LocalSolver

mathematical programming by local search

www.localsolver.com

Thierry Benoist Julien Darlay Bertrand Estellon
Frédéric Gardi Romain Megel Karim Nouioua

Focused on business needs

A solver aligned with enterprise needs

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

A solver aligned with practitioner needs

- “Model & Run”
  - Simple mathematical modeling formalism
  - Direct resolution: no need of complex tuning
- Simple and transparent pricing

Free for academics

LocalSolver

Main features

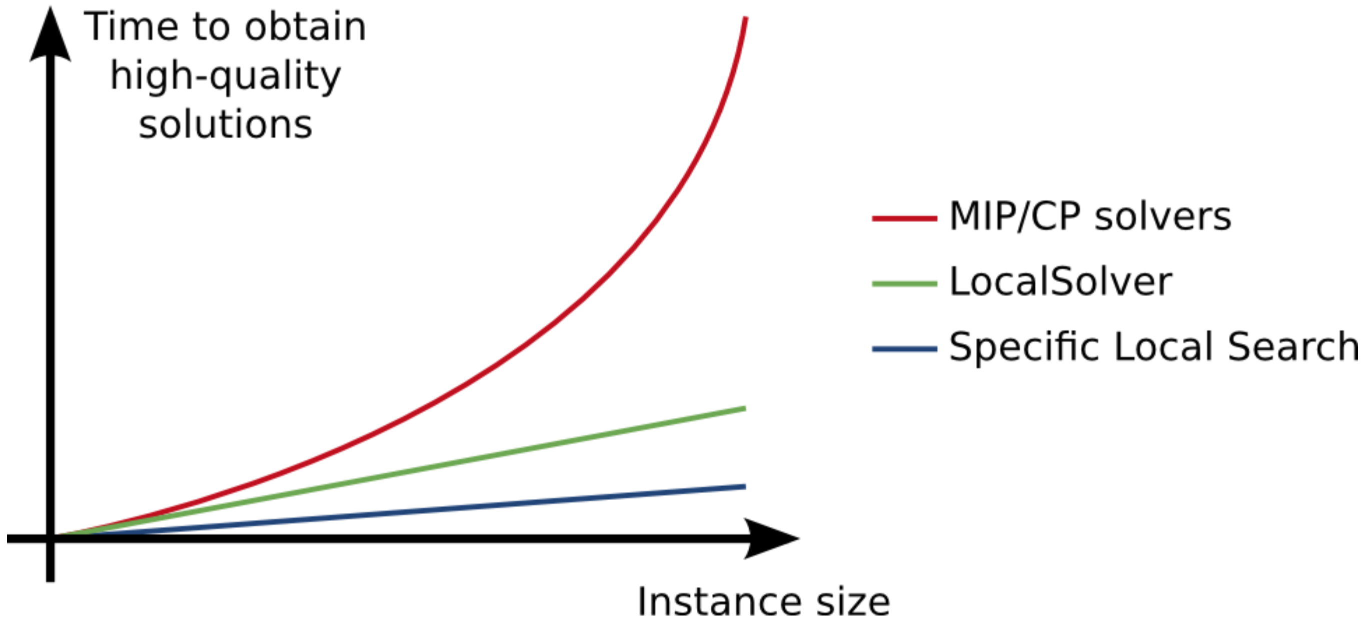
New-generation solver

- **Computing good-quality solutions by local search**
- Computing lower bounds **separately** (inference, relaxation, cuts)

High-end software

- An innovative modeling language for fast prototyping
- Lightweight object-oriented APIs: a few classes only
- Reliable and robust: quality through drastic continuous integration
- Fully portable: Windows, Linux, Mac OS (x86, x64)
- Reactive support, realized by developers themselves (even for academics)

Why local search?



