# Evolution of Algorithm Portfolio for Solving Strategies

Alice Tarzariol

Università degli Studi di Udine

June 21, 2019

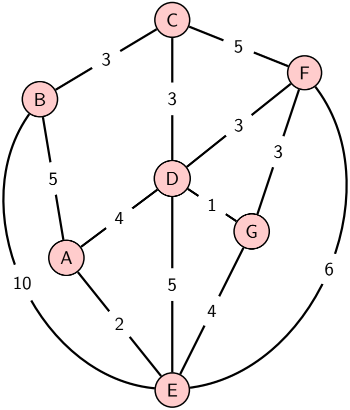
# A Scenario

# Consider a "difficult" problem...

- Many frequent instances that follow a distribution.
- E.g., finding a delivery path that minimizes the time/distance of its tour (*TSP*).

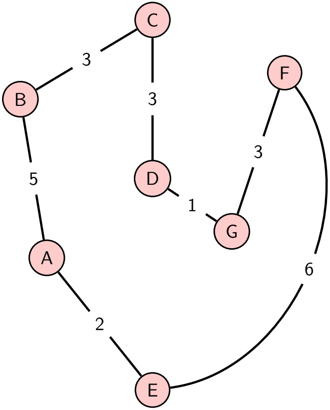
# Consider a “difficult” problem...

- Many frequent instances that follow a distribution.
- E.g., finding a delivery path that minimizes the time/distance of its tour (*TSP*).
- Models:
  - Answer Set Programming
  - Constraint Programming
  - Boolean Satisfiability
  - ...



# Consider a “difficult” problem...

- Many frequent instances that follow a distribution.
- E.g., finding a delivery path that minimizes the time/distance of its tour (*TSP*).
- Models:
  - Answer Set Programming
  - Constraint Programming
  - Boolean Satisfiability
  - . . .
- Run *a* solver to explore the searching space and find a solution.



# Solver Selection

- Given a portfolio of solvers $\mathcal{P} = \{A, B, C\}$ suitable for this problem.
- And a new problem instance $x$, drawn from the previous distribution.
- Which solver should be run on $x$?

Solver $A$ | Solver $B$ | Solver $C$

# Solver Selection

- Given a portfolio of solvers $\mathcal{P} = \{A, B, C\}$ suitable for this problem.
- And a new problem instance $x$, drawn from the previous distribution.
- Which solver should be run on $x$?

| | | |
|---|---|---|
| Solver $A$ | Solver $B$ | Solver $C$ |

- Knowledge obtained from previous problem instances.
- Solved by elements in $\mathcal{P}$ and measured according to a certain performance metric.

# Previous Approach

# Winner Takes All

Solver $A$

Solver $B$

Solver $C$

| | $A$ | $B$ | $C$ |
|---|---|---|---|
| $i_1$ | 55 | 3 | 5 |
| $i_2$ | 40 | 100 | 1 |
| $i_3$ | 6 | 100 | 150 |
| $i_4$ | 4 | 70 | 109 |
| $i_5$ | 45 | 5 | 150 |
| $avg$ | 30 | 56.6 | 83 |

## Simplifications

- Instances: Larger dataset.
- Values: Approximate performance of algorithms.

# Winner Takes All

SOLVER $A$

~~SOLVER $B$~~

~~SOLVER $C$~~

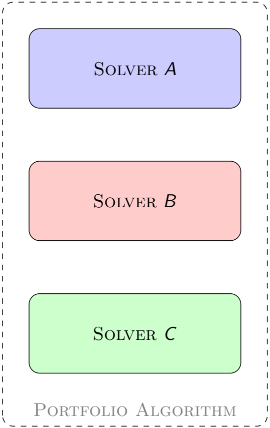
| | $A$ | $B$ | $C$ |
|---|---|---|---|
| $i_1$ | 55 | 3 | 5 |
| $i_2$ | 40 | 100 | 1 |
| $i_3$ | 6 | 100 | 150 |
| $i_4$ | 4 | 70 | 109 |
| $i_5$ | 45 | 5 | 150 |
| $avg$ | 30 | 56.6 | 83 |

## Simplifications

- Instances: Larger dataset.
- Values: Approximate performance of algorithms.

