How LocalSolver qualified with a 100-lines model?
LocalSolver

LocalSolver in a nutshell
What is LocalSolver?

The first math programming solver based on local search

- Pure model-and-run approach: no extra code to write
- Solve highly nonlinear 0-1 models
- Scale up to 10 million decision variables

\[ \rightarrow \text{Solve problems intractable with IP/CP/SAT solvers} \]

Portable software

- Fully portable: Windows, Linux, Mac OS (x86, x64)
- Light object-oriented APIs: a few classes only
- Lightweight APIs available for C++, Java, .Net

Comes with an innovative modeling language for fast prototyping
Weaknesses of tree search

- Not suited to reach quickly good “integer feasible solutions”
- Designed to prove optimality
- Exponential time: not scalable (the best IP solvers still fail to find feasible solutions for real-life instances with 10,000 binaries)
- An incomplete tree search is not more optimal than a local search

Practitioners need:

- A solver which provides high-quality solutions in seconds
- A scalable solver which tackle problems with millions of variables
- A solver which proves optimality of infeasibility when possible
How it works?

3 main layers:

LS solver must work as a LS practitioner works

1. **Incremental algorithm, sublinear evaluation**
   exploit the invariants induced by the mathematical operators
   \[ \rightarrow \] thousands of solutions explored each second

2. **Structured moves that maintain feasibility**
   moves performed on the hypergraph of decisions and constraints (ejection chains, cycles, ...)

3. **Heuristic and search strategy**
   heuristic based on simulated annealing to get out of local optima
   multithreading to ensure faster convergence and robustness
The EURO/Roadef Challenge

LocalSolver modeling
Model in 3 parts

Each step corresponds to a specific function in the modeler

1. Read the input data
   open file, read the initial assignment, read resources, groups, ...

2. Write the model
   Declare boolean variables, constraints, objectives, ...

3. Parameterize the resolution
   Set time or iteration limit, load an initial solution.

100-lines, 1 day of work
How to model with LocalSolver?

1. **Declare the decision variables**
   A decision is a variable you can't compute from other variables

2. **Declare the constraints of your problem**

3. **Declare the objectives**
Assignment of processes to machines
These decisions completely determine the solution

```
// 0-1 decisions
x[0..nbProcesses-1][0..nbMachines-1] <- bool();

\[ x_{pm} = 1 \iff \text{process } p \text{ on machine } m \]
```

Each process must be assigned to a single machine

```
for [p in 0..nbProcesses-1]
  constraint sum[m in 0..nbMachines-1](x[p][m]) == 1;
```

Capacity constraints

```
for [m in 0..nbMachines-1][r in 0..nbResources-1] {
  u[m][r] <- sum[p in 0..nbProcesses-1](require[p][r] * x[p][m]);
  constraint u[m][r] <= capacity[m][r];
}
```
Other constraints

Conflict constraints
processes of the same service must run on distinct machines

```plaintext
for [s in 0..nbServices-1][m in 0..nbMachines-1]
  constraint sum[p in processByService[s]](x[p][m]) <= 1;
```

Spread constraints
processes of the same service must spread on a set of locations

```plaintext
for [s in 0..nbServices-1] {
  coveredLocations[s] <= sum[l in 0..maxLocation](
    or[p in processByService[s]][m in machineByLocation[l]](x[p][m]));
  constraint coveredLocations [s] >= spread[s];
}
```
Objectives

Objective : Load cost

\[
\text{loadCost}[r \in 0..\text{nbResources}-1] \leftarrow \sum[m \in 0..\text{nbMachines}-1](\max(\text{u}[m][r] - \text{safety}[m][r], 0));
\]

\[
\text{totalLoadCost} \leftarrow \sum[r \in 0..\text{nbResources}-1](\text{rweight}[r] \times \text{loadCost}[r]);
\]

Objective : Balance cost

\[
\text{a}[m \in 0..\text{nbMachines}-1][r \in 0..\text{nbResources}-1] \leftarrow \text{capacity}[m][r] - \text{u}[m][r];
\]

\[
\text{for } [b \in 0..\text{nbBalances}-1] \{
\begin{align*}
\text{r1} &= \text{resource1}[b]; \\
\text{r2} &= \text{resource2}[b]; \\
\text{tg} &= \text{target}[b]; \\
\text{balanceCost}[b] &\leftarrow \sum[m \in 0..\text{nbMachines}-1](\max(\text{tg} \times \text{a}[m][r1] - \text{a}[m][r2], 0));
\end{align*}
\}
\]

\[
\text{totalBalanceCost} \leftarrow \sum[b \in 0..\text{nbBalances}-1](\text{bweight}[b] \times \text{balanceCost}[b]);
\]

Objective : Process move cost

\[
\text{processMoveCost} \leftarrow \sum[p \in 0..\text{nbProcesses}-1](\text{pcost}[p] \times \text{not}(\text{x}[p][\text{initialMachine}[p]]));
\]
### Objectives

