Set-Based Modeling in LocalSolver 6.0

www.localsolver.com

Motivations

Modeling approaches for
the *Traveling Salesman Problem*

Mixed-Integer Programming

With an **exponential number** of constraints

Minimise $\sum_{\substack{i,j \\ i \neq j}} c_{ij} x_{ij}$

Conventional Formulation (C) (Dantzig, Fulkerson and Johnson (1954))

$$\sum_{\substack{j \\ j \neq i}} x_{ij} = 1 \qquad \forall\ i \in N$$

$$\sum_{\substack{i \\ i \neq j}} x_{ij} = 1 \qquad \forall\ j \in N$$

$$\sum_{\substack{i,j \in M \\ i \neq j}} x_{ij} \leq |M| - 1 \qquad \forall\ M \subset N \text{ such that } \{1\} \notin M, |M| \geq 2$$

→ Iterative procedure to add subtour elimination constraints

Variant with $O(n^2)$ variables and constraints

SINGLE COMMODITY FLOW (**F1**) (Gavish and Graves (1978))

Both constraints are retained but we also introduce (continuous) variables:

y_{ij} = 'Flow' in an arc (i,j) $i \neq j$

and constraints:

$$y_{ij} \leq (n-1) x_{ij} \qquad \forall\ i,j \in N,\ i \neq j$$

$$\sum_{\substack{j \\ j \neq 1}} y_{1j} = n - 1$$

$$\sum_{\substack{i \\ i \neq j}} y_{ij} - \sum_{\substack{k \\ i \neq k}} y_{jk} = 1 \qquad \forall\ j \in N - \{1\}$$

In Orman & Williams : A survey of different integer programming formulations of the TSP

A compact model based on positions

```
/* Declares the optimization model. */
function model() {
    // x[i][j] equal to 1 if city j is ith visited city in the tour
    x[0..nbCities-1][0..nbCities-1] <- bool();

    // one city per position i
    for [i in 0..nbCities-1] constraint sum[j in 0..nbCities-1](x[i][j]) == 1;
    // one position per city j
    for [j in 0..nbCities-1] constraint sum[i in 0..nbCities-1](x[i][j]) == 1;

    //city[i] is the city at position i in the tour
    city[i in 0..nbCities-1] <- sum[j in 0..nbCities-1](j*x[i][j]);

    // Distance of the arc reaching the ith city
    distance[0] <- distanceWeight[city[nbCities - 1]][city[0]];
    distance[i in 1..nbCities - 1] <- distanceWeight[city[i - 1]][city[i]];

    // Minimize the total distance
    obj <- sum[i in 0..nbCities - 1](distance[i]);

    minimize obj;
}
```

→ A polynomial-size model (not an iterative procedure)

→ No artificial variables and constraints

→ But still based on binary decisions

Natural Modeling

As a permutation

The Traveling Salesman Problem (TSP)

TSP: Given a list of cities and their pairwise distances, find a shortest possible tour that visits each city exactly once.

Objective: find a permutation $a_1,\ldots,a_n$ of the cities that minimizes

$$d(a_1, a_2) + d(a_2, a_3) + \ldots + d(a_{n-1}, a_n) + d(a_n, a_1)$$

where $d(i, j)$ is the distance between cities i and j

An optimal TSP tour through Germany's 15 largest cities

In Kenneth R. Rosen: Permutations and Combinations.

Reference modeling

Garey & Johnson

[ND22] TRAVELING SALESMAN

INSTANCE: Set C of m cities, distance $d(c_i, c_j) \in Z^+$ for each pair of cities $c_i, c_j \in C$, positive integer B.

