LocalSolver

A New Kind of Math Programming Solver

Thierry Benoist Julien Darlay Bertrand Estellon
Frédéric Gardi Romain Megel

jdarlay@localsolver.com

www.localsolver.com

Who are we ?

BOUYGUES

Diversified industrial group focused on construction, telecom and media
http://www.bouygues.com

Innovation24

Optimization subsidiary of Bouygues
15 years of experience in Operations Research
http://www.innovation24.fr

LocalSolver

Math programming solver
for combinatorial or mixed optimization
http://www.localsolver.com

LocalSolver

Solver for combinatorial & continuous optimization

- Simple mathematical modeling formalism
- Allows to tackle large-scale problems
- Provides good-quality solutions in short running times

Solver based on local search

- Moves based on decisions/constraints hypergraph
- Incremental evaluation: millions of moves per minute
- Adaptive, randomized, parallelized simulated annealing with restarts

Free academic licenses

Commercial licenses from 990 €

LocalSolver 4.0

Mathematical programming solver

- **For combinatorial optimization**
- For numerical optimization
- For mixed-variable optimization

- **Provides solutions (upper bounds)**
- Provides lower bounds
- Infeasibility gap/proof, optimality gap/proof

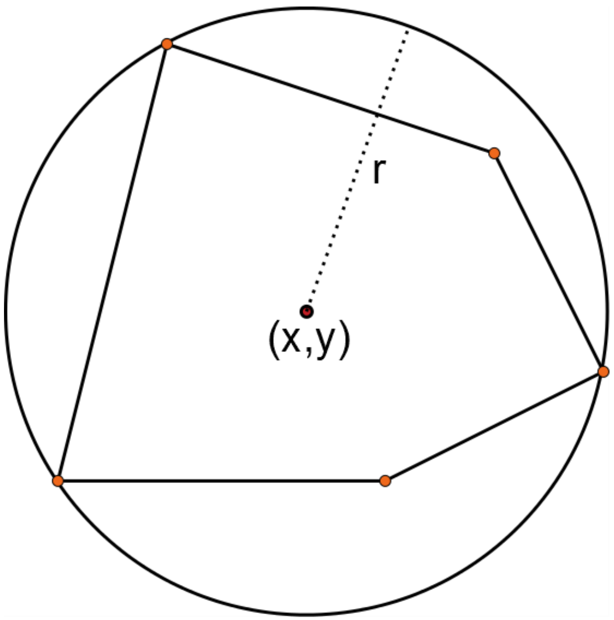
Suited for large-scale non-convex optimization

- Millions of combinatorial and/or continuous variables
- Non-convex constraints and/or objectives
- Short resolution times

Numerical optimization

Smallest Circle

- Find the circle of minimum radius including a set of points
- Two continuous decisions: ***x*** and ***y***
- The radius ***r***: expression deduced from decisions
- Straightforward quadratic model



Continuous decision

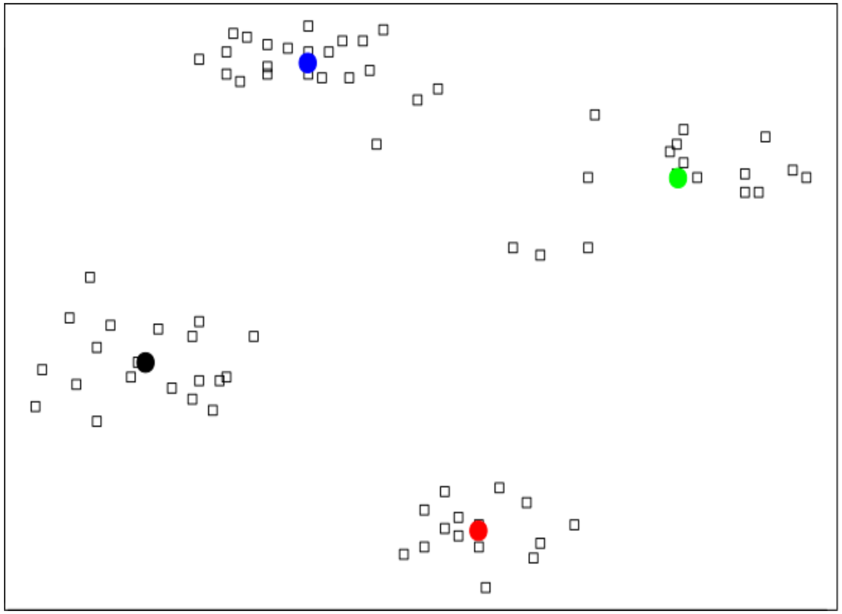
Quadratic expression

```
x <- float(minX, maxX);
y <- float(minY, maxY);

r2 <- max[i in 1..n] (pow(x-coordX[i],2) + pow(y-coordY[i],2));

minimize sqrt(r2);
```

