Reacting and Adapting to the Environment
Designing Autonomous Methods
for Multi-Objective Combinatorial Optimisation

Aymeric Blot
Supervisor: Laetitia Jourdan
Co-advisor: Marie-Éléonore Kessaci
ORKAD team, CRISiAL, Université de Lille

PhD defence – September 21, 2018

Contents

▶ Introduction
▶ Context
▶ Multi-Objective Local Search
▶ Automatic Design
▶ Wrap-up

Thesis

Reacting and Adapting to the Environment
Designing Autonomous Methods
for Multi-Objective Combinatorial Optimisation

Topic Automatic algorithm design
Context Multi-objective combinatorial optimisation
Use Case Multi-objective local search algorithms

Travelling Salesman Problem

Input Set of \( n \) cities, travel costs
Solutions Hamiltonian paths (permutations)
Quality Total cost (e.g., distance, time, money)
**Thesis**

**Reacting and Adapting to the Environment**
Designing Autonomous Methods for Multi-Objective Combinatorial Optimisation

**Environment**

<table>
<thead>
<tr>
<th>Problem</th>
<th>Circuit board drilling? Order-picking? Vehicle routing?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Search</td>
<td>Easy to improve? Stuck in local optima?</td>
</tr>
</tbody>
</table>

**Permutation Flowshop Scheduling Problem**

**Input** Set of $n$ jobs, processing times on $m$ machines

**Solutions** Jobs schedules (permutations)

**Quality** Various, e.g.:
- Makespan (max of completion times)
- Flowtime (sum of completion times)

\[M_1\]
\[M_2\]
\[\ldots\]
\[M_m\]

**Ambitions**

Automatically, in a multi-objective context:
- Design algorithms variants for specific problem characteristics
- Benefit from many existing strategies
- Avoid relying on expert knowledge

**Roadmap**

**Reacting and Adapting to the Environment**
Designing Autonomous Methods for Multi-Objective Combinatorial Optimisation

**Topic** Automatic algorithm design

**Context** Multi-objective combinatorial optimisation

**Use Case** Multi-objective local search algorithms
**Automatic Algorithm Design**

**Algorithm Performance**
- Differs with the problem
- Differs with the instance
- Depends on explicit or hidden design choices

**Ideas**
- Select from a set of existing algorithms
- Tune a specific algorithm
- Generate new algorithms

**AAD: Taxonomy Proposition**

**Algorithmic viewpoint**
- Parameters
- Components
- Algorithms

**Temporal viewpoint**
- Problem features
  - Tuning
  - Configuration
  - Mapping
- A priori features
  - Setting
  - Selection
- Search features
  - Control
  - Scheduling

**AAD: Investigated Fields**

**Roadmap**

**Reacting and Adapting to the Environment**
Designing Autonomous Methods for Multi-Objective Combinatorial Optimisation

**Topic** Automatic algorithm design

**Context** Multi-objective combinatorial optimisation

**Use Case** Multi-objective local search algorithms
Multi-Objective Optimisation

Bi-objective minimisation

- Dominated solutions
- (Optimal) archive Pareto (optimal) set

Performance Assessment

Hypervolume (1-HV)

- Spread

Roadmap

Reacting and Adapting to the Environment
Designing Autonomous Methods for Multi-Objective Combinatorial Optimisation

Topic: Automatic algorithm design
Context: Multi-objective combinatorial optimisation
Use Case: Multi-objective local search algorithms

Questions:
- General structure?
- Possible strategies?
- Efficiency?

Local Search Algorithms

“Similar solutions have similar quality”

Trajectory
- Identify neighbours
- Move the current solution
- Iterate
Multi-Objective Local Search Algorithms

Selected History

- Single trajectory
  - MOSA [Serafini, 1994]
  - TPLS [Paquete et al., 2003]
- Multiple trajectories
  - PSA [Czyzak et al., 1996]
  - MOTS [Hansen, 1997]
- Archive
  - PAES [Knowles et al., 1999]
  - PLS [Paquete et al., 2004]

MOLS Generalisation

Components

- Initialisation
- Selection
- Exploration
- Archive
- Stopping condition
- Perturbation

