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, CRIStAL, Université de Lille

PhD defence – September 21, 2018
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
Contents

▶ Introduction

▶ Context

▶ Multi-Objective Local Search

▶ Automatic Design

▶ Wrap-up
Travelling Salesman Problem

**Input**  Set of $n$ cities, travel costs

**Solutions**  Hamiltonian paths (permutations)

**Quality**  Total cost (e.g., distance, time, money)
Reacting and Adapting to the Environment
Designing Autonomous Methods
for Multi-Objective Combinatorial Optimisation

Environment

**Problem** Circuit board drilling? Order-picking? Vehicle routing?


**Search** Easy to improve? Stuck in local optima?
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)
Reacting and Adapting to the Environment
Designing Autonomous Methods
for Multi-Objective Combinatorial Optimisation

Ambitions

Automatically, in a multi-objective context:

- Design algorithms variants for specific problem characteristics
- Benefit from many existing strategies
- Avoid relying on expert knowledge
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

Temporal viewpoint:
- problem features
- a priori features
- search features

Algorithmic viewpoint:
- parameters
- components
- algorithms
AAD: Taxonomy Proposition

Algorithmic viewpoint

parameters components algorithms

problem features

Tuning Configuration Mapping

a priori features

Setting Selection

search features

Control Scheduling
AAD: Investigated Fields

Temporal viewpoint:
- Problem features
- A priori features
- Search features

Algorithmic viewpoint:
- Parameters
- Components
- Algorithms

- Tuning
- Configuration
- Mapping
- Setting
- Selection
- Control
- Scheduling
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

Solutions
Multi-Objective Optimisation

Bi-objective minimisation

\[ f_2 \]

\[ f_1 \]

Dominated solutions

(Optimal) archive Pareto (optimal) set
Performance Assessment

Hypervolume (1-HV)

Δ’ Spread
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

![Graph showing components of MOLS generalisation](image-url)
MOLS Generalisation

Components

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

![Graph showing components of MOLS Generalisation](image-url)
MOLS Generalisation

Components

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

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

Components

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

Components

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

Components

- Initialisation
- Selection
- Exploration
- Archive
- Stopping conditions
- Perturbation
## 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>iterationStagnation</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>
## 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>iterationStagnation</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: “Exhaustive” Analysis

The configuration space is structured!
Results: Parameter Distribution Analysis

Exploration strategy: × imp  × imp_ndom  × ndom

Knowledge can be extracted!
Results: Parameter Distribution Analysis

PFSP 200 jobs, 20 machines

TSP 500 cities

Exploration strategy: \( \times \) imp \( \times \) imp\_ndom \( \times \) ndom
Selection strategy: \( \circ \) all \( \triangle \) oldest \( \square \) rand

Expert knowledge is limited
Analysis

Conclusions

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

Next Step

- Automatically design efficient MOLS algorithms
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]
# Algorithm Configurators

## Automatic Algorithm Configuration

<table>
<thead>
<tr>
<th><strong>Goal</strong></th>
<th>Optimise performance over a given distribution of instances</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Mean</strong></td>
<td>Optimisation, machine learning</td>
</tr>
<tr>
<td><strong>Twist</strong></td>
<td>Data is unreliable and very expensive</td>
</tr>
</tbody>
</table>

## 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 space
instance set

Configurator

performance

instance, configuration

Target algorithm

Return best configurations

Blot, Hoos, Jourdan, Kessaci-Marmion, and Trautmann – LION 2016
Configuration Protocol

How to ensure efficient predictions?

3 Phases

- **Training**
  - On training instances
  - Multiple times (e.g., \( \times 20 \))

- **Validation**
  - All final configurations
  - On training instances

- **Test**
  - Non-dominated configurations
  - On test instances
Configuration Protocol

How to ensure efficient predictions?

3 Phases

- **Training**
  - On training instances
  - Multiple times (e.g., $\times 20$)

- **Validation**
  - All final configurations
  - On training instances

- **Test**
  - Non-dominated configurations
  - On test instances

\[ \Delta' \]

\[ 1 - HV \]
Configuration Protocol

How to ensure efficient predictions?

