AMH: a Framework to Design Adaptive MetaHeuristics

Aymeric Blot, Marie-Éléonore Kessaci-Marmion, Laetitia Jourdan

Université de Lille, Inria, UMR 9189 – CRIStAL

MIC – 06 July, 2017
Context

**Metaheuristics**
- Approximation algorithms for optimisation problems
- Few assumptions about the problem (genericity)

**Performance**
- Differs with the problem
- Differs with the instance
- Depends on its parameters

**Adaptation**
- Fine-tuning the parameters to the instance
Example: Genetic Algorithms

- **Initialisation**
- **Stop?**
  - yes
  - **Selection**
  - no
  - **Crossover**
  - **Mutation**

**Configuration**

- **Initialisation?**
- **Selection**
  - Ranking? Tournament?
- **Crossover**
  - Single-point? Two-point? Uniform?
  - Crossover rate?
- **Mutation**
  - Mutation operator?
  - Mutation rate?
- **Other**
  - Population size?
  - ...

Blot et al. AMH: a Framework to Design Adaptive MetaHeuristics
Algorithm Adaptation
What is the best configuration?

Offline Configuration

- Pre-solving external mechanism
- Find the most promising algorithm configuration
- Automatic tuning tools (e.g., irace [López-Ibáñez et al., 2016], MO-ParamILS [Blot et al., 2016], GGA++ [Ansótegui et al., 2016])

⇒ requires versatility

Online Control

- In-solving internal mechanism
- Adapt the current parameters and strategies during the search
- Generally algorithm-specific

⇒ requires dynamic implementation
# Algorithm Design Frameworks

## Some Available Frameworks

- ParadisEO\(^1\) (C++)
- jMetal\(^2\) (java)

## Adaptation?

- Possible offline configuration (not straightforward)
- Difficult online control

---

\(^1\) [http://paradiseo.gforge.inria.fr/](http://paradiseo.gforge.inria.fr/)
\(^2\) [https://jmetal.github.io/jMetal/](https://jmetal.github.io/jMetal/)
Motivation

AMH: A Single Framework

- To facilitate offline configuration
  - Single implementation $\rightarrow$ multiple algorithms
- To enable online adaptation
  - Static algorithm $\rightarrow$ dynamic / adaptive algorithm
  - Generic control mechanisms
AMH Principles

Philosophy

- Algorithm ⇔ Execution flow
- Everything is a function

Ideas

- Handle its own algorithm execution flow
- Build the algorithm at runtime
- Keep the execution flow dynamic
Offline Design in AMH

Design Process

1. Encapsulation
2. Composition
3. Integration
Online Design in AMH

Yes

Stop?

No

Selection

Crossover 1

Mutation

Yes

Stop?

No

Selection

Crossover 2

Mutation

crossover 1

crossover 2

? timeline

Blot et al. AMH: a Framework to Design Adaptive MetaHeuristics
Generic Control Mechanisms

Design Process

1. Update the component
2. Compute feedback

Proposed Control Mechanisms

- Random
- Probability matching
- Adaptive Pursuit
- Multi-armed bandit
AMH: Adaptive MetaHeuristics

- Stand-alone C++ framework
- Can be used with other C++ frameworks (e.g., ParadisEO)
- Handle the algorithm execution flow
- Easy to build an algorithm from basic blocks
- Easy to modify it during the execution
- Offers generic online mechanisms

Expectations

Validate Offline Configuration with AMH

- Implement a parametric algorithm using AMH
- Use it together with automatic configurators

Validate Online Control with AMH

- Implement online models and mechanisms
- Use them together with static algorithms
Case Study

MOLS: Multi-objective Local Search Algorithms

- Efficient metaheuristics
- Used on many problems (e.g., scheduling, routing, assignment)
- Many strategies and parameters

![Diagram showing Pareto optimal set]
Case Study: Multi-objective Local Search Algorithms

Blot et al.  AMH: a Framework to Design Adaptive MetaHeuristics
Protocol: Exhaustive Analysis vs Automatic Design

Permutation Flowshop Scheduling Problem

- Classical Taillard instances
- Bi-objective optimisation
  - Makespan
  - Flowtime

189 MOLS Configurations

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Parameter values</th>
</tr>
</thead>
<tbody>
<tr>
<td>initStrat</td>
<td>{rand, neh, ig}</td>
</tr>
<tr>
<td>selectStrat</td>
<td>{all, rand, newest, oldest}</td>
</tr>
<tr>
<td>selectSize</td>
<td>{1, 3}</td>
</tr>
<tr>
<td>explorStrat</td>
<td>{all, imp, ndom}</td>
</tr>
<tr>
<td>explorRef</td>
<td>{pick, arch}</td>
</tr>
<tr>
<td>explorSize</td>
<td>{1, 3}</td>
</tr>
</tbody>
</table>

