Meta-learning as intelligent searching through model space

Norbert Jankowski & Krzysztof Grąbczewski

Department of Informatics
Nicolaus Copernicus University
Toruń, Poland
http://www.is.umk.pl/
norbert@is.umk.pl

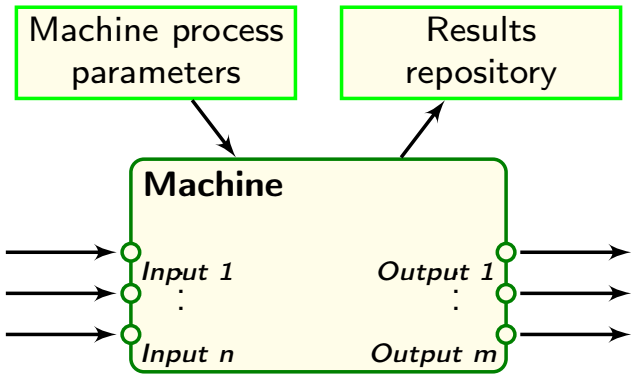
Outline

- Intelligent data mining software—Intemi
- Learning & Meta-learning
- General view of (almost) all Meta-learning algorithms
- Elements of presented meta-learning system
- Configuring of Meta-learning problem
- Decomposition of learning problem & Search space
- Graphs of generators & Intelligent machine generators
- Meta-learning loop, complexity and test task ordering
- Machine and machine evaluator, complexity approximation
- Approximation framework
- Summary: Elements of presented meta-learning system (again...)

Learning and meta-learning

A **learning problem** can be defined as $\mathcal{P} = \langle D, \mathcal{M} \rangle$, where $D \subseteq \mathcal{D}$ is a **learning dataset** and $\mathcal{M}$ is a **model space**.
Learning process of a *learning machine* $\mathcal{L}$ is a function $\mathcal{A}(\mathcal{L})$:

$$\mathcal{A}(\mathcal{L}) : \mathcal{K}_{\mathcal{L}} \times \mathcal{D} \to \mathcal{M} \tag{1}$$



- **Meta-learning is** another or rather specific **learning** of machine.
- ML = Learn how to learn, to learn as well as possible.

Restricted learning problem

The **restricted learning problem** is a **learning problem $\mathcal{P}$ with** defined **time limit** for providing the solution.

Such definition of the problem of learning from data is much **more realistic** in comparison to the original one (whenever it is used to research purpose or commercial purpose). It should be preferred also for research purposes. In a natural way, it reflects the reality of data analysis challenges, where deadlines are crucial...

The Goal of meta-learning

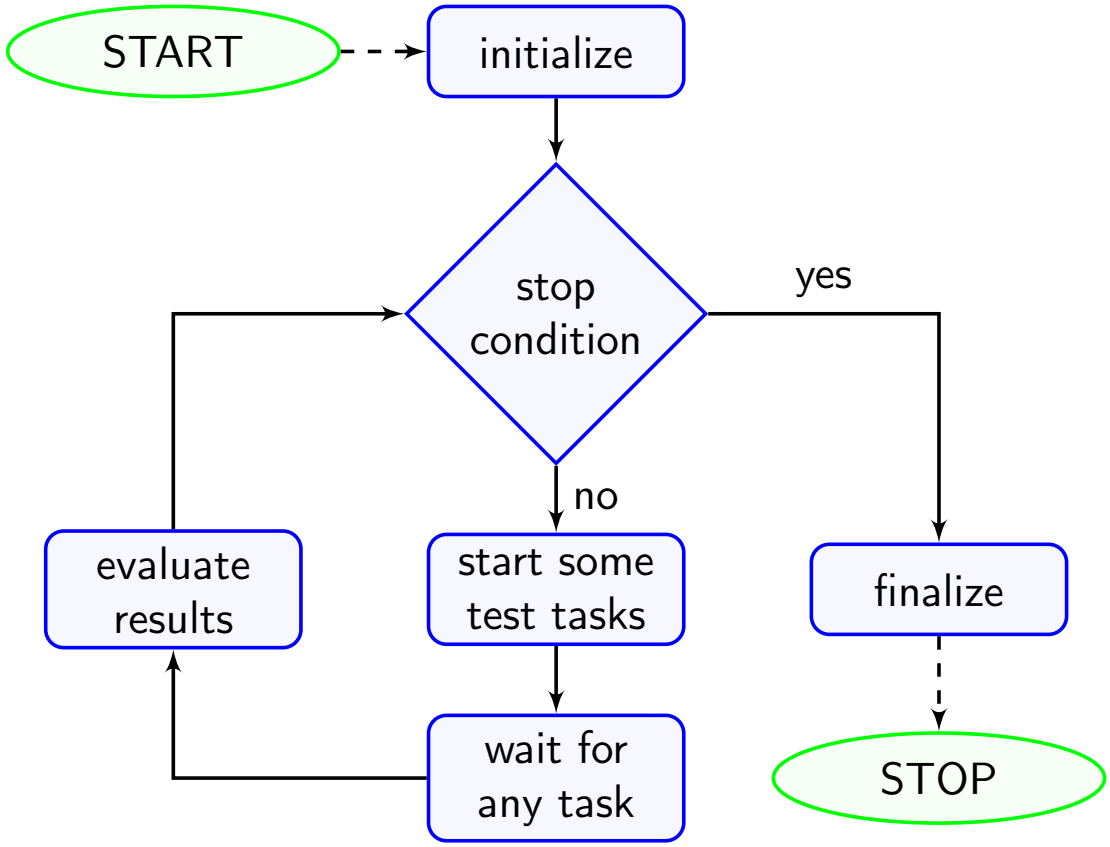
- Finding an **optimal model** for given data mining problem $\mathcal{P}$ is almost always NP hard.
- Meta-learning could find solutions which **at least** are not worst than the ones that can be found by human experts in data mining in given limited amount of time.
- Experts usually restrict their tests to a part of algorithms.