QUESTION: Is there a tour of C having length B or less, i.e., a permutation $< c_{\pi(1)}, c_{\pi(2)}, \ldots, c_{\pi(m)} >$ of C such that

$$\left(\sum_{i=1}^{m-1} d(c_{\pi(i)}, c_{\pi(i+1)}) \right) + d(c_{\pi(m)}, c_{\pi(1)}) \leqslant B \quad ? \qquad .$$

Set-based modeling

Innovative modeling concepts
for routing & scheduling problems

List Variables

Structured decisional operator list(n)

- Order a subset of values in domain {0, …, n-1}
- Each value is unique in the list

Classical operators to interact with "list"

- count(u): number of values selected in the list
- at(u,i) or u[i]: value at index i in the list
- indexOf(u,v): index of value v in the list
- contains(u,v): equivalent to "indexOf(u,v) != -1"
- disjoint(u1, u2, …, uk): true if u1, u2, …, uk are pairwise disjoint
- partition(u1, u2, …, uk): true if u1, u2, …, uk induce a partition of {0, …, n-1}

Traveling salesman

```
function model() {
  x <- list(N) ; // order n cities {0, ..., n-1} to visit
  constraint count(x) == N; // exactly n cities to visit
  minimize sum[i in 1..N-1]( Dist[ x[i-1] ][ x[i] ] )
                  + Dist[ x[N-1] ][ x[0] ] ; // minimize sum of traveled distances
}
```

TSP: Given a list of cities and their pairwise distances, find a shortest possible tour that visits each city exactly once.

Objective: find a permutation $a_1,\ldots,a_n$ of the cities that minimizes

$$d(a_1, a_2) + d(a_2, a_3) + \ldots + d(a_{n-1}, a_n) + d(a_n, a_1)$$

where $d(i, j)$ is the distance between cities i and j

An optimal TSP tour through Germany's 15 largest cities

Why not a single line model ?

```
constraint TSP(graph);
```

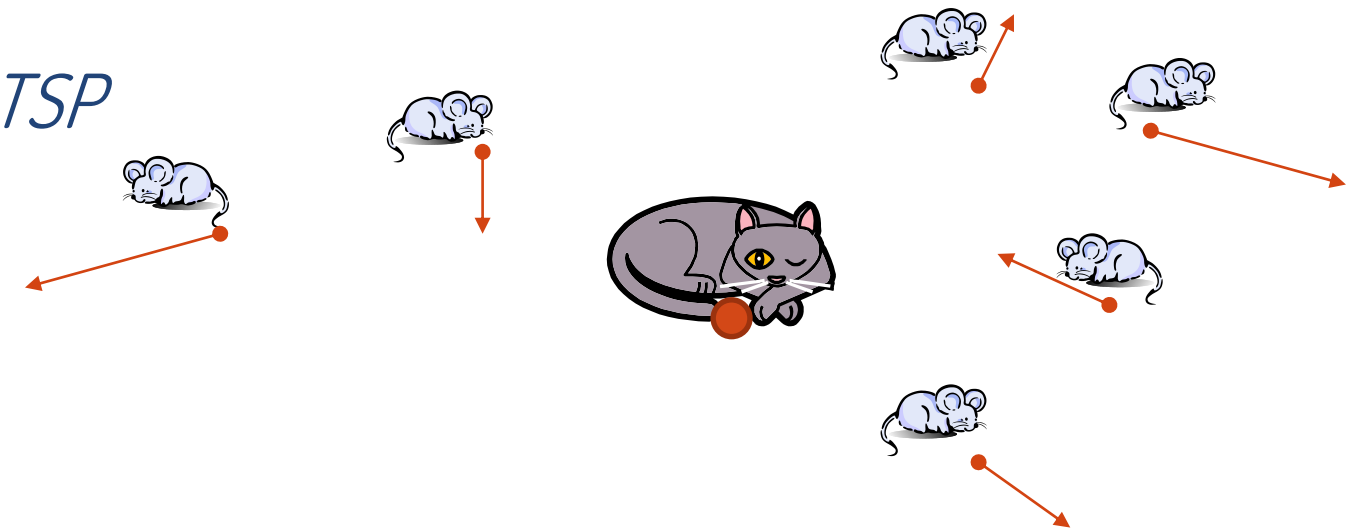
Real-world models do not fit in tight frameworks

$$\mathrm{Time}[A,B,T] = \frac{2(\alpha_x{}^2+\alpha_y{}^2)}{-2(\alpha_x\beta_x+\alpha_y\beta_y)+\sqrt{4(\alpha_x\beta_x+\alpha_y\beta_y)^2-4(\alpha_x{}^2+\alpha_y{}^2)(\beta_x{}^2+\beta_y{}^2-V^2)}} + T$$