# Algorithm Portfolio: Selection and Scheduling

# Portfolio Approach

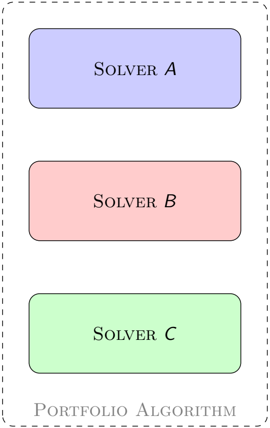



## Features Extraction

Static and/or Semi-static and/or Dynamic.

| $x_1$ | $x_2$ | $A$ | $B$ | $C$ |
|---|---|---|---|---|
| $a$ | $b$ | 55 | 3 | 5 |
| $a$ | $b$ | 40 | 100 | 1 |
| $c$ | $d$ | 6 | 100 | 150 |
| $a$ | $d$ | 4 | 70 | 109 |
| $c$ | $b$ | 45 | 5 | 150 |

# Portfolio Approach: Algorithm Selection

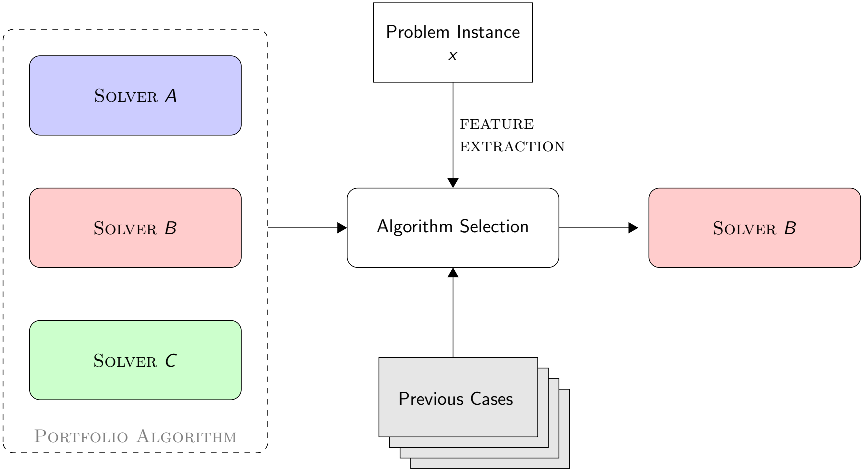



## Features Extraction

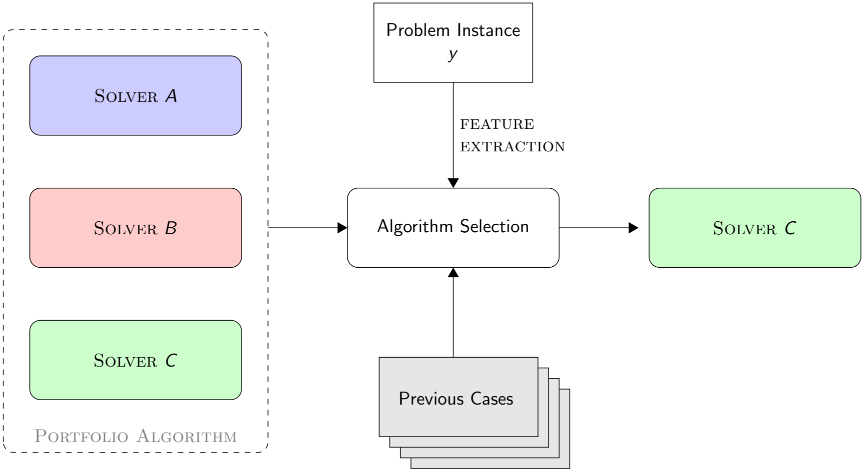
Static and/or Semi-static and/or Dynamic.

| $x_1$ | $x_2$ | $A$ | $B$ | $C$ |
|---|---|---|---|---|
| $a$ | $b$ | 55 | 3 | 5 |
| $a$ | $b$ | 40 | 100 | 1 |
| $c$ | $d$ | 6 | 100 | 150 |
| $a$ | $d$ | 4 | 70 | 109 |
| $c$ | $b$ | 45 | 5 | 150 |

