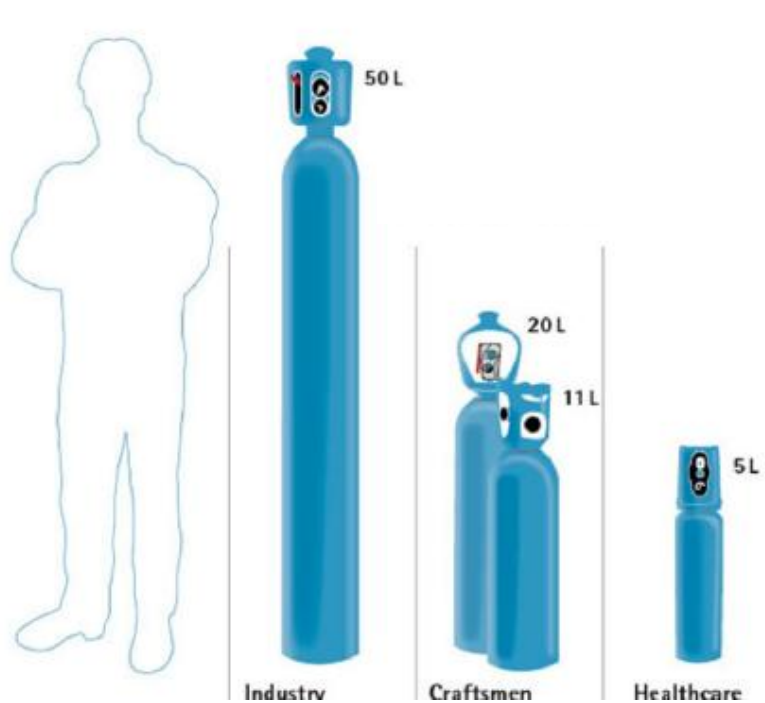
Tactical Optimization of cylinder distribution using Localsolver – from a user point of view

Troyes, 15/02/ 2013 l Michele Quattrone – Thierry Benoist l Applied Mathematics

Gaz Cylinders, a core business for Air Liquide

- From 1903, Air Liquide sends gas under pressure in a cylinder packaging.
- Gases can be pure (Oxygen, Acetylene, Nitrogen, Helium) or a mix (Air, Argon + Oxygen, etc). Application are many: Hospitals, Welding, Chemical Laboratories…



13M cylinders in circulation around the world in 2011

including

1.5M ALTOP™ CYLINDERS

15% OF ALL CYLINDERS delivered in bundles

300 filling centers

Main logistics issues

- Cylinders customers needs are at different scales (from few cyl/**year** to hundreds cyl/**day**). Common point: all customer expects an high quality of service.

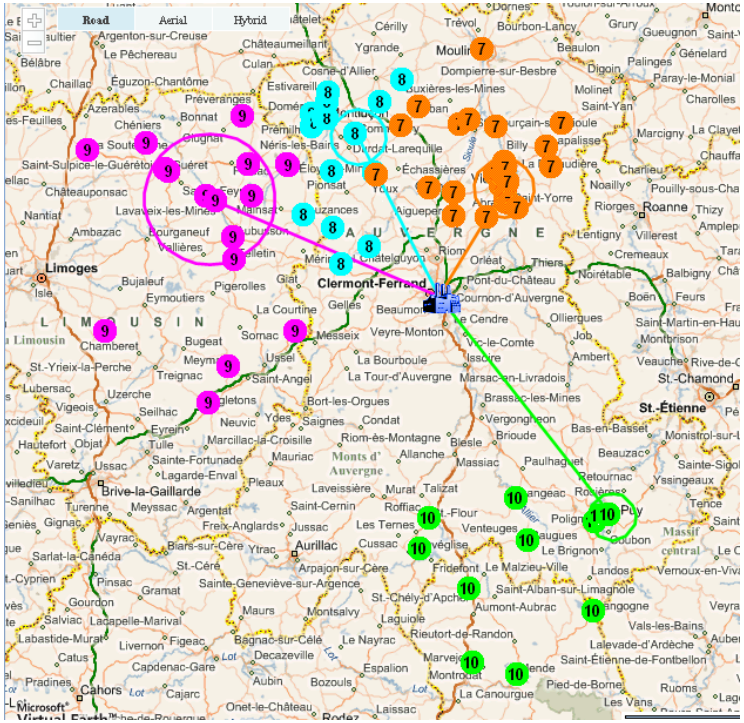
- **Broad variety of products** (purity, pressure, gas, quantity, targeted business, accessories…)

- **Cylinders transportation cost is intrinsically high** (70 kg of steel = 10 kg of gas): Production has to be near the customers and near the liquid gas production plants.

- **Production stock and Customer stock are linked** : need to have empty cylinders to fill new ones.