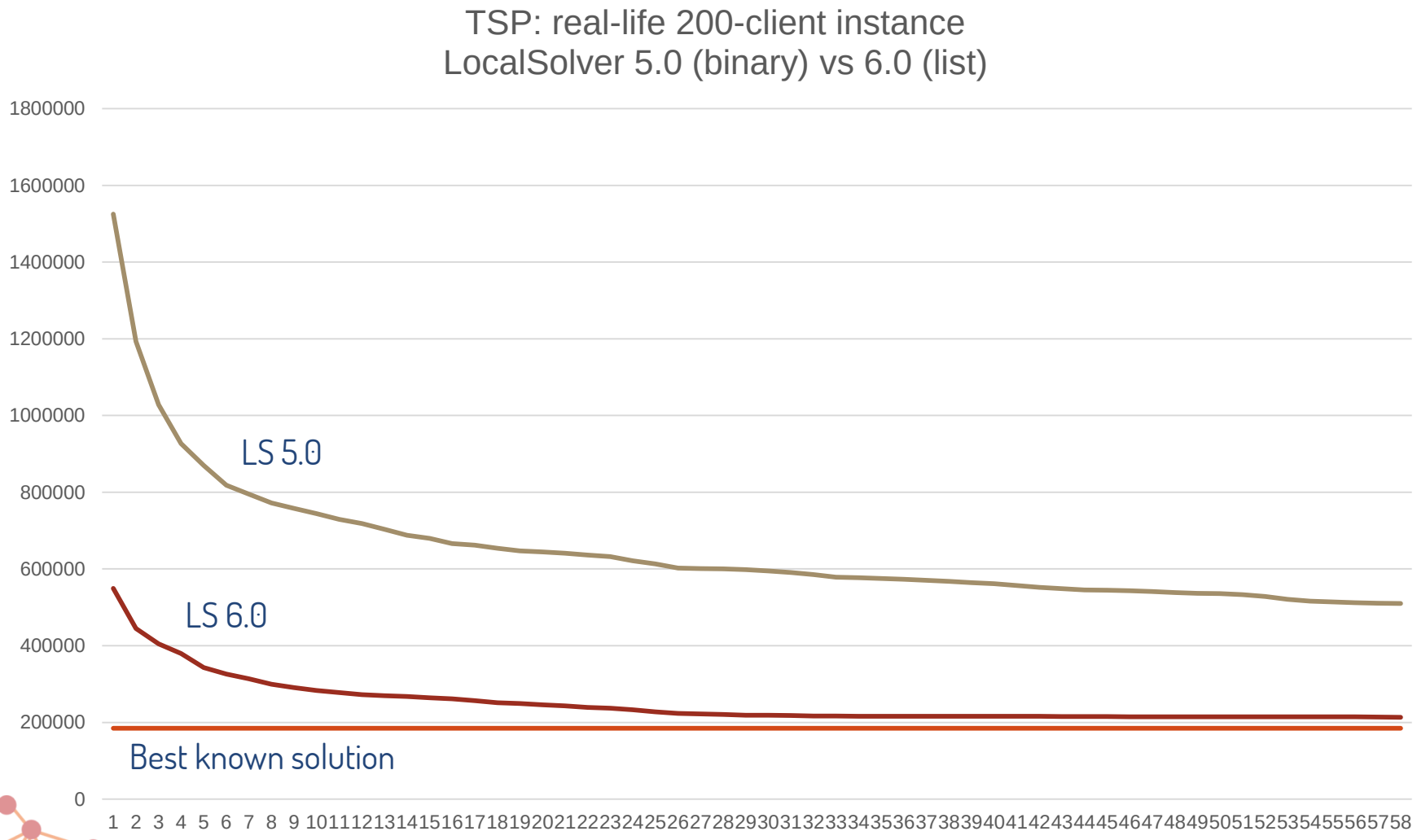
With $\alpha_x, \beta_x, \alpha_y, \beta_y$ function of A,B and T

```
constraint TSP(graph);
```
?

