Mathematical programming by Local Search

T. Benoist, J. Darlay, B. Estellon, F. Gardi, R. Megel
Innovation 24

Large industrial group with businesses in construction, telecom, media

www.bouygues.com

Operation Research subsidiary of the Bouygues group


Flagship product of Innovation 24

www.localsolver.com
LocalSolver

Why?
Practical observations

What is the most powerful tool provided by OR today?
→ Mixed Integer Linear Programming (MIP)
  • Simple and generic formalism
  • Easy-to-use solvers: “model-and-run” approach
  • Now an indispensable tool for practitioners
  • Constraint Programming (CP) is following the way

What do practitioners when MIP/CP solvers are ineffective?
→ Local Search (LS)
  • Core principle: improving the incumbent by exploring neighborhoods
  • Provides quality solutions in minutes
  • Extra costs (development, maintenance)
Our goals

A solver aligned with enterprise needs

- Provides high-quality solutions in seconds
- Scalable: tackles problems with millions of decisions
- Proves infeasibility or optimality when possible (best effort)

A solver aligned with practitioner needs

- « Model & Run »
  - Simple mathematical modeling formalism
  - Direct resolution: no need of complex tuning
- Full-version free trials with support
- Competitive pricing

http://www.localsolver.com/pricing.html

Free for academics
LocalSolver

Quick tour
Classical knapsack

8 items to pack in a sack: maximize the total value of items while not exceeding a total weight of 102 kg

```javascript
function model() {
    // 0-1 decisions
    x_0 <- bool(); x_1 <- bool(); x_2 <- bool(); x_3 <- bool();
    x_4 <- bool(); x_5 <- bool(); x_6 <- bool(); x_7 <- bool();

    // weight constraint
    knapsackWeight <- 10*x_0 + 60*x_1 + 30*x_2 + 40*x_3 + 30*x_4 + 20*x_5 + 20*x_6 + 2*x_7;
    constraint knapsackWeight <= 102;

    // maximize value
    knapsackValue <- 1*x_0 + 10*x_1 + 15*x_2 + 40*x_3 + 60*x_4 + 90*x_5 + 100*x_6 + 15*x_7;
    maximize knapsackValue;
}
```

You write the model: nothing else to do!

declarative approach = model & run
function model() {
   // 0–1 decisions
   x[0..7] <- bool();

   // weight constraint
   constraint knapsackWeight <= 102;

   // maximize value
   maximize knapsackValue;

   // secondary objective: minimize product of minimum and maximum values
   knapsackMinValue <- min[i in 0..7](x[i] ? values[i] : 1000);
   knapsackMaxValue <- max[i in 0..7](x[i] ? values[i] : 0);
   knapsackProduct <- knapsackMinValue * knapsackMaxValue;
   minimize knapsackProduct;
}

Nonlinear operators: prod, min, max, and, or, if-then-else, …

Lexicographic objectives
# Mathematical operators

<table>
<thead>
<tr>
<th>Arithmetic</th>
<th>Logical</th>
<th>Relational</th>
</tr>
</thead>
<tbody>
<tr>
<td>sum</td>
<td>prod</td>
<td>abs</td>
</tr>
<tr>
<td>min</td>
<td>max</td>
<td>dist</td>
</tr>
<tr>
<td>div</td>
<td>mod</td>
<td>exp</td>
</tr>
<tr>
<td>sqrt</td>
<td>log</td>
<td>pow</td>
</tr>
<tr>
<td>log</td>
<td>exp</td>
<td>tan</td>
</tr>
<tr>
<td>cos</td>
<td>sin</td>
<td>round</td>
</tr>
<tr>
<td>floor</td>
<td>ceil</td>
<td></td>
</tr>
</tbody>
</table>
function model() {
    // 0-1 decisions
    x[1..nbItems] <- bool();
    // weight constraint
    knapsackWeight <= sum[i in 1..nbItems](weights[i] * x[i]);
    // maximize knapsackValue
    knapsackValue <= sum[i in 1..nbItems](values[i] * x[i]);
    // close the model before solving it
    model.Close();
    LSPhase phase = localsolver.CreatePhase();
    phase.SetTimeLimit(1);
    localsolver.Solve();
}
LocalSolver

Let’s go inside
Car Sequencing

Scheduling cars on a production line

Objective = distributing options

- E.g. : at most 2 sun-roofs in any sequence of 5 cars («P/Q»)
- measure: in each window of length 5, penalty based on overcapacities = max(n-2,0) with $n$ the number of sun-roofs.

