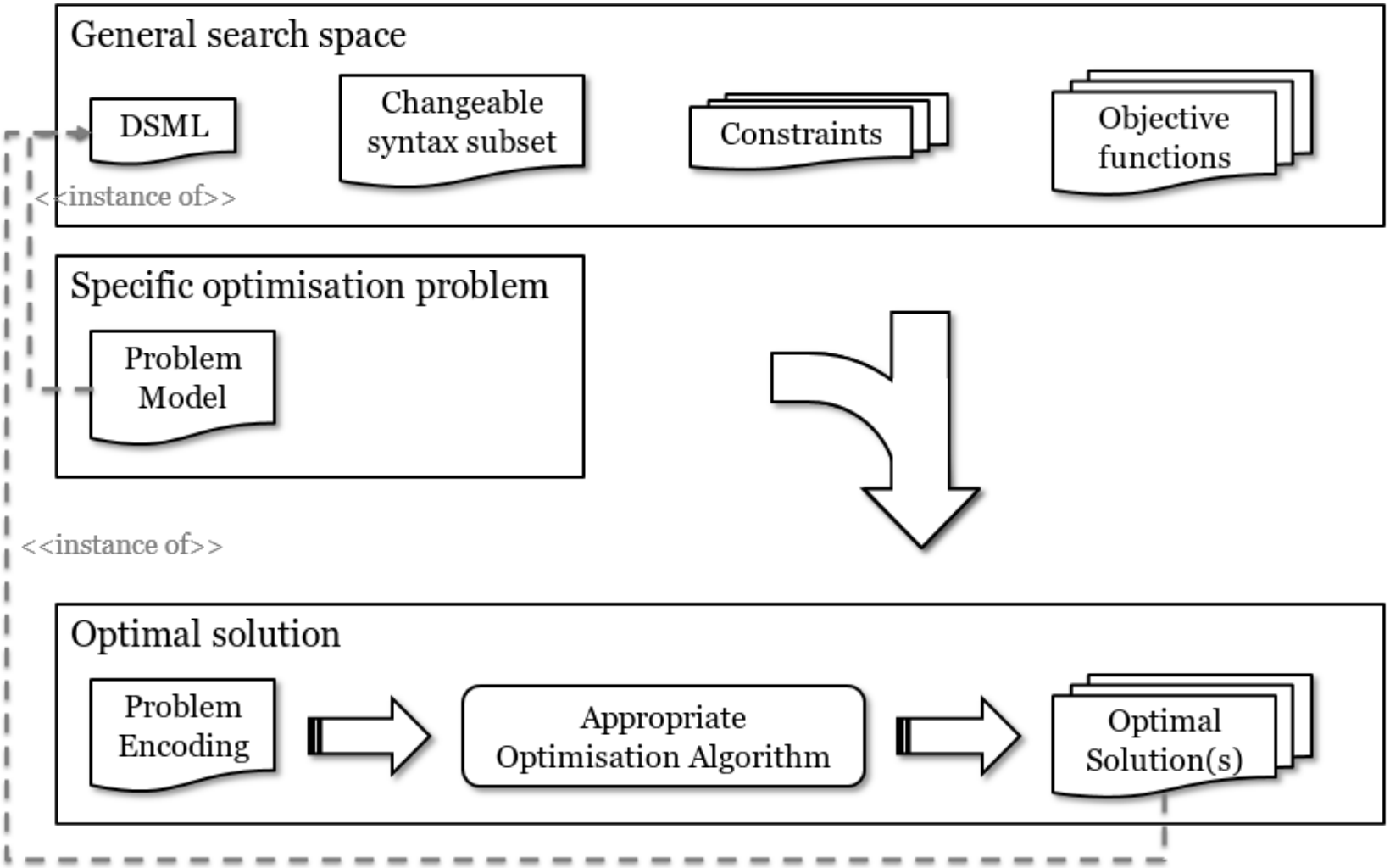
Towards Scalable Search-Based Model Engineering with MDEOptimiser Scale