<animation>
Selected MOLS Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Parameter values</th>
</tr>
</thead>
<tbody>
<tr>
<td>initStrat</td>
<td>category</td>
<td>{...}</td>
</tr>
<tr>
<td>selectStrat</td>
<td>category</td>
<td>{all, rand, newest, oldest}</td>
</tr>
<tr>
<td>selectSize</td>
<td>integer</td>
<td>N*</td>
</tr>
<tr>
<td>explorStrat</td>
<td>category</td>
<td>{all, imp, ndom, ...}</td>
</tr>
<tr>
<td>explorRef</td>
<td>category</td>
<td>{pick, arch}</td>
</tr>
<tr>
<td>explorSize</td>
<td>integer</td>
<td>N*</td>
</tr>
<tr>
<td>archiveStrat</td>
<td>category</td>
<td>{bounded, unbounded, ...}</td>
</tr>
<tr>
<td>archiveSize</td>
<td>integer</td>
<td>N*</td>
</tr>
<tr>
<td>iterationLength</td>
<td>integer</td>
<td>N*</td>
</tr>
<tr>
<td>perturbStrat</td>
<td>category</td>
<td>{restart, kick, ...}</td>
</tr>
<tr>
<td>perturbSize</td>
<td>integer</td>
<td>N*</td>
</tr>
<tr>
<td>perturbStrength</td>
<td>integer</td>
<td>N*</td>
</tr>
</tbody>
</table>

Parameter Distribution Analysis

How efficient are the generated MOLS?

- Protocol
  - 300 MOLS configurations
  - 3 PFSP + 3 TSP scenarios
  - 10 runs per instance
  - Average $(1 - HV, \Delta')$

- Scenarios
  - PFSP (10 instances)
    - 50 jobs, 20 machines
    - 100 jobs, 20 machines
    - 200 jobs, 20 machines
  - TSP (15 instances)
    - 100 cities
    - 300 cities
    - 500 cities

Results: Parameter Distribution Analysis

Analysis

Conclusions
- Generated MOLS can be very efficient
- Parameters values are meaningful

Next Step
- Automatically design efficient MOLS algorithms

The configuration space is structured!
Knowledge can be extracted!
Expert knowledge is limited
Roadmap

Reacting and Adapting to the Environment
Designing Autonomous Methods for Multi-Objective Combinatorial Optimisation

Topic Automatic algorithm design
Context Multi-objective combinatorial optimisation
Use Case Multi-objective local search algorithms

Questions:
- How to automatically design efficient MOLS?
- Is it possible to beat expert knowledge?
- How to improve adaptability?

Algorithm Configurators

Automatic Algorithm Configuration

Goal Optimise performance over a given distribution of instances
Mean Optimisation, machine learning
Twist Data is unreliable and very expensive

Single-Objective Configuration
- irace [López-Ibáñez et al., 2016]
- ParamILS [Hutter et al., 2009]
- SMAC [Hutter et al., 2010]
- GGA++ [Ansótegui et al., 2015]

Multi-Objective Configuration
- SPRINT-Race [Zhang et al., 2015]
- MO-ParamILS [Blot et al., 2016]

MO-ParamILS

- Extension of ParamILS for multiple performance indicators
- Iterated MOLS on the configuration space
- Outputs a Pareto set of configurations

Configuration Protocol

How to ensure efficient predictions?

3 Phases
- Training
  - On training instances
  - Multiple times (e.g., ×20)
- Validation
  - All final configurations
- Test
  - Non-dominated configurations
  - On test instances

\[ \Delta' = \text{animation} \]
\[ 1 - HV \]
Automatic Configuration

How efficient is our multi-objective approach?

Configurators
- ParamILS
  - Single-objective
  - \((1 - HV)\)
- ParamILS
  - Single-objective
  - \(\frac{3}{4} (1 - HV) + \frac{1}{4} \Delta'\)
- MO-ParamILS
  - Multi-objective
  - \((1 - HV), \Delta'\) simultaneously

Protocol
- Few configurations
  - 10×100 runs / 300 MOLS
  - 3 PFSP + 3 TSP scenarios
- More configurations
  - 20×1000 runs / 10920 MOLS
  - 3 PFSP + 3 TSP scenarios
- Crafted instances
  - 20×1000 runs / 10920 MOLS
  - 3 PFSP + 3 TSP scenarios

Results: Automatic Configuration

“Exhaustive” analysis: x (300 configurations)
Configurator: ○ ParamILS △ ParamILS(0.75,0.25) □ MO-ParamILS

MO-ParamILS: excellent spread, no loss of convergence

Analysis

Conclusions
- MO-ParamILS allows much better context
- Configuration of MO algorithms is a MO problem
- Problem: predicts single configurations

Next Steps
- Scheduling
  - Sequence multiple strategies
- Control
  - Interweave multiple predictions
  - Delay predictions

How to better fit the algorithm to the search?

Configuration Scheduling

\[
\begin{array}{c}
\text{Performance may vary during the search} \\
\text{Real-time decisions are difficult} \\
\text{Static schedules can be optimised offline}
\end{array}
\]
Experiments

How efficient are configuration schedules?

