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, one of the French largest corporation, €33 bn in revenues

http://www.bouygues.com

Operation Research subsidiary of the Bouygues group

http://www.innovation24.fr

LocalSolver, mathematical optimization solver commercialized by Innovation 24

http://www.localsolver.com

LocalSolver

Solver for combinatorial & continuous optimization

- Simple mathematical modeling formalism
  - C++, Java, .NET APIs, R
  - Modeling Language (LSP)
- Allows to tackle large-scale problems
- Provides good-quality solutions in short running times

Free academic licenses

Renting offers from 590€ / month

Perpetual licenses from 9900 €

LocalSolver 4.5

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

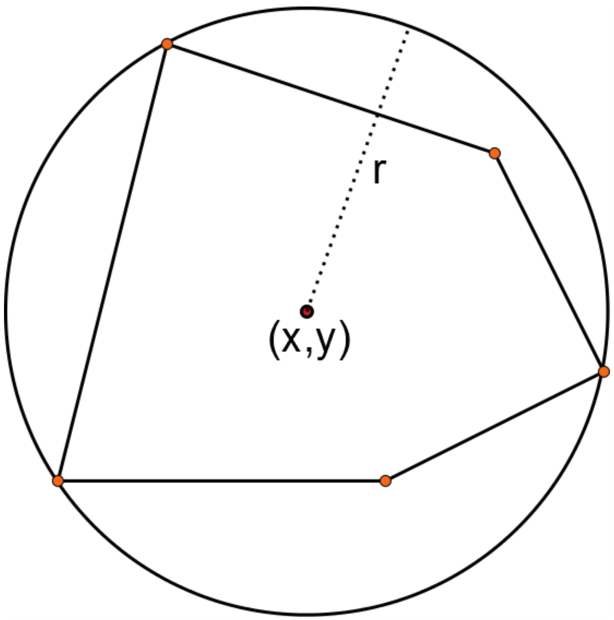
- Millions of combinatorial and/or continuous variables
- Non-convex constraints and/or objectives
- Short resolution times

Examples

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

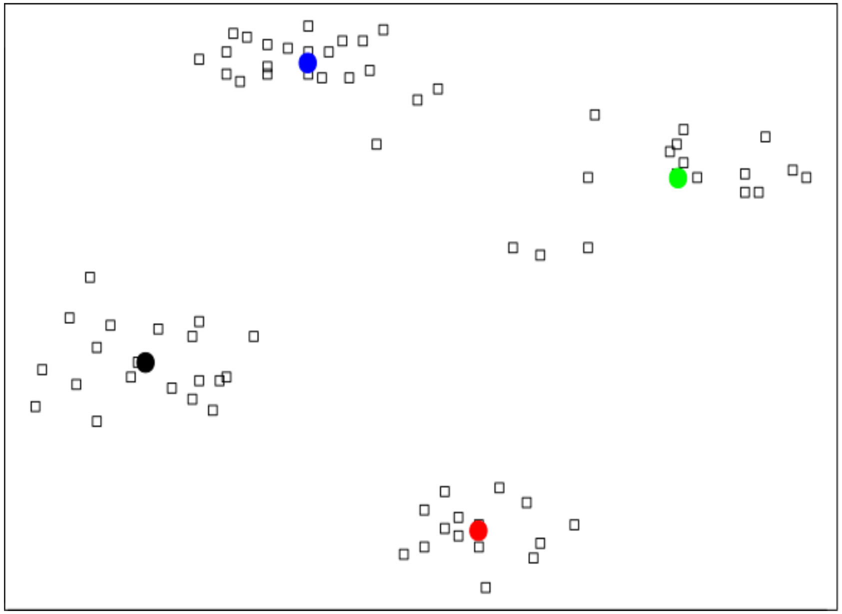
Quadratic expression

```
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



LocalSolver

Non convex constrained optimization

K-means

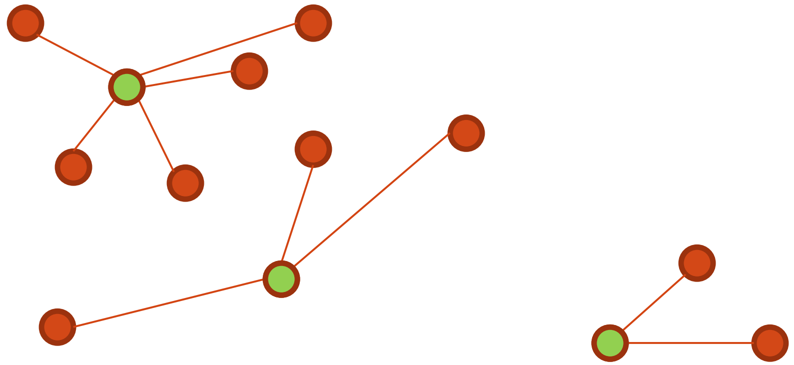
```
for[i in 1..k][j in 1..D]{
        x[i][j] <- float(mini[j],maxi[j]);
}