The Goal of meta-learning is

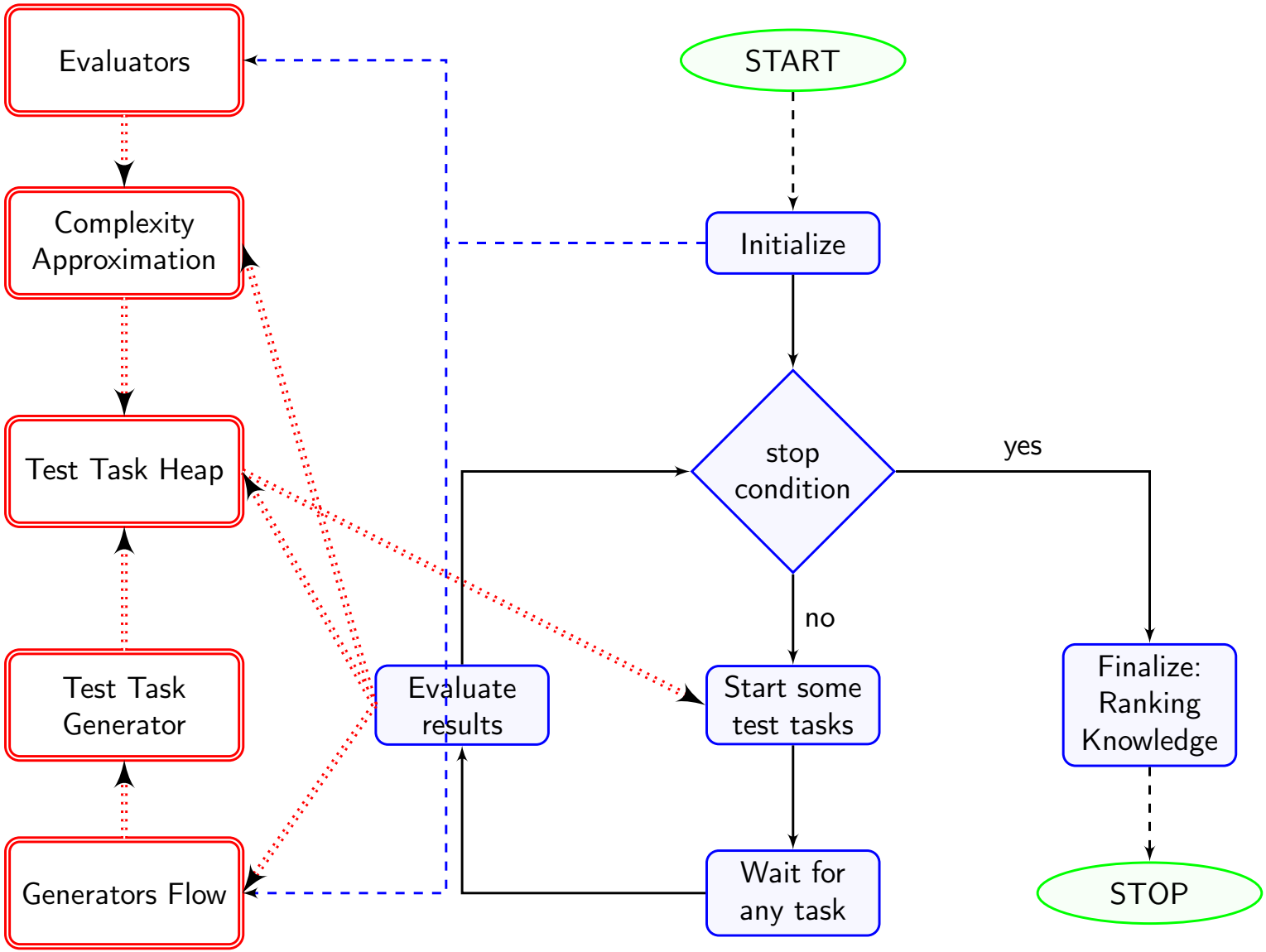
To maximize the probability of finding possibly best solution within a search space of given problem $\mathcal{P}$ in as short time as possible.