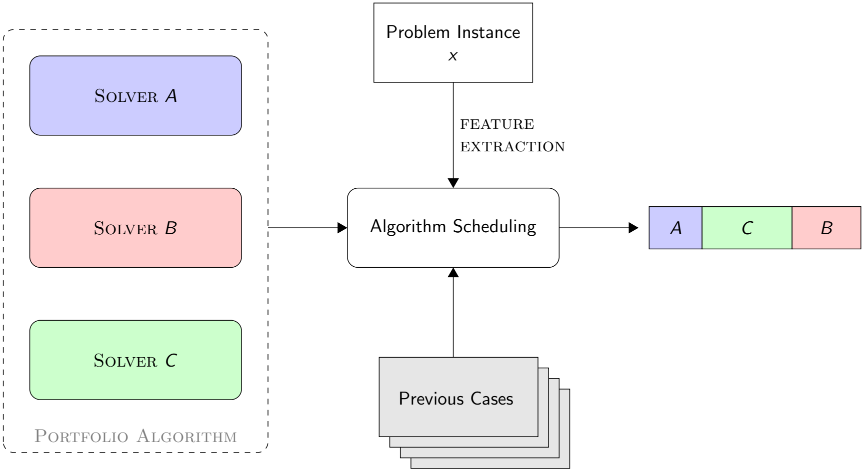
# Algorithm Selection



# Algorithm Selection



# Algorithm Scheduling



# Considerations

- Algorithm Scheduling is more robust.
- Useful if many instances are solved within brief time by different solvers.
- **Parallel Portfolio**: scheduling with multi-core architectures.

# Considerations

- Algorithm Scheduling is more robust.
- Useful if many instances are solved within brief time by different solvers.
- **Parallel Portfolio**: scheduling with multi-core architectures.

### Algorithm Selection

- SATzilla
- ME-ASP
- CLASPFOLIO

### Algorithm Scheduling

- CPHYDRA
- SUNNY (parallel)
- ASPEED (parallel)

- Both approaches: CLASPFOLIO 2.

# Considerations (Cont.)

## Implementations

- Several approaches: *Regression/Classification*, *Eager/Lazy*, etc.
- Further components and statistical techniques:
  - *Pre-Solvers*
  - *Backup Solver*
  - *Subset Portfolio Selection*
  - etc...

# Considerations (Cont.)

## Implementations

- Several approaches: *Regression/Classification*, *Eager/Lazy*, etc.
- Further components and statistical techniques:
  - *Pre-Solvers*
  - *Backup Solver*
  - *Subset Portfolio Selection*
  - etc...

## Portfolio Design

- Outperform the Single Best Solver even if sub-optimal solution.
- The portfolio solvers must present complementary strengths.

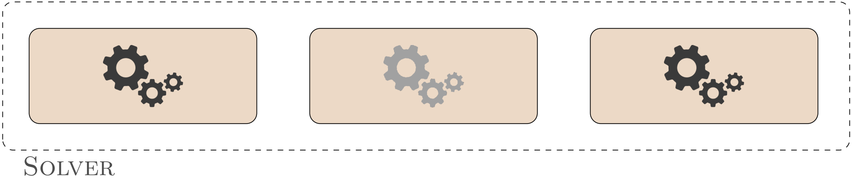
# Algorithm Configuration

# Before defining a portfolio of solvers...

- How we can find a solver that "performs well" for the considered distribution?

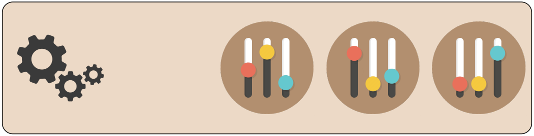
# Before defining a portfolio of solvers...

- How we can find a solver that "performs well" for the considered distribution?
- Many possibilities: e.g. local search techniques, DPLL, CDCL, etc.



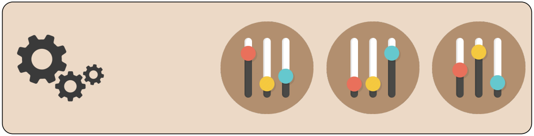
# Hyper-parameters

- Beside choosing the solving "core", the **hyper-parameters** of solvers further guide the behavior of algorithms.



# Hyper-parameters

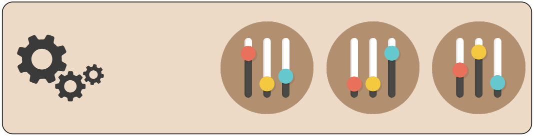
- Beside choosing the solving "core", the **hyper-parameters** of solvers further guide the behavior of algorithms.



- Huge configuration space: mix of continuous, integer and categorical domains.