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

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]);
}
```

Local Search / Direct Search

Local Search

Main idea for combinatorial optimization

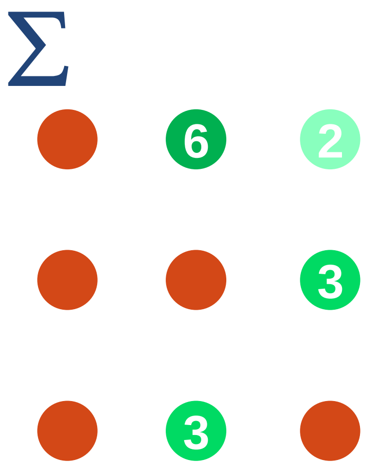
- Sequential modification of a small number of decisions
- Maintaining the feasibility of current solution
- Incremental evaluation, generally in $O(1)$ time

→ Small improvement probability but small time and space complexity

→ Simple extension to direct search in continuous optimization

A three layers architecture

- Moves based on mathematical model
- Incremental evaluation of solutions
- Heuristic to drive the search



LocalSolver

Moves

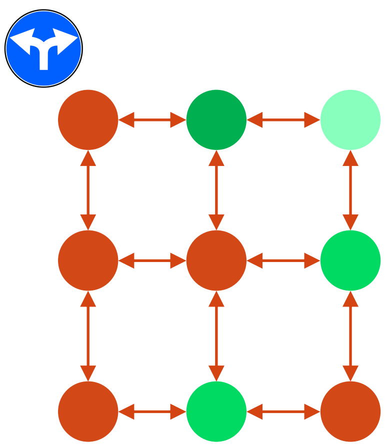
Standard moves in combinatorial optimization: “k-flips”

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

LocalSolver maintains feasibility

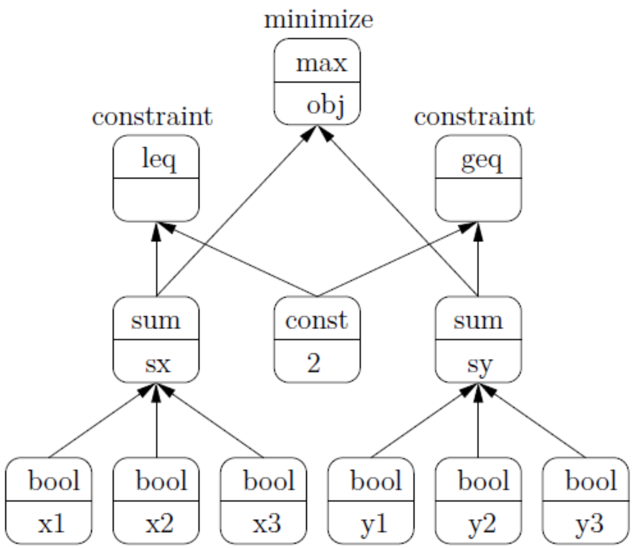
- « Destroy & repair »
- Ejection chain on constraint graph
- Use of known combinatorial structure

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



Fast exploration

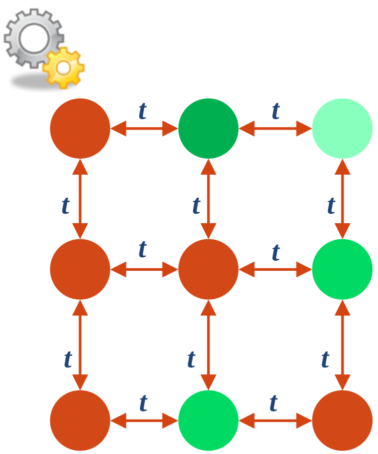
```
x1 <- bool();
x2 <- bool();
x3 <- bool();
y1 <- bool();
y2 <- bool();
y3 <- bool();
sx <- sum(x1, x2, x3);
sy <- sum(y1, y2, y3);
constraint leq(sx, 2);
constraint geq(sy, 2);
obj <- max(sx, sy);
minimize obj;
```



Incremental evaluation

- “Lazy” propagation in the expression DAG
- Usage of invariants

⟶ Millions of moves per minute



Heuristic

Online learning of moves

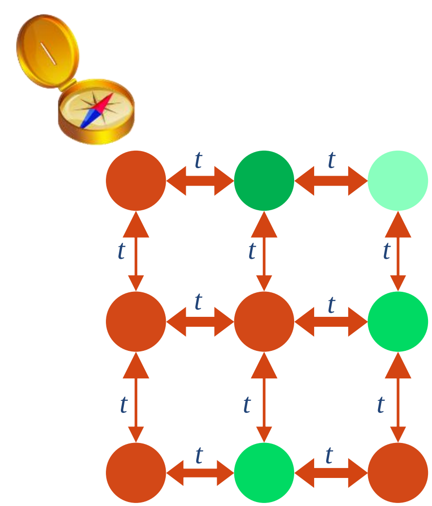
- Discard inefficient moves
- Improve efficient moves selection

Simulated annealing

- Handle non smooth objectives
- Allow degrading solutions

« Restart » + parallel search

- Avoid local optima
- Improve search space coverage



Benchmarks

Combinatorial optimization

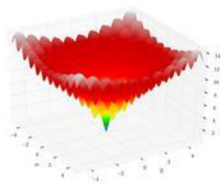
Car Sequencing : schedule cars among an assembly line