Non convex constrained optimization

K-means

- Find a partition of a set of N observations into K classes to minimize the within-cluster sum of squares
- NP-Hard, Quadratic



Non convex constrained optimization

K-means

```
for[i in 1..k][j in 1..D]{
        x[i][j] <- float(mini[j],maxi[j]);
}

for[i in 1..N]{
        d[i] <- min[l in 1..k](sum[j in 1..D]((x[l][j] - M[i][j])^2));
}

minimize sum[i in 1..nbLines](d[i]);
```

Instance	k	OPT*	LS 4.0	GAP
iris	2	152,348	152,369	0,01%
	3	78,8514	78,9412	0,11%
	4	57,2285	57,3556	0,22%
	5	46,4462	46,5363	0,19%
	6	39,04	41,7964	7,06%
	7	34,2982	34,6489	1,02%
	8	29,9889	30,3029	1,05%
	9	27,7861	28,0667	1,01%
	10	25,834	26,0521	0,84%

Constrained combinatorial optimization

Binary feature selection

- Choose the minimum number of features
- Distinguish between positive and negative observations
- Usefull to find “patterns” inside the dataset
- NP-Hard problem

Id	Diagnostic	Fatigue	Surgery	Pain	Fever
1	Negative	0	1	0	0
2	Negative	0	1	0	1
3	Negative	1	1	0	0
4	Negative	1	0	1	0
5	Positive	1	1	1	1
6	Positive	1	0	1	1
7	Positive	0	1	1	1
8	Positive	1	1	0	1

Constrained combinatorial optimization

Binary feature selection

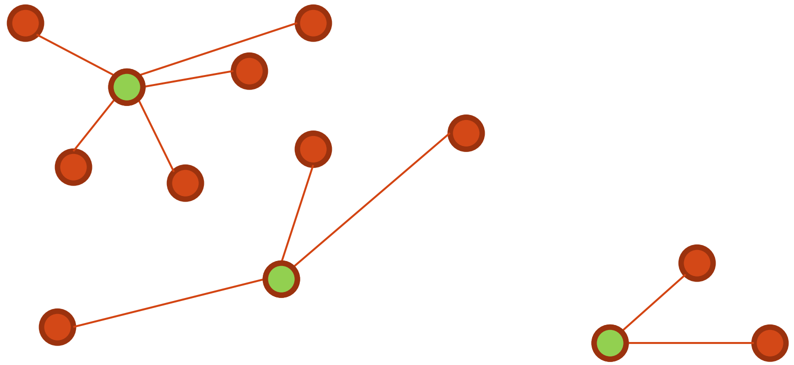
```
x[1..M] <- bool();

for[i in 1..nbObs : classes[i] == 1][j in 1..nbObs: classes[j]== 0]{
        constraint sum[k in 1..M: m[i][k] != m[j][k]] (x[k]) >= 1;
}

minimize sum[i in 1..M] (x[i]);
```

Id	Diagnostic	Fatigue	Surgery	Pain	Fever
1	Negative	0	1	0	0
2	Negative	0	1	0	1
3	Negative	1	1	0	0
4	Negative	1	0	1	0
5	Positive	1	1	1	1
6	Positive	1	0	1	1
7	Positive	0	1	1	1
8	Positive	1	1	0	1

Constrained combinatorial optimization

P-Median

- Find a subset of P elements in a set of N
- Minimize the sum of distances from each element to the closest one in P



```
function model() {
    x[1..N] <- bool();
    constraint sum[i in 1..N] (x[i]) == P;

    minDistance[i in 1..N] <- min[j in 1..N]  (x[j] ? distance[i][j] : +inf);
    minimize sum[i in 1..N] (minDistance[i]);
}
```

LocalSolver

Local search

Main idea for combinatorial optimization

- Sequential modification of a small number of decisions
- Maintaining the feasibility of current solution
- Incremental evaluation, generally in $O(1)$ time

→ Small improvement probability but small time and space complexity

In continuous optimization?