# Hyper-parameters

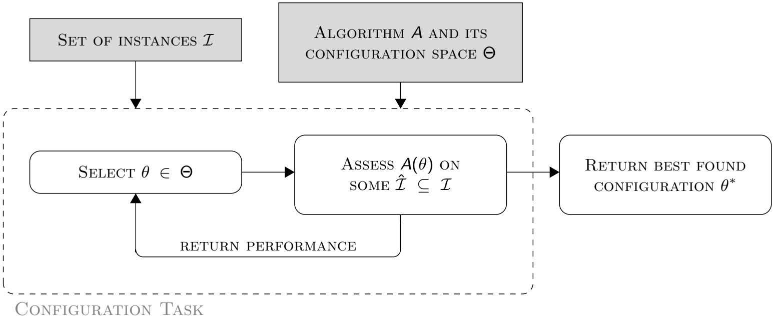
- Beside choosing the solving "core", the **hyper-parameters** of solvers further guide the behavior of algorithms.



- Huge configuration space: mix of continuous, integer and categorical domains.
- Manually tuned by domain experts: considerable time and efforts.
- Usually optimal configurations are not considered.

# Algorithm Configuration

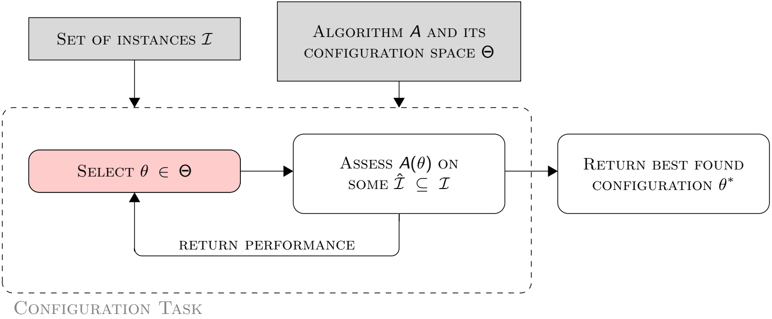
- Find automatically a configuration that performs optimally over the problem instance distribution.




- Empirical studies shown that it outperform hyperparameters tuning.

# Algorithm Configuration

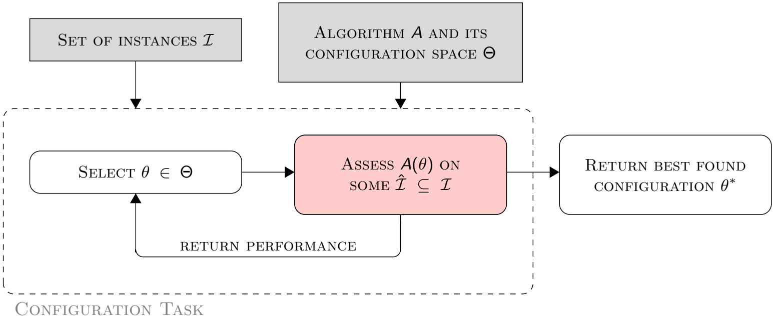
- Find automatically a configuration that performs optimally over the problem instance distribution.




- Empirical studies shown that it outperform hyperparameters tuning.

# Algorithm Configuration

- Find automatically a configuration that performs optimally over the problem instance distribution.

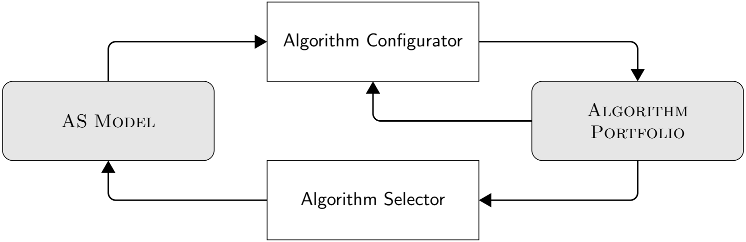



- Empirical studies shown that it outperform hyperparameters tuning.

# Combination

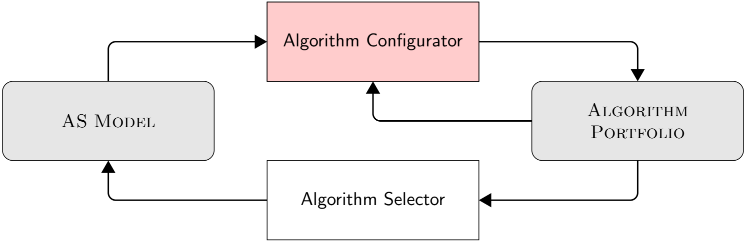
# Hydra

- Algorithm Selection that does not require a pre-defined portfolio of algorithm, but just a parametrized solver $A$.



