Overview

- **Object oriented system for algorithm selection**
  - Algorithms automatically selected and executed
- **All core components of algorithm selection are explicitly present**
  - User chooses concrete implementation for specific problem scenario
- **Minimal user-input required**
  - Executables + scripts for extracting performance and feature values
- **Applied to the Generalized Assignment Problem**

Elements of Algorithm Selection

**Core elements**

- Algorithm space \( A \)
- Instance space \( I \)
  - a distribution over \( I \) : \( D \)
- Performance mapping \( p: I \times A \rightarrow R \)
- Selection mapping \( \lambda: I \rightarrow A \)

**Helper elements**

- Feature space \( F \)
  - \( \Rightarrow \) selection mapping: \( \lambda: (I \rightarrow F) \rightarrow A \)
- Training data \( H \): set of tuples \( (i, p_\phi, a, p(a,i)) \)
- Selection mapping init strategy: \( \beta: H \rightarrow \Lambda \)
  - \( \Rightarrow \) e.g. linear regression, decision tree, K-NN

**Note:** all this is for deterministic performance

What can the system be used for?

- **Executing algorithms** and keeping track of results
  - Executables are automatically called
- **Performance is extracted and added to database**
- **Standard one-shot offline algorithm selection**
  - Uses WEKA or user-defined ML methods
- **Online algorithm selection**
  - Process new data to improve selection mapping
  - Starting from zero (no training data)
  - Active learning?

**Idea for future:** human in the loop

- **Identify instance regions with poor performance**
- **Identify the cause:**
  - No algorithm performs well \( \Rightarrow \) develop new
  - Features cannot distinguish \( \Rightarrow \) develop new
  - Init strategy is not good enough \( \Rightarrow \) find better

Using the System: Workflow

For each feature:

- **Obtain executables**
- **Write script for performance extraction**
  - For each feature:
    - **Write script for feature value extraction**

Choose a strategy \( \beta \) for creating selection mappings

\( \Rightarrow \) An interface to WEKA is implemented

Add training data (optional)

Start performing online algorithm selection

**Algorithm I** Processing online instances

1: for Online instance \( i \) do
2: \( \lambda = \beta(H) \) (get selection mapping)
3: \( f_i = \text{Extract required feature values} \)
4: \( \text{Add feature values to database} \)
5: \( a = \lambda(f_i) \) (select algorithm)
6: \( \text{Run algorithm} \)
7: \( \text{Extract performance from result file} \)
8: \( \text{Add performance to database} \)

Use Case: Generalized Assignment Problem

**Problem:** assign each job to exactly 1 agent, with job-specific resource usage and a maximum resource capacity for each agent

**Goal:** minimise assignment costs

**Applications:** scheduling, routing, production planning...

\[
\text{minimize } \sigma = \sum_{i \in I} \sum_{j \in J} a_{ij} x_{ij} \quad \text{subject to } \sum_{j \in J} a_{ij} x_{ij} \leq b_i \quad \forall i \in I, \quad \sum_{i \in I} x_{ij} = 1 \quad \forall j \in J, \quad x_{ij} \in \{0,1\} \quad \forall i \in I \text{ and } \forall j \in J.
\]

**Why GAP as use case?**

\( \Rightarrow \) It often must be solved repeatedly in limited time

- Models problems at the Operational level
- Occurs as a subroutine when solving bigger problems

**Challenges**

- Obtaining executables (and getting them to work)
- Identifying good features
- Deciding how to measure performance

Specific application must be taken into account

References


Contact

Hans Degroote
KU Leuven Department of CS
Email: Hans.Degroote@kuleuven.be