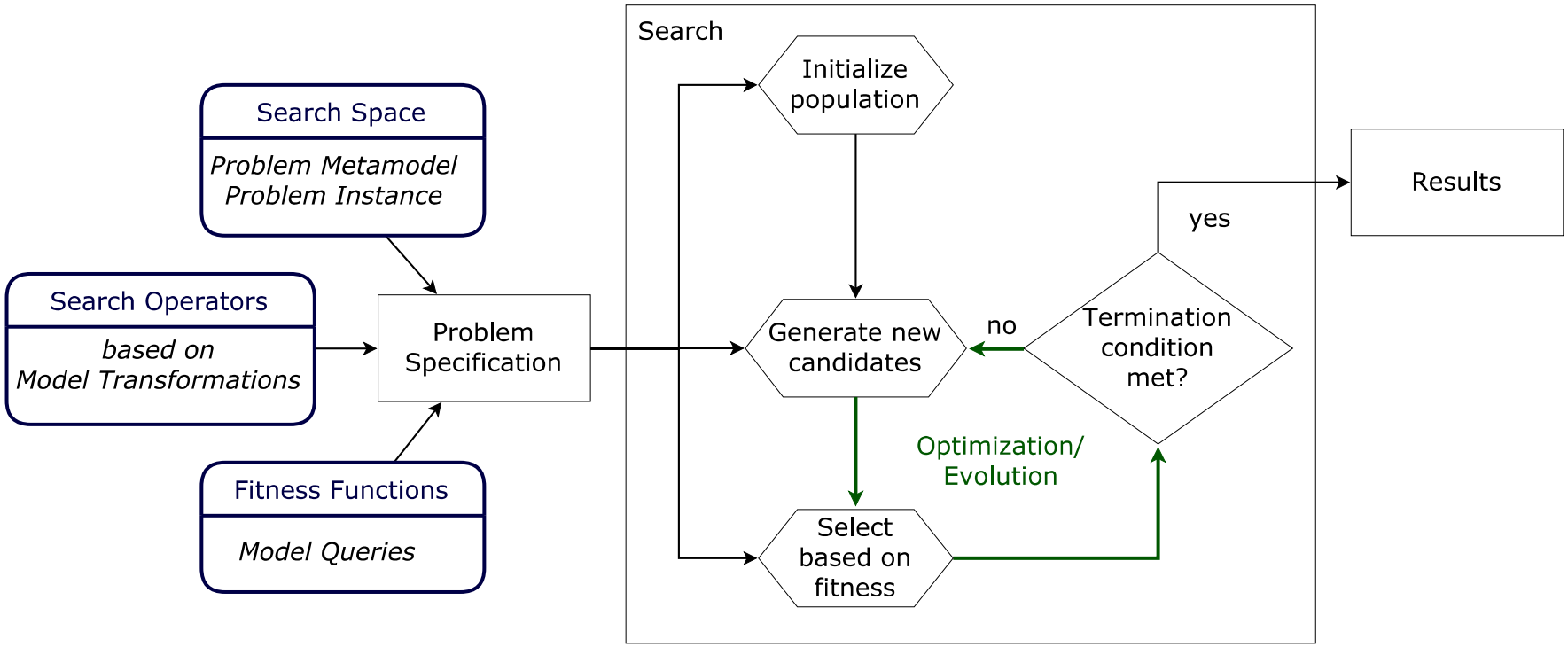
Alexandru Burdusel & Steffen Zschaler
alexandru.burdusel@kcl.ac.uk
szschaler@acm.org
King’s College London