# Hydra

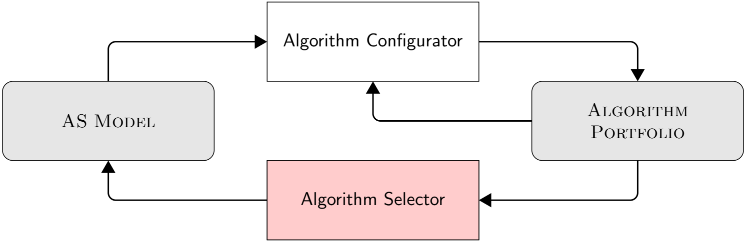
- Algorithm Selection that does not require a pre-defined portfolio of algorithm, but just a parametrized solver $A$.




- Algorithm Configuration extends a portfolio of $A$ configurations.

# Hydra

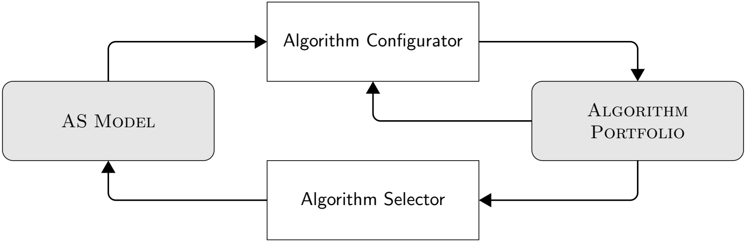
- Algorithm Selection that does not require a pre-defined portfolio of algorithm, but just a parametrized solver $A$.




- Algorithm Configuration extends a portfolio of $A$ configurations.
- Then it applies the Algorithm Selector to learn an AS Model.

# Hydra

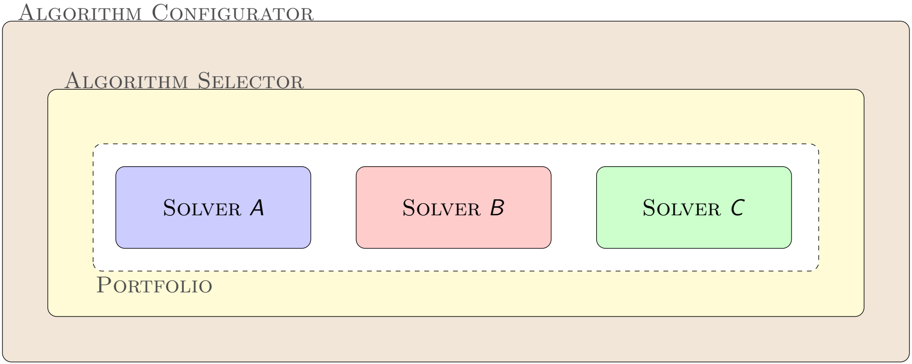
- Algorithm Selection that does not require a pre-defined portfolio of algorithm, but just a parametrized solver $A$.




- Algorithm Configuration extends a portfolio of $A$ configurations.
- Then it applies the Algorithm Selector to learn an AS Model.
- Continue until a stopping criterion is met.

# Autofolio

- Algorithm Configuration over Algorithm Selection systems with hyperparameters.
- Define the best setting that will exploit the portfolio solvers.



# Evolution of Algorithm Portfolio for Solving Strategies

- Algorithm Portfolio allows to exploit the whole set of solvers.
- But knowledge is required to find a suitable portfolio.
- Algorithm Configuration extends these systems.

# Evolution of Algorithm Portfolio for Solving Strategies

- Algorithm Portfolio allows to exploit the whole set of solvers.
- But knowledge is required to find a suitable portfolio.
- Algorithm Configuration extends these systems.
- Outstanding results: portfolio systems are not allowed to participate to "standard" solving competitions.
- ASlib: train and evaluate new systems.

# Evolution of Algorithm Portfolio for Solving Strategies

- Algorithm Portfolio allows to exploit the whole set of solvers.
- But knowledge is required to find a suitable portfolio.
- Algorithm Configuration extends these systems.
- Outstanding results: portfolio systems are not allowed to participate to "standard" solving competitions.
- ASlib: train and evaluate new systems.

Questions?