LocalSolver

A mathematical optimization solver based on neighborhood search

Julien Darlay

jdarlay@localsolver.com

www.localsolver.com

EA 2017
Paris, France

Who we are

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

Operations Research subsidiary of Bouygues
Mathematical optimization solver
http://www.localsolver.com

Swiss Army Knife for math optimization

Model & Run

Discrete, Numerical, Black-Box

Fast & Scalable

Innovative Resolution Technology

Agenda

1. Origins of LocalSolver
2. Quick tour & examples
3. A look inside LocalSolver
4. How to write good models for LocalSolver
5. New features in LocalSolver 7.5

Origins of LocalSolver

Automate local search

Local search

An iterative improvement method

- Explore a neighborhood of the current solution
- Small or large neighborhoods
- First improve

$\longrightarrow$ 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 solution satisfy them
- Fast: each iteration runs in sublinear or even constant time
- Focus only on models

⟶ Solutions in short running times + ability to scale

Existing tools

Libraries and frameworks

- Complex to handle
- Limited to practitioners having good programming skills
- Don't address key points (ex: moves)

Solvers integrating "pure" local search

- Pioneering works in SAT community
- MIP & CP: a few attempts but a limited impact (Nonobe & Ibaraki 2001)
- MIP & CP: a lot of heuristic ingredients but no "pure" local search

LocalSolver project

2007: Beginning of the project

- Define a generic modeling formalism (close to MIP) suited for a local search-based resolution (*model*)
- Develop an effective solver based on pure local search with first principle: “to do what an expert would do” (*run*)

2010: First version of LocalSolver

- Large-scale combinatorial problems – especially assignment, packing, covering, partitioning problems – out of scope of classical solvers
- Integration in our own optimization solutions
- First uses outside LocalSolver

LocalSolver project

One major version per year focused on functionalities

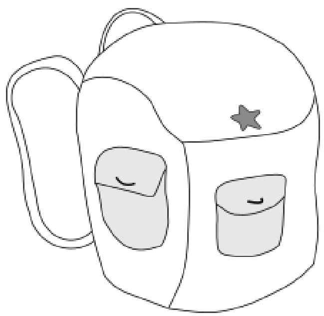
- Continuous & Integer decisions
- Set based models
- Inconsistency core
- Black-Box optimization

One minor version per year focused on performances

- Continuous optimization algorithms
- MIP techniques
- Preprocessing

Quick tour & examples

Knapsack



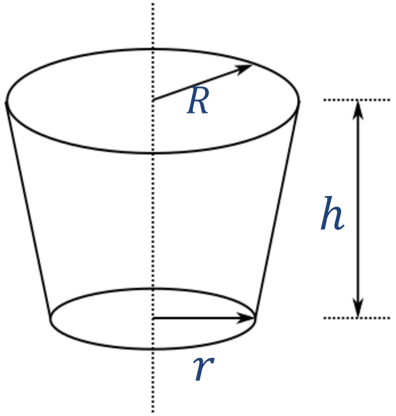
Given a set of items, each with a weight and a value, determine a subset of items in such a way that their total weight is less than a given bound and their total value is as large as possible.

```
function model() {

    x[i in 0..nbItems-1] <- bool();
    knapsackWeight <- sum[i in 0..nbItems-1](weights[i] * x[i]);
    constraint knapsackWeight <= knapsackBound;

    knapsackValue <- sum[i in 0..nbItems-1](prices[i] * x[i]);
    maximize knapsackValue;
}
```

Nothing else to write: “model & run” approach

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

Parametric optimization



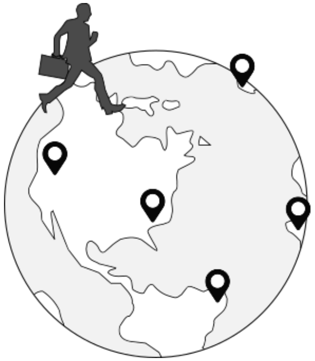
Maximize the volume of a bucket with a given surface of metal

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

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

```
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;
}
```

Traveling salesman



Given a list of N cities and the distances between each pair of cities, what is the shortest possible route that visits each city exactly once and returns to the origin city?