16/09/2019

Introduction to SBME



SBME Key Idea



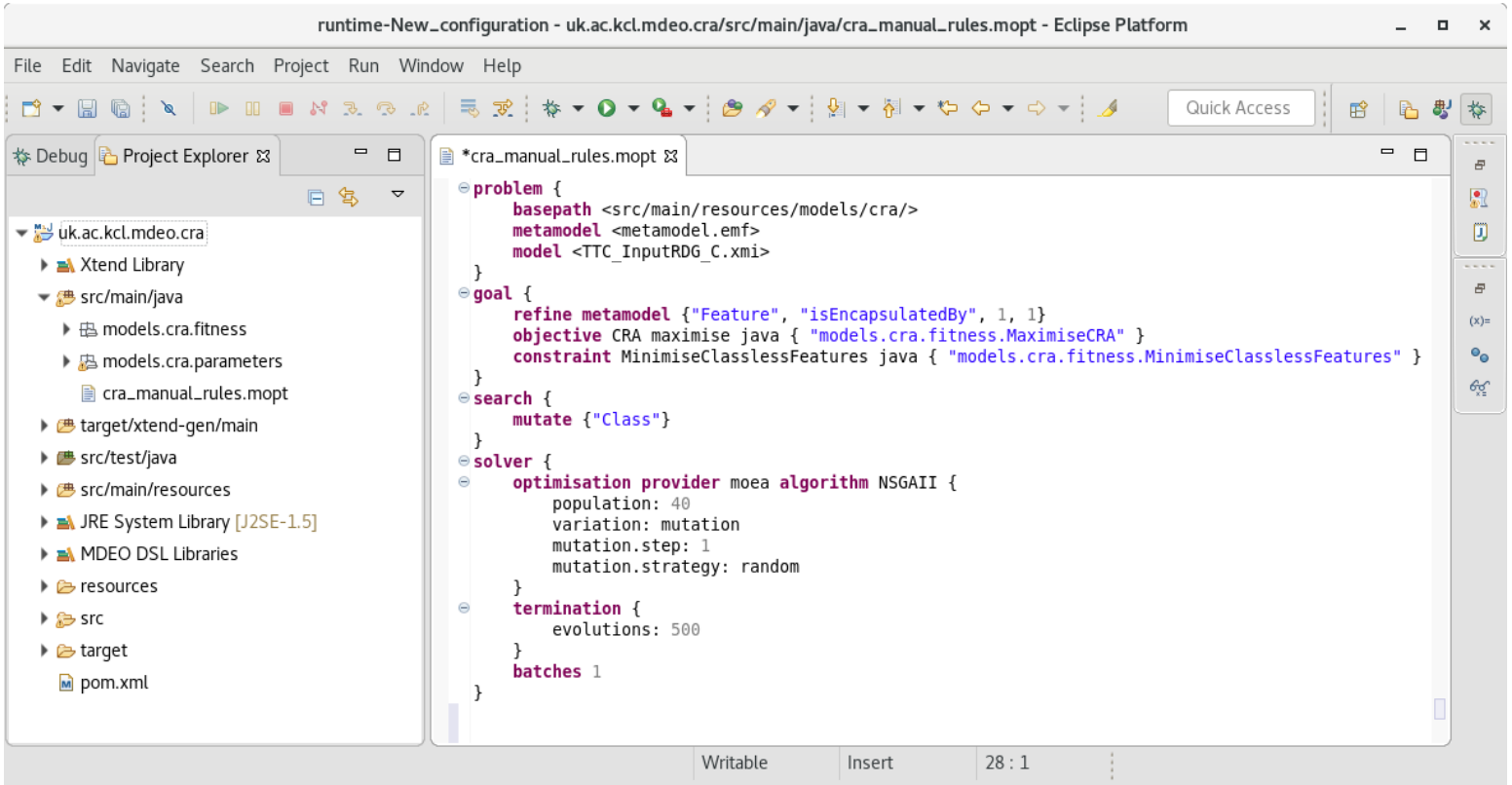
- Search Operators: Mutation, Crossover
- Solution Encoding: Rule-Based(MOMoT) , Model-Based (MDEOptimiser)

Running SBME Experiments

- **SBME approaches are often evaluated through case study based evaluation.**
- **Result metrics have to be evaluated using statistical tests (eg: Mann-Whitney U test, ANOVA, Kruskal-Wallis, Cohen's D etc.)**
- **To perform statistical significance tests we need at least 30 samples for a single configuration to calculate the metrics.**
- **Some authors recommend 1000 samples for some tests*.**

- *A. Arcuri, L. Briand *A Hitchhiker's guide to statistical tests for assessing randomized algorithms in software engineering*

GUI Experiments



Problems with this approach?

- Experiments not executed in isolation.
- (Many) manual steps required. It’s easy to make mistakes
- Difficult to reproduce the experiment conditions.

Headless Experiments

- Solution to running experiments using a GUI?

- Run in headless mode on a standalone server. Better application isolation.

- Lower risk of other OS/User background tasks starting unexpectedly.

```
alxbrd@local:/
File  Edit  View  Search  Terminal  Help
[alxbrd@local /]$ ssh exp@server 'java -classpath "plugins/*" at.ac.tuwien.big.momot.interfaces.cli.Run ArchitectureSearch.momot'
```

Problems with this approach?

- Still (many) manual steps required.

- Complex task if there are many experiments to run.

Scripted Headless Experiments

- Solution to manually running headless experiments?

- Use a script that loops through a set of given configurations and executes each one.

