of values selected in the list
- **get**(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 $\{0, \dots, n-1\}$



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  
}
```

Could you imagine a simpler model?

- Natural declarative model: straightforward to understand
- Common set-oriented concepts: easy to learn
- Even easier for people with basic programming skills
- Compact: linear in the size of input → highly scalable



CVRP benchmarks

CVRP – instances A

- 32 to 80 clients, 10 trucks max.
- 27 instances
- 5 minutes of running time
- LS binary: 3 % avg. opt. gap
- **LS list: 1 % avg. opt. gap**

CVRP – instances X100–500

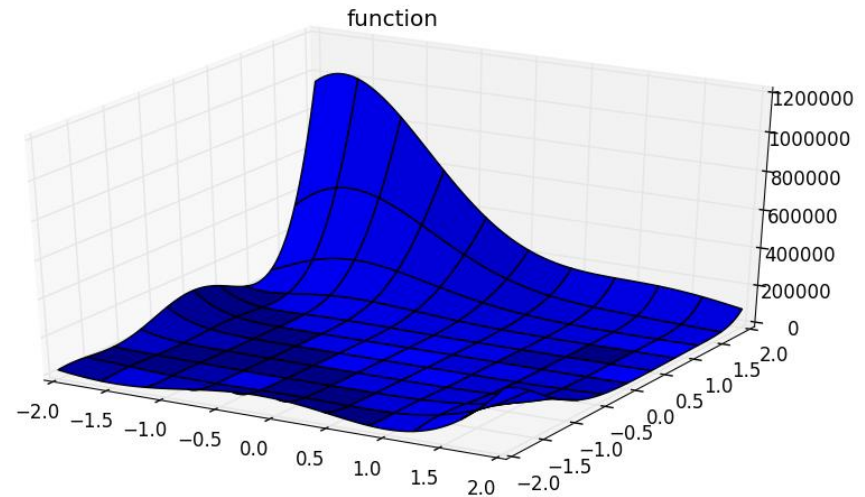
- 100 to 500 clients, 138 trucks max.
- 67 instances
- 5 minutes of running time
- LS binary: N/A
- **LS list: 9 % avg. opt. gap**



Black-box optimization

Context

- Unknown objective (oracle)
- Costly to evaluate
- Derivative-free
- Continuous & integer decisions
- Bounds on decisions



Many applications in engineering

- Multidisciplinary/parametric optimization
 - Simulation optimization (noisy, nondeterministic)
- Design optimization of materials/systems: mechanics, electricity, logistics, etc.



John N. Hooker (2007)

“Good and Bad Futures for Constraint Programming (and Operations Research)”
Constraint Programming Letters 1, pp. 21-32

“Since modeling is the master and computation the servant, no computational method should presume to have its own solver.

This means there should be no CP solvers, no MIP solvers, and no SAT solvers. All of these techniques should be available in a single system to solve the model at hand.

They should seamlessly combine to exploit problem structure. Exact methods should evolve gracefully into inexact and heuristic methods as the problem scales up.”