A *class* is a set of identical cars

- Her with 3 options A, B and C: AB is the class of cars featuring options A and B
Model

\[ X_{cp} = 1 \iff \text{The car in position } p \text{ belongs to class } c \]

\[
\begin{align*}
\text{X[c in 1..nbClasses][p in 1..nbPositions] } & \leftarrow \text{ bool}(); \\
\text{for[c in 1..nbClasses]} & \\
& \text{constraint } \text{sum[p in 1..nbPositions](X[c][p]) } = \text{ card[c];} \\
\text{for[p in 1..nbPositions]} & \\
& \text{constraint } \text{sum[c in 1..nbClasses](X[c][p]) } = 1; \\
\text{op[o in 1..nbOptions][p in 1..nbPositions] } & \leftarrow \\
& \text{or[c in 1..nbClasses : options[c][o]](X[c][p]);} \\
\text{nbVehicles[o in 1..nbOptions][j in 1..nbPositions-Q[o]+1] } & \leftarrow \\
& \text{sum[k in 1..Q[o]](op[o][j+k-1]);} \\
\text{violations[o in 1..nbOptions][j in 1..nbPositions-Q[o]+1] } & \leftarrow \text{ max(nbVehicles[o][j] } - \text{ P[o], 0 );} \\
\text{obj } & \leftarrow \text{ sum[o in 1..nbOptions][p in 1..nbPositions-Q[o]+1](violations[o][p]);} \\
\end{align*}
\]

That’s all!
Solving

How does LocalSolver solves this model?

1. Find an initial solution (here a random assignment of cars)
2. Apply generic moves
Small-neighborhood moves

Classical moves for Boolean Programming: “k-flips”

- Moves lead in majority to infeasible solutions
- Feasibility is hard to recover, implying a slow convergence
- Then no solver integrates an effective “pure local search” approach

Our moves tend to preserve the feasibility

- Can be viewed as a destroy-and-repair approach
- Can be viewed as ejection chains in the constraint hypergraph
- Can be specific to special combinatorial structures (when detected)
How does LocalSolver solves this model?

1. Find an initial solution (here a random assignment of cars)
2. Apply generic moves

- Exchanges (2 cars)
- Exchanges involving 3 cars or more
- Simple change
- Etc.

Key points:
- Simple changes will be eliminated after a few seconds since they fail systematically.
- The global search strategy is a randomized simulated annealing (parameterized)
- LocalSolver launches several concurrent search (the number of threads is a parameter)
- Some moves will be focused on windows with overcapacities

http://www.localsolver.com/technology.html
LocalSolver

Benchmarks
Car sequencing in Renault’s plants

Some instances are public. This problem was submitted as ROADEF Challenge in 2005: http://challenge.roadef.org/2005/en

Example: instance 022_EP_ENP_RA_F_S22_J1

- Small instance: 80,000 variables, including 44,000 binary decisions
- State of the art: 3,109 obtained by a specific local search algorithm
- Best lower bound: 3,103

Results

- Gurobi 5.0: 3.116647e+07 in 10 min | 25,197 in 1 hour
- LocalSolver 3.0: 3,478 in 10 sec | 3,118 in 10 min
2012 ROADEF Challenge

Reassignment of processes to machines, with different kinds of constraints (mutual exclusion, resources, etc.)

More than 100,000 binary decisions
Only 1 day of work
LocalSolver qualified for final round (ranked 24/80)
Some results obtained on the hardest MIPLIB instances

- Lower objective is better
- 5 minutes time limit for both LocalSolver and MIP
- Models are not suitably modeled for LocalSolver

<table>
<thead>
<tr>
<th>Problem</th>
<th>Variables</th>
<th>LS 3.1</th>
<th>MIP</th>
</tr>
</thead>
<tbody>
<tr>
<td>ds-big</td>
<td>4.9 M</td>
<td>9 844</td>
<td>62 520</td>
</tr>
<tr>
<td>ivu06-big</td>
<td>27.0 M</td>
<td>479</td>
<td>9 416</td>
</tr>
<tr>
<td>ivu52</td>
<td>2.5 M</td>
<td>4 907</td>
<td>16 880</td>
</tr>
<tr>
<td>mining</td>
<td>5.3 M</td>
<td>- 65 720 600</td>
<td>902 969 000</td>
</tr>
<tr>
<td>ns1853823</td>
<td>1.1 M</td>
<td>2 820 000</td>
<td>4 670 000</td>
</tr>
<tr>
<td>rmine14</td>
<td>1.3 M</td>
<td>- 3 470</td>
<td>- 171</td>
</tr>
<tr>
<td>rmine21</td>
<td>6.7 M</td>
<td>- 3 658</td>
<td>- 185</td>
</tr>
<tr>
<td>rmine25</td>
<td>14.0 M</td>
<td>- 3 052</td>
<td>- 161</td>
</tr>
</tbody>
</table>
LocalSolver

Business cases
Business cases

- Supply Chain Optimization
- Workforce planning
- TV Media Planning
- Logistic clustering
- Street lighting maintenance planning
- Network deployment planning
- Energy optimization for tramway lines
- Placement of nuclear fuel assemblies in pools
- Painting shop scheduling
- Transportation of equipment
Global Supply Chain

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

A challenging context for LocalSolver

- 20,000,000-variable model including 3 millions binaries
- A rich model involving setup costs, delivery times, packaging...
- A vain attempt to solve the problem with MIP solvers
- LocalSolver finds a high-quality solution in minutes
Street lighting

Tomorrow at 8:30 in this room

Long Term Planning with LocalSolver
by Romain Megel
LocalSolver

Roadmap
Integrating MIP, CP, SAT techniques with LS into an all-in-one solver for large-scale mixed-variable non-convex optimization
LocalSolver
mathematical programming by local search

www.localsolver.com
Growing community!