- Easy to scale to a large number of tools and experiment runs.

alxbrd@local:/

File Edit View Search Terminal Help

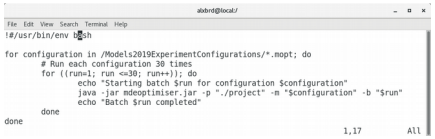
```
!#/usr/bin/env bash

for configuration in /Models2019ExperimentConfigurations/*.mopt; do
        # Run each configuration 30 times
        for ((run=1; run <=30; run++)); do
                echo "Starting batch $run for configuration $configuration"
                java -jar mdeoptimiser.jar -p "./project" -m "$configuration" -b "$run"
                echo "Batch $run completed"
        done
done
                                                                    1,17          All
```

Problems with this approach?

- Some experiment batches may take a long time to run(hours, days, weeks)

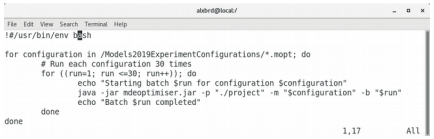
- Complex task if there are many experiments to run.

Distributed Headless Experiments

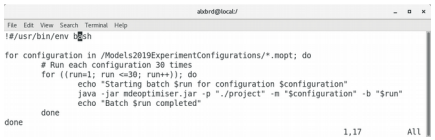


Server 1

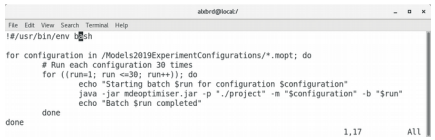
Server 2



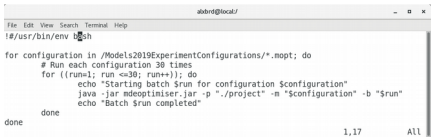
Server 3



Server 4



Server 5



Server 5

Problems with this approach?

- Servers have to be configured and initialised
- Deploying the right files to the right server and retrieving the results is not a trivial task.
- Hard to keep track of / restart failed jobs. Logging is difficult.
- More scripting required to automate this process.
- The generated results files have to be interpreted.

Introducing MDEOptimiser Scale

Design Goals

- Offer a text-based DSL for specifying SBME experiments;
- Require minimal user configuration and setup;
- Provide an extensible framework that makes it easy to add new tools and additional hardware support for automated experiments execution;
- Propose a common interface for collecting experiment metrics from SBME tools;
- Offer support for automatically interpreting experiment metrics using common statistical metrics.

```
MDEO Scale usage instructions

java -jar scale.jar [options...] arguments...
 -projectPath VAL : Tool base path used to load artifacts defined in the DSL.
 -specPath VAL    : Specification file name to execute.

Example: java -jar scale.jar  -projectPath VAL -specPath VAL
```

MDEOptimiser Scale DSL

```
infrastructure "aws batch" {
    type aws
    account "default"
    environment "compute_environment.json"
}
experiment "CRA Case Study" {
    parameters {
        batches: 30
        artifacts: "src/java/resources/models/"
    }
    model "TTC_InputRDG_A" {
        task "MDEO" {
            run "cra_model_a.mopt"
            dependencies "./libraries/cra.jar"
        }
        task "MOMoT" {
            run "cra_model_a.momot"
            dependencies "./libraries/cra.jar"
        }
    }
    model "TTC_InputRDG_B" {
        task "MDEO" {
            run "cra_model_b.mopt"
            dependencies "./libraries/cra.jar"
        }
        task "MOMoT" {
            run "cra_model_b.momot"
            dependencies "./libraries/cra.jar"
        }
    }
}
```