Automatic Design Tool

- MO-ParamILS
- Performance indicators
  - Convergence
  - Spread
Result: Exhaustive Analysis vs MO-ParamILS

Exhaustive computational time: 115 days; MO-ParamILS: 7 days

PFSP Taillard instances – 50 jobs

Optimal Configurations

<table>
<thead>
<tr>
<th>Init</th>
<th>Selection</th>
<th>Exploration</th>
</tr>
</thead>
<tbody>
<tr>
<td>ig</td>
<td>oldest</td>
<td>3</td>
</tr>
<tr>
<td>ig</td>
<td>rand</td>
<td>3</td>
</tr>
<tr>
<td>ig</td>
<td>all</td>
<td>-</td>
</tr>
<tr>
<td>ig</td>
<td>newest</td>
<td>3</td>
</tr>
<tr>
<td>ig</td>
<td>all</td>
<td>-</td>
</tr>
<tr>
<td>ig</td>
<td>rand</td>
<td>1</td>
</tr>
<tr>
<td>ig</td>
<td>newest</td>
<td>3</td>
</tr>
<tr>
<td>ig</td>
<td>newest</td>
<td>3</td>
</tr>
<tr>
<td>ig</td>
<td>oldest</td>
<td>3</td>
</tr>
<tr>
<td>ig</td>
<td>oldest</td>
<td>3</td>
</tr>
</tbody>
</table>

[Blot et al., GECCO 2017]
Result: Exhaustive Analysis vs MO-ParamILS

Exhaustive computational time: 115 days; MO-ParamILS: 7 days

PFSP Taillard instances – 50 jobs

Optimal Configurations

<table>
<thead>
<tr>
<th>Init</th>
<th>Selection</th>
<th>Exploration</th>
</tr>
</thead>
<tbody>
<tr>
<td>ig</td>
<td>oldest</td>
<td>imp 3</td>
</tr>
<tr>
<td>ig</td>
<td>rand</td>
<td>imp 3</td>
</tr>
<tr>
<td>ig</td>
<td>all</td>
<td>arch 1</td>
</tr>
<tr>
<td>ig</td>
<td>newest</td>
<td>pick 3</td>
</tr>
<tr>
<td>ig</td>
<td>all</td>
<td>arch 1</td>
</tr>
<tr>
<td>ig</td>
<td>rand</td>
<td>arch 1</td>
</tr>
<tr>
<td>ig</td>
<td>newest</td>
<td>arch 1</td>
</tr>
<tr>
<td>ig</td>
<td>oldest</td>
<td>arch 1</td>
</tr>
</tbody>
</table>

[Blot et al., GECCO 2017]
Result: Exhaustive Analysis vs MO-ParamILS

Exhaustive computational time: 115 days; MO-ParamILS: 7 days

PFSP Taillard instances – 50 jobs

Optimal Configurations

<table>
<thead>
<tr>
<th>Init</th>
<th>Selection</th>
<th>Exploration</th>
</tr>
</thead>
<tbody>
<tr>
<td>ig</td>
<td>oldest</td>
<td>3</td>
</tr>
<tr>
<td>ig</td>
<td>rand</td>
<td>3</td>
</tr>
<tr>
<td>ig</td>
<td>all</td>
<td>-</td>
</tr>
<tr>
<td>ig</td>
<td>newest</td>
<td>3</td>
</tr>
<tr>
<td>ig</td>
<td>all</td>
<td>-</td>
</tr>
<tr>
<td>ig</td>
<td>rand</td>
<td>1</td>
</tr>
<tr>
<td>ig</td>
<td>newest</td>
<td>3</td>
</tr>
<tr>
<td>ig</td>
<td>newest</td>
<td>3</td>
</tr>
<tr>
<td>ig</td>
<td>oldest</td>
<td>3</td>
</tr>
<tr>
<td>ig</td>
<td>oldest</td>
<td>3</td>
</tr>
</tbody>
</table>
Validate Offline Configuration with AMH

▶ Implement a parametric algorithm using AMH
▶ Use it together with automatic configurators

Validate Online Control with AMH

▶ Implement online models and mechanisms
▶ Use them together with static algorithms
Conclusion

AMH: *Adaptive MetaHeuristics*[^4]

- Stand-alone C++ framework
- Tested with MOLS
- Facilitates offline configuration
- Enable online control

Take-home Message

- Keep designing alternative strategies
- Adapt automatically your algorithms

[^4]: https://github.com/amh-framework