Tactical Optimization of cylinders distribution

- Given one filling plant and the customers (up to 1000).
- **Problem**: how to group customers optimally so that:
  - Each customer has one or more fixed day of possible delivery per week
  - The average roundtrip over a group must be compatible with the truck (capacity & working time).
- **Optimization Stackes**: 5% of over 110M€/year



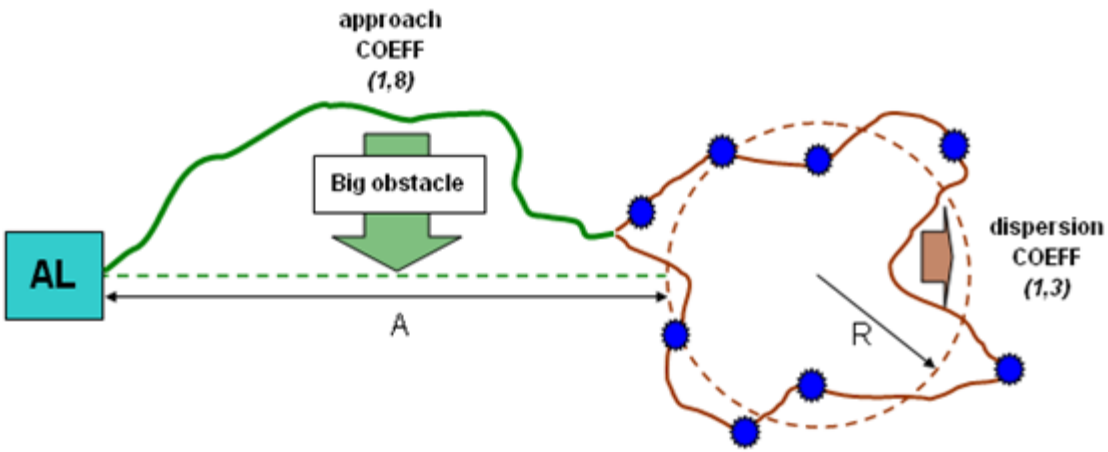
- A custom made Tabou search solver was developped in 2009 (ODRAL).
- **Objective**: Evaluate *rapidly* a possible alternative to this solver.

Model: Centroids roundtrip estimator

- The key point of this problem is to determine the cost of an “average roundtrip” of a group of customers (cluster), given:
  - The position of the customers and the filling plant
  - The cylinders demand and their frequency

- Estimator is based on the centroid concept: the centroid is the customer of the cluster that minimize this distance:

$$dist(AirLiquidePlant; c\_centroid) + \frac{\sum_{c \in cluster} consumption(c) * dist(c; c\_centroid)}{\sum_{c \in cluster} consumption(c)}$$



LocalSolver helps to « go straight » to a running model

- **First difficulty:** the cost function is non linear; some variables are discrete. Linearization is maybe possible but non-natural.
  - **In Localsolver** we can write the cost function in **a very direct way**, using min, max and if-then operators:
  - From simple decision variables we derive all non linear costs

```
function model() {
  // decision variables
  // x[i][j] = 1 if customer i is assigned to cluster j
  //(with 0 the joker cluster for unassigned customers)
  x[i in 1..nbCustomers][j in 0..nbClusters]  <- bool();
```

Conditional operators, divisions, etc.

```
// yearly frequency of a cluster: only multiple of 25
clusterFrequency[j in 1..nbClusters] <- nbvisitcluster[j]==0 ?
                                        0 :
                                        (((nbvisitcluster[j] + (25 - 1) ) / 25) * 25);
```

LocalSolver helps to « go straight » to a running model

- **Second difficulty:** The initial solution (a set of cluster that fit all constraints) is not trivial.
  - **Localsolver supports multiple objectives in a lexicographic order:** By consequence, we put the strongest constraints (all customer must be in a cluster) as a firsts order objective, using a « joker customer » to put initially all customers:

```
// First objective = assign a cluster to each customer
minimize sum[i in 1.. nbCustomers](x[i][0]);

// Second objective = minimize cluster cost
cost <- sum[j in 1..nbClusters](clusterCost[j]* clusterFrequency[j]);
minimize cost;
```

Localsolver Model: highlights

// decision variables (note: cluster 0 is a joker cluster):

```
function model() {
  // decision variables
  // x[i][j] = 1 if customer i is assigned to cluster j
  //(with 0 the joker cluster for unassigned customers)
  x[i in 1..nbCustomers][j in 0..nbClusters]  <- bool();
```

// main constraints:

```
// constraint 1: each customer is assigned to exactly one cluster
for[i in 1..nbCustomers] {
  constraint sum[j in 0..nbClusters](x[i][j]) == 1;
}

// constraint 2-3: TimeSlots cannot exceed 4 slots (2 days) -
// and have to last at least 3 timeslots if it is a two day delivery
// constraint 4: truck capacity must be respected;
for[j in 1..nbClusters] {
  constraint clusterTimeslots[j] <= 4;
  constraint clusterTimeslots[j]*clusterFrequency[j] <= 500;
  constraint clusterDemand[j] <= CAPACITY*clusterFrequency[j];
}
```

Final consideration and next steps:

- First results:
  - The model produces solutions that are comparable to the existent solver in a comparable amount of time. However, we did not yet added all the business constraints.
  - We demonstrate to my R&D department that Localsolver is advantageous to fast prototype a solver that solver a real business problem:
    - Time initially allocated to this project: **0 m/y**
    - Time to develop the model until first start: **1 week**
    - Time to import data in the model: **one day**
    - Time to add a new business constraints or objective: **minutes**
    - Time to convince my boss to invest on Localsolver: **5 months**
- Next steps:
  - Finish the implementation of all business constraints on the model.
  - Challenge current solver solutions using and try new constraint approaches.