AWS Batch Compute Environment configuration

Define experiments with multiple input models

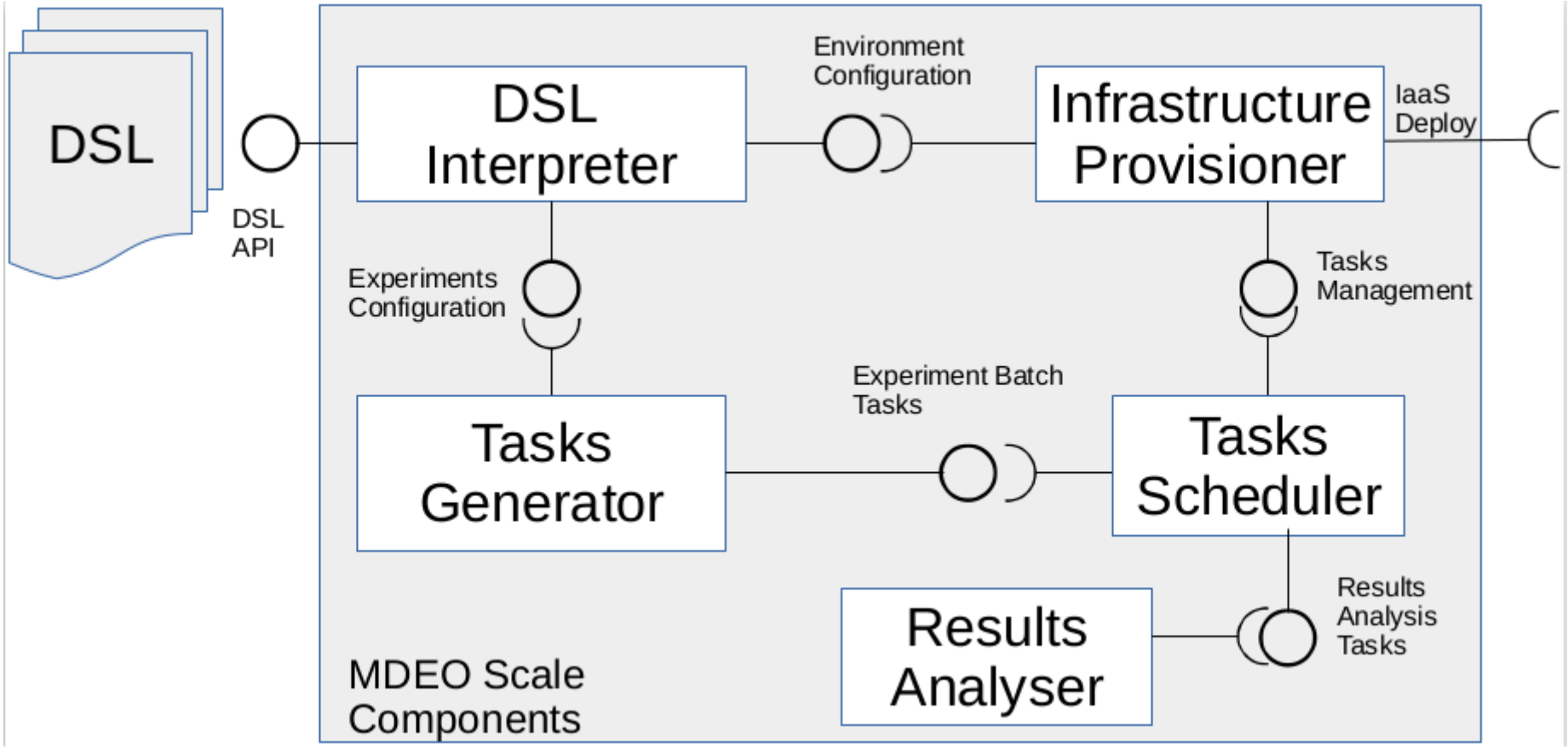
Set global experiment parameters

Group tools by input model

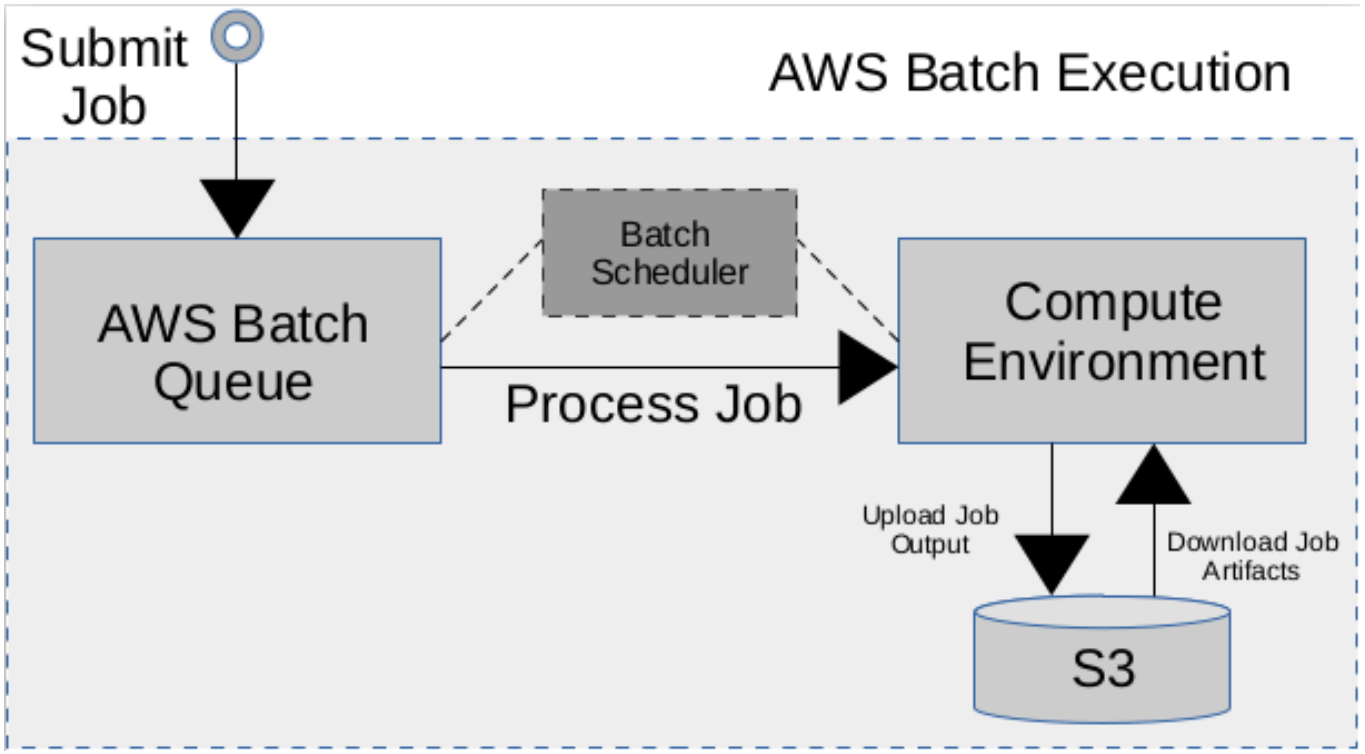
Specify classpath dependencies for each model.

Specify what tool spec to execute for each model. The tool type is inferred from the extension.

Tool Architecture



Experiments Infrastructure



- MDEOptimiser Scale uses Amazon Web Services Batch to process jobs.
- AWS Batch is a free service, only the used computing resources are charged.

Extension Points

- **Add new tools by implementing the IScaleTask interface**
- **Deploy a TaskExecutor service instance to a Docker container containing a running tool instance**

```
@Override
public Properties getContainerProperties() {

  var containerProperties = new Properties();

  containerProperties.setProperty("image", "mdeoscale/mdeoptimiser");
  containerProperties.setProperty("vcpus", "1");
  containerProperties.setProperty("memory", "2048");
  containerProperties.setProperty("command", "java -jar /var/app/scale-wrapper.jar");