3 Phases

- Training
  - On training instances
  - Multiple times (e.g., ×20)

- Validation
  - All final configurations
  - On training instances

- Test
  - Non-dominated configurations
  - On test instances
### Automatic Configuration

#### How efficient is our multi-objective approach?

<table>
<thead>
<tr>
<th>Configurators</th>
<th>Protocol</th>
</tr>
</thead>
<tbody>
<tr>
<td>▶ ParamILS</td>
<td>▶ Few configurations</td>
</tr>
<tr>
<td>▶ Single-objective</td>
<td>▶ 10×100 runs / 300 MOLS</td>
</tr>
<tr>
<td>▶ (1 − HV)</td>
<td>▶ 3 PFSP + 3 TSP scenarios</td>
</tr>
<tr>
<td>▶ ParamILS</td>
<td>▶ More configurations</td>
</tr>
<tr>
<td>▶ Single-objective</td>
<td>▶ 20×1 000 runs / 10 920 MOLS</td>
</tr>
<tr>
<td>▶ $\frac{3}{4}(1 − HV) + \frac{1}{4} \Delta'$</td>
<td>▶ 3 PFSP + 3 TSP scenarios</td>
</tr>
<tr>
<td>▶ MO-ParamILS</td>
<td>▶ Crafted instances</td>
</tr>
<tr>
<td>▶ Multi-objective</td>
<td>▶ 20×1 000 runs / 10 920 MOLS</td>
</tr>
<tr>
<td>▶ (1 − HV), $\Delta'$ simultaneously</td>
<td>▶ 3 PFSP + 3 TSP scenarios</td>
</tr>
</tbody>
</table>
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
Configuration Scheduling

How to better fit the algorithm to the search?

Configuration Schedules

- Performance may vary during the search
- Real-time decisions are difficult
- Static schedules can be optimised offline
Experiments

How efficient are configuration schedules?

## Protocol

- **$K = 1$** ($k = 1$)
  - Exhaustive analysis; single configurations
  - 60 configurations = 60 schedules
- **$K = 2$** ($k \in \{1, 2\}$)
  - Automatic configuration; up to two configurations
  - $20 \times 1000$ runs / 10860 schedules
- **$K = 3$** ($k \in \{1, 2, 3\}$)
  - Automatic configuration; up to three configurations
  - $20 \times 10000$ runs / 658860 schedules
Selected $K = 1$ Configuration Schedules

$\{ k = 1 \} (T) \quad 60$ schedules
Selected $K = 2$ Configuration Schedules

- $(T/2, T/2)$
- $(T/4, 3T/4)$
- $(3T/4, T/4)$

$3 \times 60^2 + 60 = 10,860$ schedules
Selected $K = 3$ Configuration Schedules

\[ (T/3, T/3, T/3) \]
\[ (T/4, T/4, T/2) \]
\[ (T/2, T/4, T/4) \]
\[ (T/2, T/2) \]
\[ (T/4, 3T/4) \]
\[ (3T/4, T/4) \]
\[ (T) \]

\( t = 0 \) \hspace{1cm} \( T \) \hspace{1cm} \text{time}

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

$K = k = 1$ exhaustive analysis
PFSP 50 jobs, 20 machines

$\Delta'$

Better balanced algorithms!

Automatic design
PFSP 50 jobs, 20 machines

$\Delta'$
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

## Offline Design
- Prediction based
- Instance classes / distributions
- Computationally expensive

## Online Design
- Adaptation based
- Single current instance
  - *Slight* overhead

## 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]
Adaptive MOLS Algorithm

Iterated MOLS

MOLS_A → MOLS_B → MOLS_C

Combine

Perturb.