Protocol

<table>
<thead>
<tr>
<th>$K = 1$ ($k = 1$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exhaustive analysis; single configurations</td>
</tr>
<tr>
<td>60 configurations = 60 schedules</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$K = 2$ ($k \in {1, 2}$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Automatic configuration; up to two configurations</td>
</tr>
<tr>
<td>20×1000 runs / 10860 schedules</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$K = 3$ ($k \in {1, 2, 3}$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Automatic configuration; up to three configurations</td>
</tr>
<tr>
<td>20×10000 runs / 658860 schedules</td>
</tr>
</tbody>
</table>

Selected $K = 3$ Configuration Schedules

\[
\begin{align*}
(T/3, T/3, T/3) & \quad \text{timed} \\
(T/4, T/4, T/2) & \quad \text{timed} \\
(T/2, T/4, T/4) & \quad \text{timed} \\
(T/2, T/2) & \quad \text{timed} \\
(T/4, 3T/4) & \quad \text{timed} \\
(3T/4, T/4) & \quad \text{timed} \\
(T) & \quad \text{timed} \\
\end{align*}
\]

\[3 \times 60^3 + 3 \times 60^2 + 60 = 658,860 \text{ schedules}\]

Results: Configuration Scheduling

<table>
<thead>
<tr>
<th>$K = k = 1$ exhaustive analysis</th>
</tr>
</thead>
<tbody>
<tr>
<td>PFSP 50 jobs, 20 machines</td>
</tr>
</tbody>
</table>

\[\Delta' \quad \text{Pareto dominated} \]

\[\Delta' \quad \text{dominated} \]

Better balanced algorithms!

K = k = 1 Pareto dominated

Analysis

Conclusions

| $k = 1$ schedules are limited |
| Schedules can be optimised offline |
| Combinatorial explosion |

Offline Adaptation

| Schedules are still predicted |
| No real-time decisions |
Control

<table>
<thead>
<tr>
<th>Offline Design</th>
<th>Online Design</th>
</tr>
</thead>
<tbody>
<tr>
<td>▶ Prediction based</td>
<td>▶ Adaptation based</td>
</tr>
<tr>
<td>▶ Instance classes / distributions</td>
<td>▶ Single current instance</td>
</tr>
<tr>
<td>▶ Computationally expensive</td>
<td>▶ Slight overhead</td>
</tr>
</tbody>
</table>

Motivations

▶ Use control as an extension of offline learning
▶ Take advantage of multiple strategies during the run
▶ Delay the final prediction

Control Mechanisms

Generic Parameter Control

▶ Random
▶ Probability based
▶ Multi-armed bandits
▶ Reinforcement learning

[Karafotias et al., 2015]

Experiments

Can efficient strategies be determined online?

Protocol

▶ 2 simple control mechanisms
▶ 12 PFSP scenarios
▶ 200 runs per scenario

Strategies

▶ 3 arms (imp, imp-ndom, ndom)
▶ 2 arms (imp-ndom, ndom)
▶ 3 → 2 arms

Simple Control Mechanisms

▶ Uniform random: \( p_i(t + 1) = \frac{1}{N} \)
▶ \( \varepsilon \)-greedy: \( p_i(t + 1) = \begin{cases} (1 - \varepsilon) + \frac{\varepsilon}{N}, & \text{if } i = \arg \max_j q_j(t) \\ \varepsilon / N, & \text{otherwise} \end{cases} \)
### Results: 3-arm Ranking

Wilcoxon signed ranked tests, Friedman post-hoc analysis

<table>
<thead>
<tr>
<th>Approach</th>
<th>Instance ($n, m$)</th>
<th>Avg.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>20</td>
<td>50</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>10</td>
</tr>
<tr>
<td>imp</td>
<td>5 5 5 5 5 5 5 5 5 5 5 5</td>
<td>5</td>
</tr>
<tr>
<td>imp-ndom</td>
<td>4 4 3 4 4 4 4 1 2 1 2 1</td>
<td>2.8</td>
</tr>
<tr>
<td>ndom</td>
<td>1 1 3 1 1 1 1 1 1 1 1 1</td>
<td>1.2</td>
</tr>
<tr>
<td>rand_3</td>
<td>1 1 1 1 1 1 1 1 2 3 3 3</td>
<td>1.6</td>
</tr>
<tr>
<td>greedy_3</td>
<td>1 1 1 1 1 1 1 1 2 3 3 3</td>
<td>1.6</td>
</tr>
</tbody>
</table>