Technology

Autonomous local search

- Generic moves based on decisions/constraints hypergraph
- Incremental evaluation: **million moves per minute**
- Adaptive simulated annealing through learning
- Multithreaded search: ready for many-core world

Efficient C++ implementation

- Preprocessing: model reduction & reformulation
- Low-level cache-aware code optimization
- Highly-optimized memory management

Knapsack

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

Binary decision variables

Integer intermediate variables

```
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;
}
```

The user writes the model: nothing else to do!

declarative approach = model & run

LocalSolver

Multiobjective knapsack

```
function model() {
  // 0-1 decisions
  x[0..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;

  // 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

Arithmetic		Logical	Relational	Hybrid
sum	prod	not	==	if
min	max	and	!=	array + at
div	mod	or	<=	
abs	sqrt	xor	>=	
			<	
			>	

Modeling APIs

```
function model() {
  // 0-1 decisi
  x[1..nbItems]

  // weight con
  knapsackWei
  constraint kn

  // maximize
  knapsackValu
  maximize kn
}
```

C++

```
#include "localsolver.h"

using namespace localsolver;
```

Java

```
import localsolver.*;

public class Toy {
```

C#

```
using System;
using localsolver;

public class Toy
{
  static void Main()
  {
    int[] weights = {10, 60, 30, 40, 30, 20, 20, 2};
    int[] values = {1, 10, 15, 40, 60, 90, 100, 15};

    LocalSolver localsolver = new LocalSolver();
    LSModel model = localsolver.GetModel();

    // 0-1 decisions
    LSExpression[] x = new LSExpression[8];
    for (int i = 0; i < 8; i++)
      x[i] = model.CreateExpression(LSOperator.Bool);

    // knapsackWeight <- 10*x0 + 60*x1 + 30*x2 + 40*x3 + 30*x4 + 20*x5 + 20*x6 + 2*x7;
    LSExpression knapsackWeight = model.CreateExpression(LSOperator.Sum);
    for (int i = 0; i < 8; i++)
      knapsackWeight.AddOperand(model.CreateExpression(LSOperator.Prod, weights[i], x[i]));

    // knapsackWeight <= 102;
    model.AddConstraint(model.CreateExpression(LSOperator.Leq

    // knapsackValue <- 1*x0 + 10*x1 + 15*x2 + 40*x3 + 60*x4 + 90*x5 + 100*x6 + 15*x7;
    LSExpression knapsackValue = model.CreateExpression(LSOperator.Sum);
    for (int i = 0; i < 8; i++)
      knapsackValue.AddOperand(model.CreateExpression(LSOperator.Prod, values[i], x[i]));

    // maximize knapsackValue;
    model.AddObjective(knapsackValue, LSObjectiveDirection.Maximize);