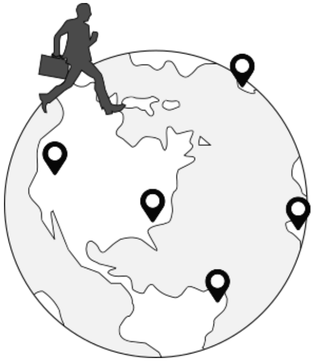
MIP models are bad for local search

- X_{ij} is one if city i is before city j in the solution
- Exactly one entering and leaving arc per city
- Subtour eliminations 2^n constraints

Assignment model is a good alternative

- X_{ip} is one if city i is in position p in the tour
- Each city is exactly in one position
- N^2 Decisions
- N constraints

Traveling salesman



Given a list of N cities and the distances between each pair of cities, what is the shortest possible route that visits each city exactly once and returns to the origin city?

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

Efficient model

- Textbook-like (Garey & Johnson)
- Compact
- Highly-scalable

Mathematical operators

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

+ operator **call** : to call an external native function which can be used to implement your own operator

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 540 vehicles

- Small instance: 80,000 variables including 44,000 binary decisions
- State of the art: 3,109 by specific local search (winner of the Challenge)
- Lower bound: 3,103

Results

Minimization

- MIP Solver: 3.027e+06 in 10 min | 194,161 in 1 hour
- LocalSolver: 3,140 in 10 sec | 3,113 in 10 min

Applications

Central Entity (fr)	Airbus Operations (fr)	ALFI (fr)	CRCD (fr)	Future Architect (jp)	Hitachi YRL (jp)	Helmut Schmidt U. (de)	Henan Normal U. (cn)
Altran Prime (fr)	A-SIS (fr)	Bacallan (jp)	Bouygues E&S (fr)	Horasphere (ca)	Inner Mongolia N. U. (cn)	INSA Rennes (fr)	IT-CE (fr)
Bouygues Immobilier (fr)	Bouygues Telecom (fr)	Bouygues SA (fr)	CLAI (it)	Kagawa Prefecture (jp)	Kyoto University (jp)	LAC (it)	Nanjin University (cn)
University of Coimbra (pt)	Colas (uk)	Leeds, CU-Boulder (us)	Dongbei University (cn)	NIES (jp)	MBDA (fr)	MédiaTransports (fr)	Mediavision (fr)
DTP (fr)	Eco-Management (fr)	EDF R&D (fr)	EdgeStone IT (cn)	Mereo (fr)	Mie University (jp)	MSI (jp)	Osaka University (jp)
Elettric 80 (it)	EMBIX (fr)	CRIGEN (fr)	ESIEE Paris (fr)	Otaru University (jp)	Pasco Shikishima (jp)	Renault (fr)	UFRN (br)
Eurodecision (fr)	French Army (fr)	Fujitsu Laboratories (jp)	Fujitsu SIL (jp)	Rovi Corporation (us)	Senshu University (jp)	Shanghai Jia Tong U. (cn)	SNCF Réseau (fr)
Timeplus (fr)	Universal Studios (jp)	VERI (fr)	Zhongnan University (cn)	Socio Logiciels (fr)	Stellar Labs (us)	TF1 Publicité (fr)	Tongji University (cn)