Control fails on larger instances

### Results: 2-arm Ranking

Wilcoxon signed ranked tests, Friedman post-hoc analysis

<table>
<thead>
<tr>
<th>Approach</th>
<th>Instance ($n, m$)</th>
<th>Avg.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>20</td>
<td>50</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>10</td>
</tr>
<tr>
<td>imp-ndom</td>
<td>4 4 3 4 4 4 4 4 4 4 4 1</td>
<td>3.7</td>
</tr>
<tr>
<td>ndom</td>
<td>1 1 3 1 1 1 1 1 1 1 1 1</td>
<td>1.2</td>
</tr>
<tr>
<td>rand_2</td>
<td>1 1 1 1 1 1 1 1 1 1 1 1</td>
<td>1.1</td>
</tr>
<tr>
<td>greedy_2</td>
<td>1 1 1 1 1 1 1 1 1 1 1 1</td>
<td>1.1</td>
</tr>
</tbody>
</table>

imp was the culprit

### Results: Long Term Learning Ranking

Wilcoxon signed ranked tests, Friedman post-hoc analysis

<table>
<thead>
<tr>
<th>Approach</th>
<th>Instance ($n, m$)</th>
<th>Avg.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>20</td>
<td>50</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>10</td>
</tr>
<tr>
<td>rand_3</td>
<td>4 4 2 4 4 4 4 4 4 4 4 3</td>
<td>3.8</td>
</tr>
<tr>
<td>rand_ltl_50</td>
<td>3 1 2 1 1 1 3 3 3 2 3 3</td>
<td>2.2</td>
</tr>
<tr>
<td>rand_ltl_20</td>
<td>1 1 2 1 1 1 1 1 1 2 2 2</td>
<td>1.3</td>
</tr>
<tr>
<td>rand_2</td>
<td>1 1 1 1 1 1 1 1 1 1 1 1</td>
<td>1</td>
</tr>
<tr>
<td>greedy_3</td>
<td>1 1 1 1 4 4 4 4 4 4 4 3</td>
<td>2.9</td>
</tr>
<tr>
<td>greedy_ltl_50</td>
<td>1 1 1 1 1 1 3 3 3 3 2 3</td>
<td>1.9</td>
</tr>
<tr>
<td>greedy_ltl_20</td>
<td>1 1 1 1 3 1 1 1 1 2 2 2</td>
<td>1.3</td>
</tr>
<tr>
<td>greedy_2</td>
<td>1 1 1 1 1 1 1 1 1 1 1 1</td>
<td>1</td>
</tr>
</tbody>
</table>

Ineffective arms should be automatically removed

### General Contributions and Conclusions

**Automatic Algorithm Design**
- Taxonomy proposition
- Multi-objective configuration, MO-ParamILS
  - MO algorithms are better optimised using a MO configurator
- Configuration scheduling
  - Better balanced algorithms can be predicted
- Control as extension of automatic configuration
  - Some design choices can be postponed to the search itself

**Multi-objective Optimisation**
- Wider generalisation of MOLS algorithms
- Automatic design of multi-objective algorithms
Short-Term Perspectives

Automatic design
▶ Extension to other algorithms
▶ Other multi-objective configurators
▶ Robustness in configurators

Automatic configuration
▶ Validation on other types of problems

Configuration scheduling
▶ Guided experimentation protocol
▶ More semantic representation

Online mechanisms
▶ More strategies, more complex mechanisms

Long-Term Perspectives

Anytime Behaviour of Algorithms

Insight Other applications of multi-objective algorithm design
Example Quality/running time trade-off
Ideas ▶ Designing for multiple running times
▶ Area-under-the-curve as fitness
▶ Configuration scheduling

Artificial Configuration Spaces

Insight Automatic configuration extremely time-expensive
Problem So is developing/improving/comparing configurators
Ideas ▶ Semantic parameter analysis
▶ Zero-cost configuration spaces

Publications I

Blot, Hoos, Jourdan, Kessaci-Marmion, and Trautmann – LION 2016
MO-ParamILS: A Multi-objective Automatic Algorithm Configuration Framework

Blot, Pernet, Jourdan, Kessaci-Marmion, and Hoos – EMO 2017
Automatically Configuring Multi-objective Local Search Using Multi-objective Optimisation

Blot, Kessaci-Marmion, and Jourdan – MIC 2017
AMH: a new Framework to Design Adaptive Metaheuristics

Blot, Kessaci-Marmion, and Jourdan – GECCO 2017
Automatic design of multi-objective local search algorithms: case study on a bi-objective permutation flowshop scheduling problem

Publications II

Blot, Kessaci, Jourdan, and de Causmaecker – LION 2018
Adaptive Multi-Objective Local Search Algorithms for the Permutation Flowshop Scheduling Problem

Blot, López-Ibáñez, Kessaci, and Jourdan – PPSN 2018
Archive-aware Scalarisation-based Multi-Objective Local Search for a Bi-objective Permutation Flowshop Problem

Blot, Hoos, Kessaci, and Jourdan – ICTAI 2018
Automatic Configuration of Multi-objective Optimization Algorithms. Impact of Correlation between Objectives

Blot, Kessaci, and Jourdan – Journal of Heuristics, 2018
Survey and Unification of Local Search Techniques in Metaheuristics for Multi-objective Combinatorial Optimisation