?
## 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 $\rightarrow$ 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></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>20</th>
<th>50</th>
<th>100</th>
<th>200</th>
<th>500</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>imp</strong></td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td><strong>imp-ndom</strong></td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td><strong>ndom</strong></td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td><strong>rand_3</strong></td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td><strong>greedy_3</strong></td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

Control fails on larger instances
## 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  10  20</td>
<td>5  10 20</td>
</tr>
<tr>
<td><strong>imp</strong></td>
<td>5  5  5</td>
<td>5  5  5</td>
</tr>
<tr>
<td><strong>imp-ndom</strong></td>
<td>4  4  3</td>
<td>4  4  4</td>
</tr>
<tr>
<td><strong>ndom</strong></td>
<td>1  1  3</td>
<td>1  1  1</td>
</tr>
<tr>
<td><strong>rand_3</strong></td>
<td>1  1  1</td>
<td>1  1  1</td>
</tr>
<tr>
<td><strong>greedy_3</strong></td>
<td>1  1  1</td>
<td>1  1  1</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 50 100 200 500</td>
<td></td>
</tr>
<tr>
<td></td>
<td>5 10 20</td>
<td>5 10 20</td>
</tr>
<tr>
<td><strong>imp-ndom</strong></td>
<td>4 4 3</td>
<td>4 4 4</td>
</tr>
<tr>
<td><strong>ndom</strong></td>
<td>1 1 3</td>
<td>1 1 1</td>
</tr>
<tr>
<td><strong>rand_2</strong></td>
<td>1 1 1</td>
<td>1 1 1</td>
</tr>
<tr>
<td><strong>greedy_2</strong></td>
<td>1 1 1</td>
<td>1 1 1</td>
</tr>
</tbody>
</table>

*imp was the culprit*
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 10 20</td>
<td>5 10 20</td>
</tr>
<tr>
<td>imp-ndom</td>
<td>4 4 3</td>
<td>4 4 4</td>
</tr>
<tr>
<td>ndom</td>
<td>1 1 3</td>
<td>1 1 1</td>
</tr>
<tr>
<td>rand_2</td>
<td>1 1 1</td>
<td>1 1 1</td>
</tr>
<tr>
<td>greedy_2</td>
<td>1 1 1</td>
<td>1 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 (5\ 10\ 20)</td>
<td>50 (5\ 10\ 20)</td>
</tr>
<tr>
<td>rand_3</td>
<td>4  4  2</td>
<td>4  4  4</td>
</tr>
<tr>
<td>rand_ltl_50</td>
<td>3  1  2</td>
<td>1  1  1</td>
</tr>
<tr>
<td>rand_ltl_20</td>
<td>1  1  2</td>
<td>1  1  1</td>
</tr>
<tr>
<td>rand_2</td>
<td>1  1  1</td>
<td>1  1  1</td>
</tr>
<tr>
<td>greedy_3</td>
<td>1  1  1</td>
<td>1  4  4</td>
</tr>
<tr>
<td>greedy_ltl_50</td>
<td>1  1  1</td>
<td>1  1  1</td>
</tr>
<tr>
<td>greedy_ltl_20</td>
<td>1  1  1</td>
<td>1  3  1</td>
</tr>
<tr>
<td>greedy_2</td>
<td>1  1  1</td>
<td>1  1  1</td>
</tr>
</tbody>
</table>

Ineffective arms should be automatically removed
## 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 10 20</td>
<td>5 10 20</td>
</tr>
<tr>
<td>rand_3</td>
<td>4 4 2 4 4 4</td>
<td>4 4 4 4 4 3</td>
</tr>
<tr>
<td>rand_ltl_50</td>
<td>3 1 2 1 1 1</td>
<td>3 3 3 2 3 3</td>
</tr>
<tr>
<td>rand_ltl_20</td>
<td>1 1 2 1 1 1</td>
<td>1 1 1 2 2 2</td>
</tr>
<tr>
<td>rand_2</td>
<td>1 1 1 1 1 1</td>
<td>1 1 1 1 1 1</td>
</tr>
<tr>
<td>greedy_3</td>
<td>1 1 1 1 4 4</td>
<td>4 4 4 4 4 3</td>
</tr>
<tr>
<td>greedy_ltl_50</td>
<td>1 1 1 1 1 1</td>
<td>3 3 3 3 2 3</td>
</tr>
<tr>
<td>greedy_ltl_20</td>
<td>1 1 1 1 3 1</td>
<td>1 1 1 1 2 2</td>
</tr>
<tr>
<td>greedy_2</td>
<td>1 1 1 1 1 1</td>
<td>1 1 1 1 1 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
This frame intentionally left blank.
Publications 1

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