Optimization Software.
Heuristic Solvers for VRPs.

Introduction to Local Solver.

Ruslan Sadykov

INRIA Bordeaux—Sud-Ouest

28 October 2022

LocalSolver : an optimization solver

- Originally, a local search meta-heuristic solver
- Today, it can provide lower bounds in some cases
- A model-and-run solver

Setting up Python interface

```
pip install localsolver -i https ://pip.localsolver.com
```

LocalSolver : knapsack example

```python
import localsolver
with localsolver.LocalSolver() as ls :
    model = ls.model

    # Decision variables x[i]
    x = [model.bool() for i in range(nb_items)]

    # Weight constraint
    knapsack_weight = model.sum(x[i] * weights[i]
                                for i in range(nb_items))
    model.constraint(knapsack_weight <= knapsack_bound)

    # Maximize value
    knapsack_value = model.sum(x[i] * values[i]
                               for i in range(nb_items))
    model.maximize(knapsack_value)

    model.close()
    ls.param.time_limit = 20

    ls.solve()
```

Full example : `examples/knapsack/knapsack.py`

LocalSolver : variables, constraints, objectives

Variables

Boolean, floating-point, integer, set, list

Constraints

Arithmetic, relational, logical, conditional, set related, « element-like »

```
# These two formulations are equivalent
model.constraint(knapsackWeight <= 102)
weightCst = knaspackWeight <= 102
model.constraint(weightCst)
```

Objectives

Can be hierarchical

```
model.maximize(revenues)
model.minimize(resources)
model.maximize(desiderata)
```

Collection (list and set) variables

- Defined by an unique constant operand n
- A value of a set (or list) variable is an (ordered) collection of pairwise different integers within domain $[0, n-1]$
- A set of list do not necessarily contain all values in $[0, n-1]$

Special operators

- `count` (number of elements in a collection)
- `contains` (a collection contains or not an element)
- `disjoint` (collections are disjoint or not)
- `cover` (collections cover all elements or not)
- `partition` (collections form a partition of all elems or not)
- `at` (element at a position of a list)
- `indexOf` (position of an element in a list, or -1)

Usage

Lists → routing problems
Sets → packing problems

Travelling Salesman Problem

```python
model = ls.model

# A list variable : cities[i] is the index of the ith city in the tour
cities = model.list(nb_cities)

# All cities must be visited
model.constraint(model.count(cities) == nb_cities)

# Create a LocalSolver array for the distance matrix in order to be able to access it with "at" operators.
distance_array = model.array(distance_weight)

# Minimize the total distance
dist_selector = model.lambda_function(
  lambda i : model.at(distance_array, cities[i-1], cities[i]))
obj = (model.sum(model.range(1, nb_cities), dist_selector)
  + model.at(distance_array, cities[nb_cities-1], cities[0]))
model.minimize(obj)

model.close()
ls.param.time_limit = 5
ls.solve()
```

Full example : `examples/tsp/tsp.py`

Capacitated Vehicle Routing Problem

Model : `docs/exampletour/vrp.html`
Full example : `examples/cvrp/cvrp.py`

Capacitated Vehicle Routing Problem with Time Windows

Model : `docs/exampletour/vrptw.html`
Full example : `examples/cvrptw/cvrptw.py`

Multi-Depot Vehicle Routing Problem (Location-Routing)

Model : `docs/exampletour/location-routing-problem-lrp.html`
Full example : `examples/location_routing_problem/location_routing_problem.py`