- Known under another name: *direct = derivative-free = zeroth-order search*
- Don't use gradients (1st order) nor Hessian (2nd order)
- Ex: Nelder-Mead simplex algorithm
- Mainly used in <u>unconstrained</u> non-convex optimization

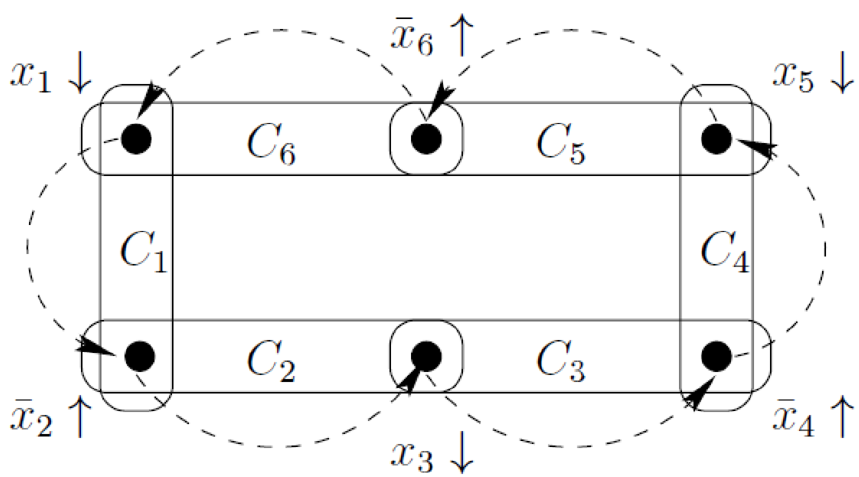
Neighborhoods

Standard moves in combinatorial optimization: “k-flips”

- Could lead to infeasible solution on real instances
- If feasibility is hard to reach: slow convergence

LocalSolver maintains feasibility

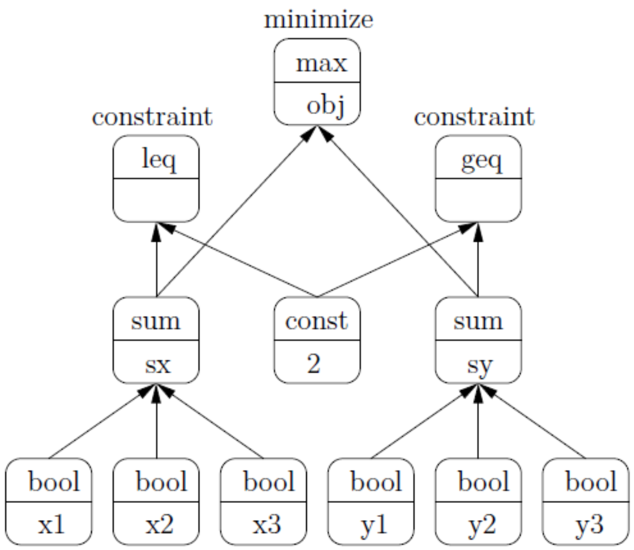
- « Destroy & repair »
- Ejection chain on constraint graph
- Use of known combinatorial structure



LocalSolver

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 in the expression DAG
- Usage of invariants

⟶ Millions of moves per minute

Toward an “all-in-one” solver

One solver to tackle all kinds of problems

- Discrete, numerical, or mixed-variable optimization
- From small-scale to large-scale problems
- Best effort to prove infeasibility or optimality
- Able to scale heuristically faced with large problems

One solver offering the best of all optimization techniques

- Local and direct search
- Constraint propagation and inference
- Linear and mixed-integer programming
- Nonlinear programming (convex and non-convex)
- Dynamic programming
- Specific algorithms: paths, trees, flows, matchings, etc.

LocalSolver

LocalSolver

A New Kind of Math Programming Solver

Thierry Benoist   Julien Darlay   Bertrand Estellon
Frédéric Gardi   Romain Megel

jdarlay@localsolver.com

www.localsolver.com