    // close the model before solving it
    model.Close();
    LSPhase phase = localsolver.CreatePhase();
    phase.SetTimeLimit(1);
    localsolver.Solve();
```

createExpression

addOperand

Car sequencing in Renault's plants

Scheduling cars along painting and assembly lines

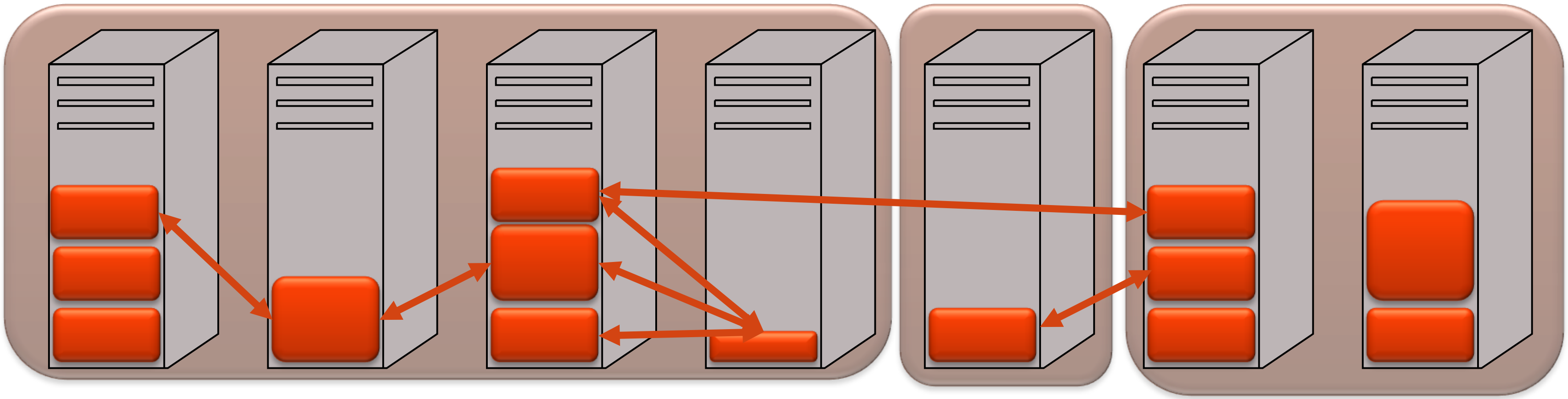
- Classical car sequencing = space car options along the line
- No more than K consecutive cars with the same color
- Minimize the number of paint color changes as secondary objective

Large-instances to tackle

- 1300 cars to sequence → 400 000 binary decisions
- MIP or CP solvers unable to find feasible solutions after hours
- LocalSolver provides much better solutions than Renault in seconds

2012 ROADEF/EURO Challenge

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



More than 100 000 binary decisions

LSP model with 200 lines, written in 1 day of work

LocalSolver qualified for final round (ranked 25/80)

Application guide

When using LocalSolver?

- MIP solvers find no (quality) solution
- MIP solvers find quality solutions but too slowly
- Writing MIP models is complicated due to nonlinearities
- CP seems to be a better choice than MIP

LocalSolver is suited for:

- Nonlinear assignment: car sequencing, frequency assignment
- Packing & Covering: media planning, machine scheduling, graph partitioning
- Facility location, logistic clustering, telecom network optimization
- Workforce scheduling, group planning, nurse rostering

Customers & Partners

LocalSolver

Academic Users

technische universität dortmund

東北大学
TOHOKU UNIVERSITY

BOĞAZİÇİ UNIVERSITY

THE UNIVERSITY OF EDINBURGH

Université catholique de Louvain
UCL

UCC
Coláiste na hOllscoile Corcaigh, Éire
University College Cork, Ireland

UNSW
THE UNIVERSITY OF NEW SOUTH WALES
SYDNEY · CANBERRA · AUSTRALIA

DONGHUA UNIVERSITY

UPPSALA UNIVERSITET

UNIVERSITÀ DEGLI STUDI DI GENOVA

Universidad de Concepción
Chile

POLITECNICO DI MILANO

Université de Valenciennes et du Hainaut-Cambrésis

université angers

UNIVERSITY of LIMERICK
OLLSCOIL LUIMNIGH

The University of Nottingham

Universidade de São Paulo
Brasil

UNI GRAZ

THE HONG KONG POLYTECHNIC UNIVERSITY
香港理工大學

GEORG-AUGUST-UNIVERSITÄT GÖTTINGEN

uff Universidade Federal Fluminense

Fast-growing community!

TARBIAT MODARES UNIVERSITY

uco UNIVERSITÉ CATHOLIQUE DE L'OUEST

EISTI Ecole Internationale des Sciences du Traitement de l'Information

ONERA THE FRENCH AEROSPACE LAB

THE UNIVERSITY OF TOKYO

ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ ΗΠΕΙΡΟΥ
EPIRUS INSTITUTE OF TECHNOLOGY
ARTA , ZIP CODE : 47100 , TEL NUMBER : +30 26810 50000

UNIVERSITÉ DE NANTES

CSIRO

UPC UNIVERSITAT POLITÈCNICA DE CATALUNYA BARCELONATECH

भारतीय प्रौद्योगिकी संस्थान खड़गपुर
Indian Institute of Technology Kharagpur

KING SAUD UNIVERSITY

UNIVERSITÄT DUISBURG ESSEN

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

ENSTA ParisTech

STUDIORUM UNIVERSITAS FLORENTINA

universität wien

IFSTTAR

Universidad Tecnológica de Bolívar

JFLI Japanese-French Laboratory for Informatics

SIGILLUM UNIVERSITATIS DAKARENSIS LUX MEA LEX

DAUPHINE UNIVERSITÉ PARIS

Linköping University

Grenoble INP

Roadmap

LocalSolver 3.0 : October 2012

- Floating-point coefficients
- New math operators: log, exp, pow, cos, sin, tan
- Major performance improvements: new local-search moves
- Improved preprocessing (model reduction & reformulation)

LocalSolver 4.0 : March 2013

- Binary + **continuous** decisions
- Local-search moves on continuous decisions
- Better capabilities for proving optimality or infeasibility

→ **Large-scale mixed-variable nonlinear programming (MINLP)**

For more details

T. Benoist, B. Estellon, F. Gardi, R. Megel, K. Nouioua.
LocalSolver 1.x: a black-box local-search solver for 0-1 programming.
4OR, A Quarterly Journal of Operations Research 9(3), pp. 299-316.

http://www.localsolver.com

fgardi@localsolver.com