10 sec	100	200	300	400	500
Gurobi 5.5	140	274	X	429	513
LocalSolver 4.0	8	5	8	10	19
60 sec	**100**	**200**	**300**	**400**	**500**
Gurobi 5.5	3	66	1	356	513
LocalSolver 4.0	6	4	3	5	6
600 sec	**100**	**200**	**300**	**400**	**500**
Gurobi 5.5	3	2	*0	1	20
LocalSolver 4.0	4	*0	*0	2	*0

Nonconvex optimization

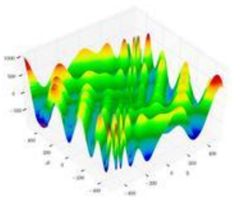
Nearly optimal solution in a few seconds on several artificial landscape from the literature

Oldenhuis (2009). Test functions for global optimization algorithms. Matlab



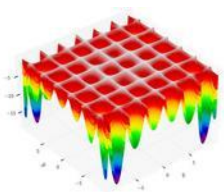
$$f(x,y) = -20\exp\left(-0.2\sqrt{0.5\left(x^2+y^2\right)}\right) - \exp\left(0.5\left(\cos\left(2\pi x\right)+\cos\left(2\pi y\right)\right)\right) + 20 + e.$$

gap (%) < 10^{-6}



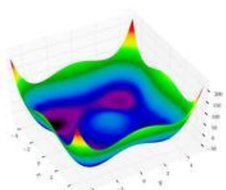
$$f(x,y) = -\left(y+47\right)\sin\left(\sqrt{\left|y+\frac{x}{2}+47\right|}\right) - x\sin\left(\sqrt{\left|x-\left(y+47\right)\right|}\right).$$

gap (%) < 10^{-4}



$$f(x,y) = -\left|\sin\left(x\right)\cos\left(y\right)\exp\left(\left|1-\frac{\sqrt{x^2+y^2}}{\pi}\right|\right)\right|.$$

gap (%) < 10^{-4}



$$f(\boldsymbol{x}) = \frac{\sum_{i=1}^{n} x_i^4 - 16x_i^2 + 5x_i}{2}.$$

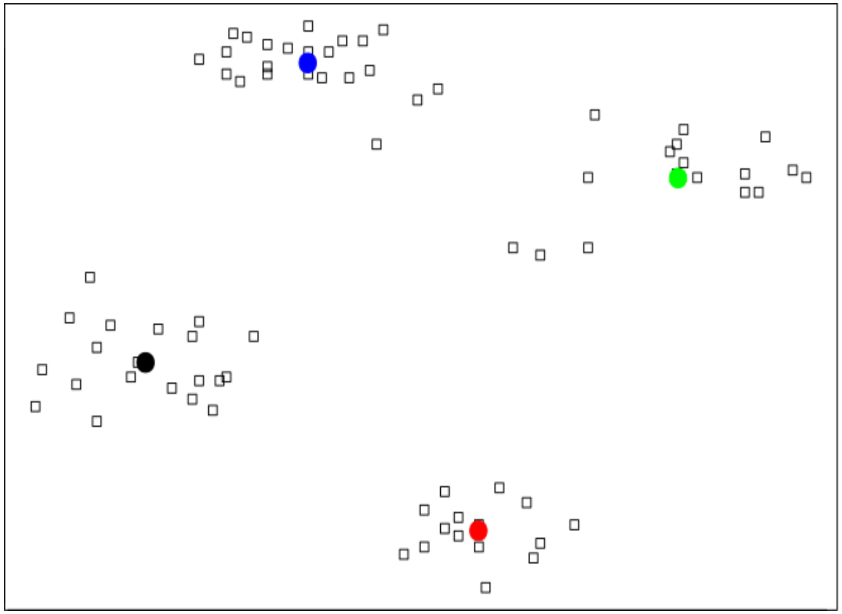
n = 10 → 10000

gap (%) < 10^{-6} → 10^{-1}

K-means

K-means

Machine learning problem



Instance	k	OPT*	LS 4.0	GAP
ruspini	2	89337	89337,9	0,00%
	3	51063,4	51063,5	0,00%
	4	12881	12881,1	0,00%
	5	10126,7	10126,8	0,00%
	6	8575,41	8670,86	1,11%
	7	7126,2	7159,13	0,46%
	8	6149,64	6158,26	0,14%
	9	5181,64	5277,11	1,84%
	10	4446,28	4856,98	9,24%
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%
glass	20	114,646	120,048	4,71%
	30	63,2478	74,1251	17,20%
	40	39,4983	58,3912	47,83%
	50	26,7675	52,4679	96,01%

*[Aloise et al. 2012]

Conclusion

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
- Black box optimization

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