Kinetic TSP



Performance?



Vehicle routing

```
function model() {
  x[1..K] <- list(N) ; // for each truck, order the clients to visit
  constraint partition( x[1..K] ); // each client is visited once
  distances[k in 1..K] <- sum[i in 1..N-1]( dist( x[k][i-1], x[k][i]) )
            + dist( x[k][N-1], x[k][0] ); // traveled distance for each truck

  minimize sum[k in 1..K]( distances[k] ); // minimize total traveled distance
}
```

	TSP	VRP
Normal	Count(x)=N	partition(x[1..K])
Prize-collecting	maximize sum(…)	disjoint(x[1..K])

CVRP benchmarks

CVRP – instances A

- 32 to 80 clients, 10 trucks max.
- 27 instances
- 5 minutes of running time
- LS binary: almost infeasible
- LS list: 1 % avg. opt. gap

CVRP – instances X100-500

- 100 to 500 clients, 138 trucks max.
- 67 instances
- 5 minutes of running time
- LS binary: almost infeasible
- LS list: 9 % avg. opt. gap

CVRPTW benchmarks

CVRPTW – instances Solomon R100

- 101 to 112 clients, 19 trucks max.
- 13 instances
- 5 minutes of running time
- LS binary: N/A
- LS list: 3 % avg. opt. gap

CVRPTW – instances Solomon R200

- 201 to 208 clients, 4 trucks max.
- 8 instances
- 5 minutes of running time
- LS binary: N/A
- LS list: 8 % avg. opt. gap