**Objective : Service move cost**

```plaintext
for [s in 0..nbServices-1] {
    nbMoved[s] <- sum[p in 0..nbProcesses-1 : service[p] == s](!x[p][initialMachine[p]]);
    serviceMoveCost <- max[s in 0..nbServices-1](nbMoved[s]);
}
```

**Objective : Machine move cost**

```plaintext
for [p in 0..nbProcesses-1] {
    m0 = initialMachine[p];
    machineMoveCost[p] <- sum[m in 0..nbMachines-1 : m != m0](mcost[m0][m] * x[p][m]);
}
```

**Total cost**

```plaintext
obj <- totalLoadCost + totalBalanceCost + wpmc * processMoveCost + wsmc * serviceMoveCost + wmmc * totalMachineMoveCost;
```

**minimize** obj;
Qualification results

<table>
<thead>
<tr>
<th>instance</th>
<th>variables</th>
<th>binaries</th>
<th>solution</th>
<th>best</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1-2</td>
<td>1,812,044</td>
<td>100,000</td>
<td>787,434,004</td>
<td>777,532,896</td>
</tr>
<tr>
<td>A1-3</td>
<td>1,423,438</td>
<td>100,000</td>
<td>583,014,803</td>
<td>583,005,717</td>
</tr>
<tr>
<td>A1-4</td>
<td>753,404</td>
<td>50,000</td>
<td>272,304,480</td>
<td>252,728,589</td>
</tr>
<tr>
<td>A1-5</td>
<td>229,213</td>
<td>12,000</td>
<td>727,578,410</td>
<td>727,578,309</td>
</tr>
<tr>
<td>A2-1</td>
<td>1,415,324</td>
<td>100,000</td>
<td>5,934,529</td>
<td>198</td>
</tr>
<tr>
<td>A2-2</td>
<td>3,769,381</td>
<td>100,000</td>
<td>1,163,672,839</td>
<td>816,523,983</td>
</tr>
<tr>
<td>A2-3</td>
<td>3,843,977</td>
<td>100,000</td>
<td>1,555,764,432</td>
<td>1,306,868,761</td>
</tr>
<tr>
<td>A2-4</td>
<td>1,537,771</td>
<td>50,000</td>
<td>2,089,185,551</td>
<td>1,681,353,943</td>
</tr>
<tr>
<td>A2-5</td>
<td>1,556,017</td>
<td>50,000</td>
<td>575,691,649</td>
<td>336,170,182</td>
</tr>
</tbody>
</table>

100-lines model, 1 day of work, 11 million solutions explored in 5 min
LocalSolver qualified (25/80)
The EURO/Roadef Challenge

For the B instances?
Boolean model has its limits
  • With 4GB of RAM, LocalSolver tackles B1, B2 & B3 instances
  • For other instances, a machine with 40GB of RAM is required

Solution: decompose the model
  • Take a subset of machines (20,000 decisions)
  • Optimize with LocalSolver on this subset for 1 second
  • Repeat the operation 300 times

Same model, one more day of work
15 million solutions explored in 5 min
## Final stage results

<table>
<thead>
<tr>
<th>instance</th>
<th>machines</th>
<th>processes</th>
<th>LS 2.0 direct</th>
<th>LS 2.0 based</th>
</tr>
</thead>
<tbody>
<tr>
<td>B1</td>
<td>100</td>
<td>5,000</td>
<td>4,443,248,534</td>
<td>3,997,678,428</td>
</tr>
<tr>
<td>B2</td>
<td>100</td>
<td>5,000</td>
<td>1,368,865,436</td>
<td>1,163,729,413</td>
</tr>
<tr>
<td>B3</td>
<td>100</td>
<td>20,000</td>
<td>351,813,894</td>
<td>266,280,383</td>
</tr>
<tr>
<td>B4</td>
<td>1,000</td>
<td>10,000</td>
<td>5,796,304,487</td>
<td>4,682,013,089</td>
</tr>
<tr>
<td>B5</td>
<td>100</td>
<td>40,000</td>
<td>1,048,102,941</td>
<td>1,015,121,228</td>
</tr>
<tr>
<td>B6</td>
<td>200</td>
<td>40,000</td>
<td>9,537,599,318</td>
<td>9,550,921,033</td>
</tr>
<tr>
<td>B7</td>
<td>4,000</td>
<td>40,000</td>
<td>RAM exploded</td>
<td>16,340,742,734</td>
</tr>
<tr>
<td>B8</td>
<td>100</td>
<td>50,000</td>
<td>1,323,157,749</td>
<td>1,316,777,967</td>
</tr>
<tr>
<td>B9</td>
<td>1,000</td>
<td>50,000</td>
<td>RAM exploded</td>
<td>15,959,363,471</td>
</tr>
<tr>
<td>B10</td>
<td>5,000</td>
<td>50,000</td>
<td>RAM exploded</td>
<td>19,314,990,649</td>
</tr>
</tbody>
</table>