A look inside LocalSolver

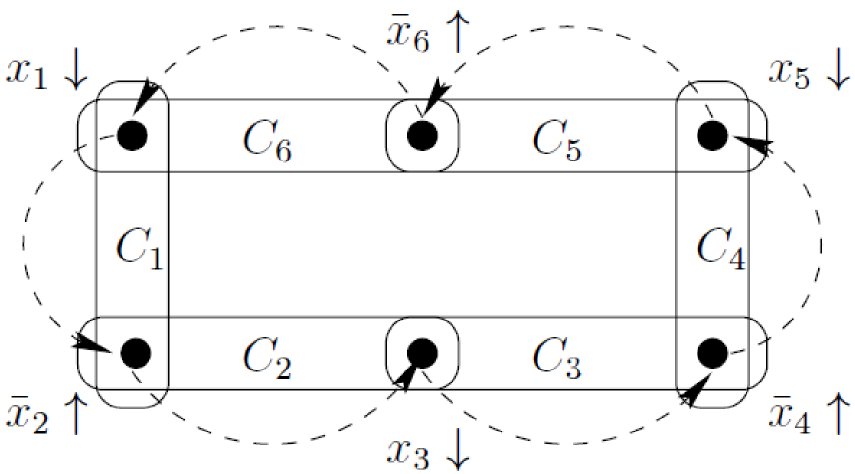
Small, structured neighborhoods

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

Destroy & Repair

- Break feasibility with one or several moves
- Retrieved it with a series of other moves

Integer programming neighborhood

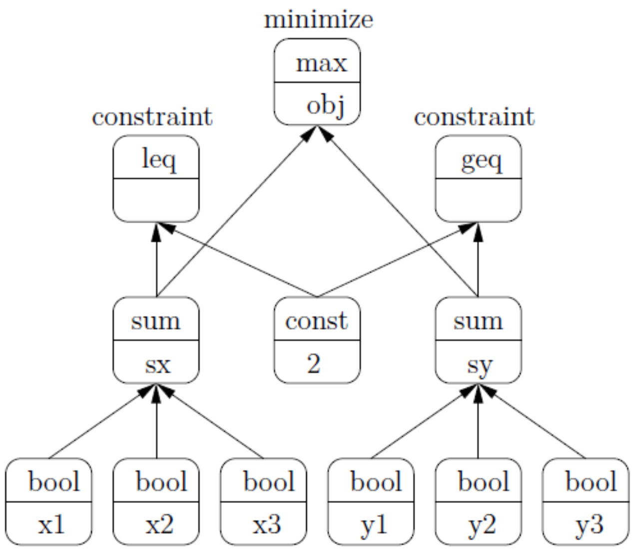
- Exploit a linear substructure
- Use rounding techniques for integer programming

Direction search

- Compute a good direction
- Line search along this direction

Fast exploration

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



Incremental evaluation

- Lazy propagation of modifications induced by a move in the DAG
- Exploitation of invariants induced by math operators

⟶ Millions of moves evaluated per minute of running time

Heuristic

Online learning of moves

- Discard inefficient moves
- Improve efficient moves selection

Simulated annealing

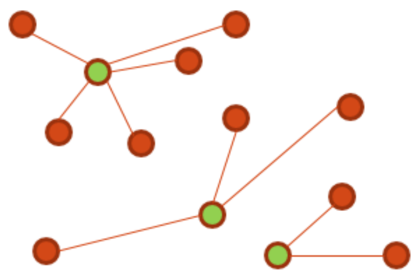
- Handle non smooth objectives
- Allow degrading solutions

Restarts + parallel search

- Avoid local optima
- Improve search space coverage

How to write a good model

P-Median



Select a subset P among N points and affect each point in N to a point in P such that the total distance is minimized

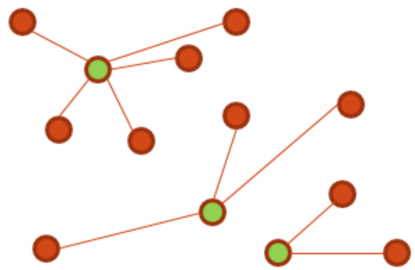
```
x[1..N] <- bool();
y[1..N][1..N] <- bool();

constraint sum[i in 1..N] (x[i]) <= p;
for[i in 1..N]{
   constraint sum[j in 1..N] (y[i][j]) == 1;
}
for[i in 1..N][j in 1..N]{
   constraint y[i][j] <= x[j];
}

minimize sum[i in 1..N][j in 1..N] (y[i][j] * w[i][j]);
```

- $N^2 + N$ Decisions (ex: N=900)
- $N^2 + N + 1$ Constraints
- Needs to simultaneously modify x and y
- 7s to feasibility, gap=350% after 10s ☹

How can we improve this model for local search ?