  return containerProperties;
}
```

Extension Points TaskExecutor


```
{
  "job":{
    "task":{
      "className":"uk.ac.kcl.inf.mdeoptimiser.infrastructure.scale.interpreter.experiments.tasks.types.MDEOptimiser",
      "name":"MDEO",
      "modelName":"TTC_InputRDG_A",
      "experimentInstanceName":"2019-08-16-02-32-11-CRACaseStudy",
      "experimentName":"CRA Case Study",
      "taskFiles":{
        "2019-08-16-02-32-11-CRACaseStudy/TTC_InputRDG_A/MDEO/cra_model_a.mopt":{
          "path":"cra_model_a.mopt"
        },
        "2019-08-16-02-32-11-CRACaseStudy/TTC_InputRDG_A/MDEO/libraries/cra.jar":{
          "path":"libraries/cra.jar"
        }
      },
      "id":"MDEO_ee7fbf8a-0f81-4011-a515-988a5906c268",
      "command":"cra_model_a.mopt",
      "taskDependencies":{
        "2019-08-16-02-32-11-CRACaseStudy/TTC_InputRDG_A/MDEO/libraries/cra.jar":{
          "path":"libraries/cra.jar"
        }
      }
    },
    "jobName":"exp_CRACaseStudy_model_MDEO_task_TTC_InputRDG_A_batch_1",
    "batchNumber":1
  }
}
```


TaskExecuter receives a list of files to download from S3 to run this job.

Job ID used as a logging key.

