A METHODOLOGY FOR DESIGNING REAL-LIFE APPLICATIONS

Éric Taillard

EIVD

University of Applied Sciences of Western Switzerland,
Yverdon-les-bains, Switzerland
CONTENT OF THE TALK

Problem definition
Data collection
Problem modelling

Neighbourhoods
Key element in metaheuristics; basic, reduced, expanded

Large problem instances
Problem decomposition; POPMUSIC

Multiobjective optimization
Path relinking

Dynamic problems
Adaptive memory programming

Synthesis
UNDERSTAND THE PROBLEM

Most difficult part of a practical project

The answer is not necessarily in relation with the original question

— What type of lorries do I have to buy?
— Sell the two oldest of your fleet!

Visit the operational unit

An efficient way to know what is already done

Identify the true constraints

Example: (almost) no carrying company respects the laws

Identify the right objective(s)

The customer typically want to diminish the fixed, incompressible costs
COLLECT DATA

Ask for the right data

Concise

Coherent

Typical mistake: asking for a complete distance matrix

Too many data

Data not coherent (e.g. triangle inequalities not respected)

Typically takes more than 50% of the time of the project
USE A GOOD MODEL

Turbine runner balancing:

Fig. 1. Schematic figure of a jet engine.

Source: Mason & Rönnqvist, C&OR 24, 1997

$n$ blades of weight $w_i (i = 1, \ldots, n)$

$n$ angular positions $\theta_i = i/2\pi (i = 1, \ldots, n)$ or, more generally: cartesian coordinates $(x_i, y_i)$

Objective: find a positions $p_i (i = 1, \ldots, n)$ for each blade that minimizes:

$$\left( \sum_{i=1}^{n} w_i \cdot x_{p_i} \right)^2 + \left( \sum_{i=1}^{n} w_i \cdot y_{p_i} \right)^2$$
TURBINE BALANCING

Alternate formulation (Laporte & Mercure, EJOR 35, 1988):

Quadratic assignment problem with:

- Flow matrix: $f_{ij} = w_i w_j$
- Distance matrix: $d_{ij} = \cos(\theta_i - \theta_j)$

Objective: find a permutation $p$ that minimizes:

$$\sum_{i=1}^{n} \sum_{j=1}^{n} f_{ij} \cdot d_{p_ip_j}$$

Less general

Works only for angular positions

Container vessel loading?

More complex

Objective computation $O(n^2)$ versus initial formulation $O(n)$
DIFFERENCE BETWEEN ACADEMIC AND REAL-LIFE

Academic QAP

Data : Flow matrix \((f_{ij})\) ; distance matrix \((d_{ij})\) ; \(n = 25 \ldots 150 \ldots (729)\)

Constraints : \(p \in \text{permutation}\)

Objective : Minimize \(\sum_{i=1}^{n} \sum_{j=1}^{n} f_{ij} \cdot d_{p_ip_j}\)

Real-life QAP

Data : Flow matrices : \((f_{ij}^1), (f_{ij}^2)\) ; distances matrices : \((d_{ij}^1), (d_{ij}^2)\)

\(n = 60, \ldots, 200, \ldots, (32000)\)

Constraints : Blocks \(p = \left(p_{b_1}^{n_1}, p_{b_2}^{n_2}, \ldots, p_{b_m}^{n_m}\right)\)

Computational time : 3’ at most

Objectives : Minimize \(\sum_{i=1}^{n} \sum_{j=1}^{n} f_{ij}^{1} \cdot d_{p_ip_j}^{1}\) and \(\sum_{i=1}^{n} \sum_{j=1}^{n} f_{ij}^{2} \cdot d_{p_ip_j}^{2}\)
Find the right balance:

Too many hard constraints
- Difficult to find a feasible solution
- Difficult to move from one solution to another

Too many soft constraints
- Solutions not feasible in real life
- Finding good penalty associated with a violated constraint

Balance depends on solving method:

Constraint programming
- As many hard constraints as possible (reduce solutions space)

Noising methods
- Many soft constraints (for adding noise in the objective)
**NEIGHBOURHOOD**

**Definition of good neighbourhoods is a key element in many practical applications**

A local optimum (relatively to an appropriate neighbourhood) is often convenient in real-life applications (and there is often no time to do much more)

Interesting academic property: this instance is hard to solve exactly

Real-life interest: ?
SIMPLE NEIGHBOURHOODS

Examples for the VRP ($n$ customers, $m$ tours)

**Insertion** (1-interchange): $O(nm)$ neighbours

**Exchange** (2-interchange): $O(n^2)$ neighbours

**Generalization**: $\lambda$-interchange: $O(n^\lambda)$ neighbours.
RESTRICTION OF NEIGHBOURHOOD SIZE

Candidate list

Example 1: CROSS-neighbourhood: $O(n^4/m^2)$ neighbours

Example 2: Granular tabu search (Toth & Vigo 1998)

Consider only the shorter edges adjacent to each customers

$O(n^2) \rightarrow O(n)$
Neighbourhood Expansion: Composite Moves

Ejection Chains

- Avoid atomic changes
- Expand neighbourhood size without increasing complexity too much
- Perform jumps in solution space

Example for the VRP:


Find the max flow from $s$ to $t$ with lowest cost

$c_{ij}$: cost of removing customer $i$ and inserting it in tour $j$
**LARGE NEIGHBOURHOOD : “ROTATION”**

Finding the best rotation with ejection chains: $O(nm^2)$
DECOMPOSITION OF LARGE PROBLEMS

Large neighbourhood, POPMUSIC

Solution composed of parts $s_1, \ldots, s_p$

Take a seed part $s_i$

Build a sub-problem: $s_i + r$ “closest” parts

Optimize sub-problem

Recompose solution

Repeat with another seed part

Taillard 1993, Shaw 1998

VRP

Taillard & Voss 2001

POPMUSIC

Taillard, Taillard & Wälti 2003

Location-allocation
FROM SINGLE TO MULTI-OBJECTIVE OPTIMIZATION

Path relinking

X. Gandibleux, H. Morita, N. Katoh, MIC 2003

Objective 2

Solution 1

Path from 1 to 2

Local searches

Solution 2

Objective 1
**Dynamic Problems**

Often, practical problems are dynamic (e.g. arrival of new customers, job finished, breakdown)

**Adaptive Memory Programming (AMP)**

**General algorithm**

- Initialize memory
- Repeat, until a stopping criterion met:
  - Build a *provisory solution* with the help of informations in memory
  - Improve *provisory solution* with a local search
  - Update *memory* with informations obtained with new solution

- **Ant systems**
  - Memory $\equiv$ *pheromone traces*

- **Evolutionary algorithms, Scatter Search**
  - Memory $\equiv$ *population of solutions*

- **Vocabulary building**
  - Memory $\equiv$ *pieces of solutions*

**Provocative comment**: Memories and building procedures for all these methods are equivalent.
Memory management

Initialization

Adaptive memory

Exploitation

Optimization process

Dynamic Data

Low level control

[s]

[ms]

Computing power not used by real time

Real time management

AMP TECHNOLOGY
CONCLUSION: METHODOLOGY PROPOSAL

Design of a complex method

Adaptive Memory Programming

POPMUSIC

Local search, Taboo search, SA, VNS

Ejection chains, Candidate list

Simple Neighbourhood

POPMUSIC

Exact Optimization

A less complex design