- Less decisions
- Less constraints
- Reduce "distance" between feasible solution

P-Median



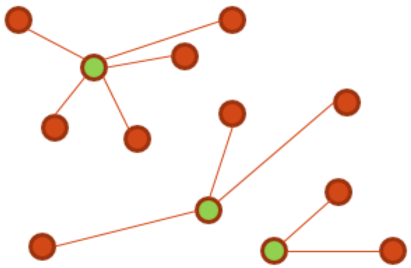
Remove decisions and replace them with possibly non linear expressions

```
y[1..N][1..N] <- bool();

x[j in 1..N] <- or[i in 1..N] (y[i][j]);

for[i in 1..N]{
   constraint sum[j in 1..N] (y[i][j]) == 1;
}

constraint sum[i in 1..N] (x[i]) <= p;

minimize sum[i in 1..N][j in 1..N] (y[i][j] * w[i][j]);
```

- N^2 Decisions (ex: N=900)
- N + 1 Constraints
- Feasibility in 4 seconds
- Gap 58% after 10s ☺

Did we pick the right set of decisions ?

P-Median



The decisions are the points in P since the affectation is not constrained

```
x[1..N] <- bool() ;
constraint sum[i in 1..N]( x[i] ) <= P ;
minDist[i in 1..N] <- min[j in 1..N]( x[j] ? Dist[i][j] : Inf ) ;
minimize sum[i in 1..N]( minDist[i] ) ;
```

Best model for LocalSolver

- N Decisions (ex: N=900)
- 1 Constraint
- Each move can only change 1 decision
- 0s to find feasibility, gap=6% after 10s ☺ ☺

New features

LocalSolver 7.5

Variadic operators

Variadic operators (lambda expressions)

sum(a..b, i => f(i)) = $\sum_{i=a}^{b} f(i)$

Dynamic Range $[a, b]$

Function f(i)

```
function model() {
  route <- list(clientsCount);
  demandsOnRoute = sum(0..count(route)-1, i => demands[route[i]]);
  /* ... */
```

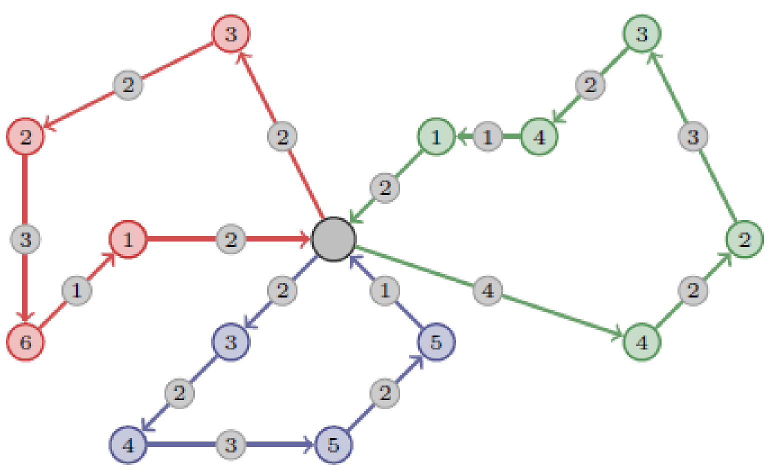
Vehicle routing

```
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] <- A sum with a variable number of terms

  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])

Vehicle routing

```
function model() {
  routes[1..k] <- list(clientsCount);
  constraint partition[i in 1..k](routes[i]);
  for [r in 1..k] {
     route <- routes[r];
     l <- count(route);
     constraint sum(0..l-1, i => demands[route[i]]) <= capacity;
     dist[r] <- sum(0..l-2, i => distance(route[i], route[i+1]))
                  + distance(depot, route[0]) + distance(route[l-1], depot);
  }
  minimize sum[r in 1..k](dist[r]);
}
```



Improved large neighborhoods

Continuous Optimization

- Faster computation of first order information
- New neighborhood based on classical algorithms (Conjugate gradient, BFGS)

Mixed Integer optimization

- Better linearization of operators (min, max, and…)
- Larger neighborhood (performance improvement)

Roadmap

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.”

A mathematical optimization solver based on neighborhood search

Julien Darlay

jdarlay@localsolver.com

www.localsolver.com

EA 2017
Paris, France