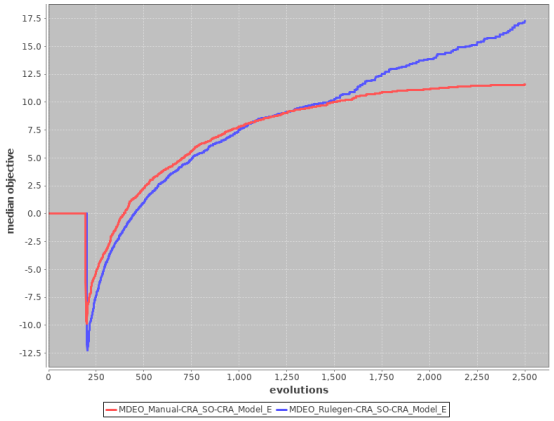
TaskExecutor Job Payload Example

Experiments Data Collection

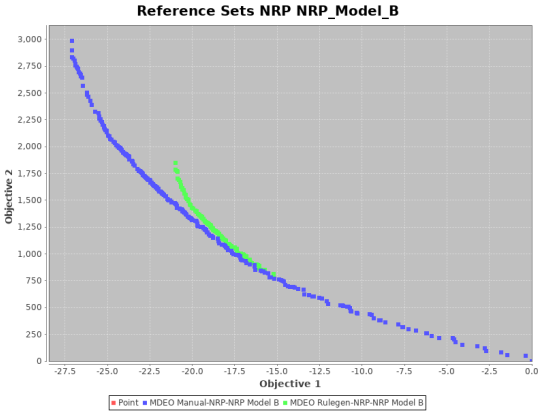
- **Using the instrumentation functionality provided by MOEAFramework:**
- **org.moeaframework.analysis.collector**

- **Supported collectors:**
- **NFE, Elapsed Time, Approximation Set, Population, Population Size**

Example Tool Outputs



Unary quality indicator chart



Pareto Front charts

tools	MDEO_Manual-CRA_SO-CRA_Model_E	MDEO_Rulegen-CRA_SO-CRA
MDEO_Manual-CRA_SO-CRA_Model_E P Value	1.0	1.3949E-10
MDEO_Manual-CRA_SO-CRA_Model_E U Value	450.0	884.0
MDEO_Manual-CRA_SO-CRA_Model_E Cohen D	0.0	Large
MDEO_Rulegen-CRA_SO-CRA_Model_E P Value	1.3949E-10	1.0
MDEO_Rulegen-CRA_SO-CRA_Model_E U Value	884.0	450.0
MDEO_Rulegen-CRA_SO-CRA_Model_E Cohen D	Large	0.0

Statistical Testing Tables

configurations	steps	metricMean	metricMedian	metricMax	metricMin	metricStdDev	skewness	kurtosis
MDEO_Manual CRA A	500	2.289	2.333	0.850	3.000	0.552	-0.679	-0.509
MDEO_Rulegen CRA A	500	3.000	3.000	3.000	3.000	0.000	NaN	NaN
MDEO_Manual CRA B	500	1.963	1.865	1.238	3.104	0.514	0.642	-0.032
MDEO_Rulegen CRA B	500	3.001	3.167	1.826	4.083	0.599	-0.470	-0.376
MDEO_Manual CRA C	500	2.177	2.224	1.148	3.240	0.572	-0.089	0.824
MDEO_Rulegen CRA C	500	3.137	3.129	2.110	3.806	0.428	-0.539	-0.039
MDEO_Manual CRA D	2000	5.423	5.191	3.557	7.041	0.837	0.068	0.339
MDEO_Rulegen CRA D	2000	9.742	9.863	7.634	12.273	1.257	-0.176	0.782
MDEO_Manual CRA E	2500	11.719	11.572	8.879	14.691	1.639	0.122	0.663
MDEO_Rulegen CRA E	2500	17.018	17.323	11.698	20.051	1.604	-1.106	-3.176

Summary Statistics Tables

Future Directions

- Support more IaaS providers
- Support for specifying additional tools directly in the DSL
- Expand beyond the field of SBME
- Add support for more metrics
- Support for additional algorithms to help identify the best solutions (Eg: clustering)

Conclusion

- MDEO Scale is an SBME experiment workflow DSL
- Supported tools: MDEOptimiser, MOMoT
- Looking forward to your feedback.
- Source code available on Github
- https://github.com/mde-optimiser/scale