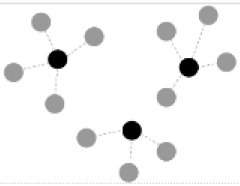
Beyond routing problems

Scheduling, planning, sequencing

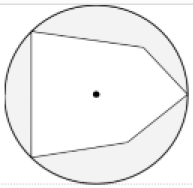
Example tour

Toy ★

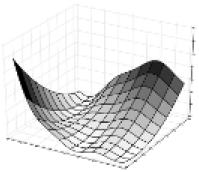
Knapsack ★



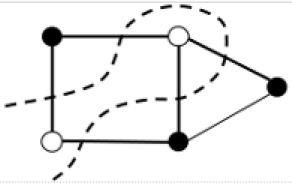
P-median ★



Smallest circle ★

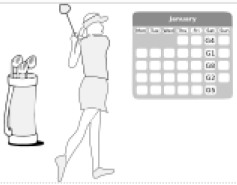


Branin function ★

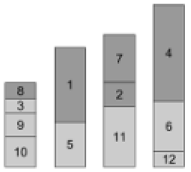


Max cut ★

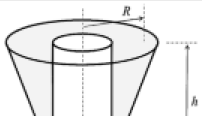
Car sequencing ★★

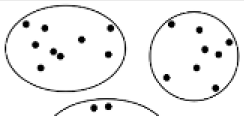


Social golfer ★★

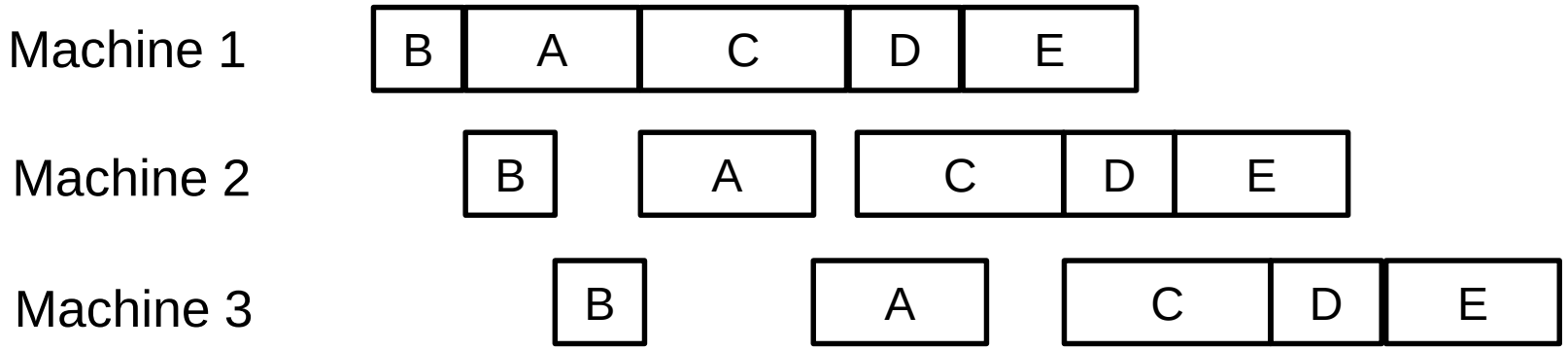


Steel mill slab design ★★





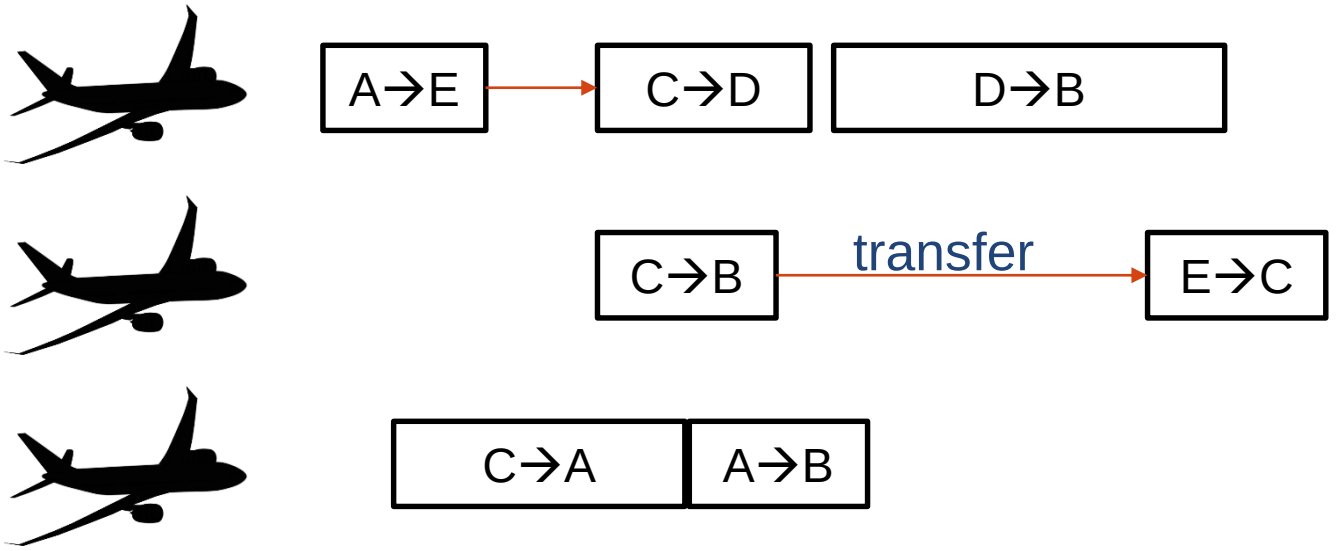
Flow-shop scheduling



Since we are looking for a permutation of jobs the model is straightforward with a single list variable

Planning

Flights to plane assignments



A solution is a partition of flights into K lists (one per plane)

The goal is to minimize the total transfer times

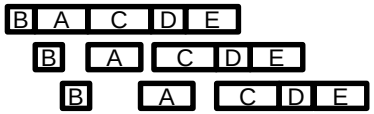
Conclusion

List Variables are a first step towards set-based modeling in LocalSolver

This higher level of modeling yields simple and compact models producing high quality solutions for

Routing

Scheduling



Planning

STELLAR

RCPSP

Leibniz Universität Hannover

Any other sequencing problem