- Complexity is used to order machines.

General meta-learning scheme



Elements of presented meta-learning system



Configuring Meta-learning Problem

MLA need NOT be a mistery!

Appropriate configuring of MLA should enable find solution of any $\mathcal{P}$.

Ex.: If we want use it for classification or approximation, don't fix it inside MLA!

Configuration of MLA is be specified by

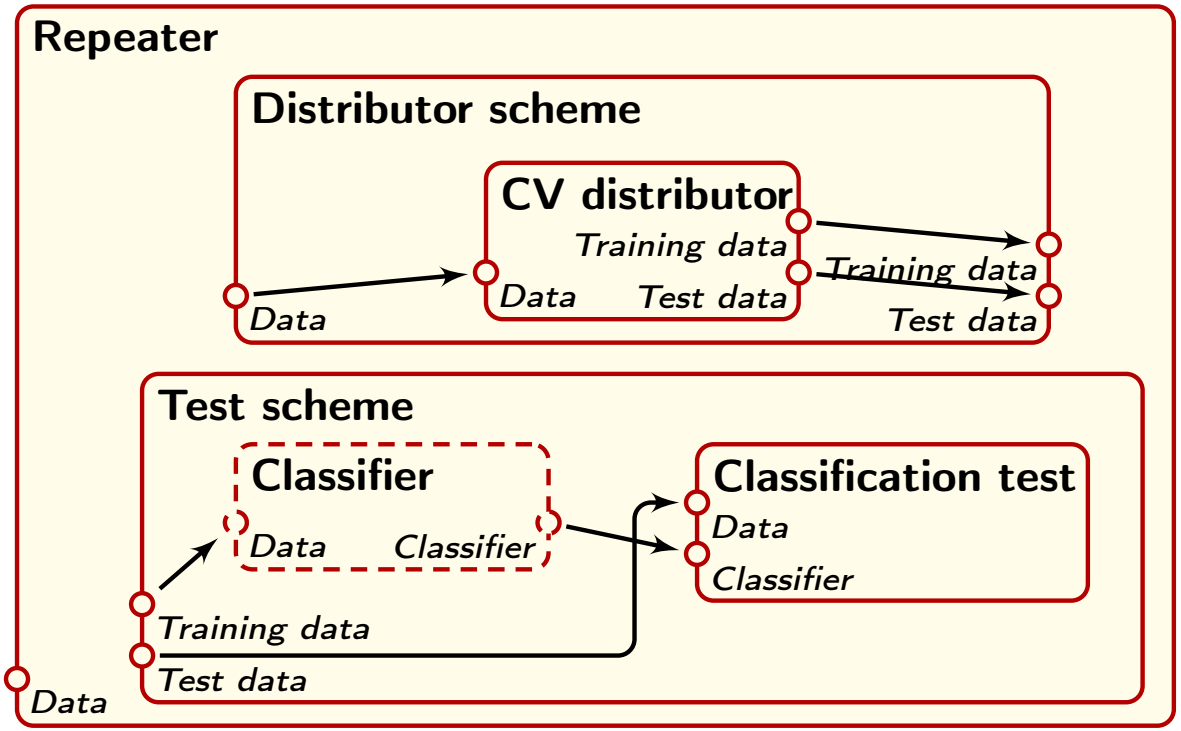
- Stating of Functional Searching Space
- Defining the Goal of Meta-learning (of problem $\mathcal{P}$)
- Defining the Stop Condition
- Defining the Attractiveness Module
- Defining Initial Meta-knowledge

Above elements of configuration guarantees that algorithm is universal.

Defining the Goal of Meta-learning for particular problem $\mathcal{P}$

Test procedure + evaluation of results

- Strict and flexible definition of the goal!



- Test procedure finished $\Rightarrow$ Results evaluation $\equiv$ Query
- Query definition = evaluation of results:
  - which nodes, which values, how to transform

Search space & Decomposition of learning problem

To decompose the problem $\mathcal{P} = \langle D, \mathcal{M} \rangle$ into **subproblems:**

$$\mathcal{P} = [\mathcal{P}_1, \ldots, \mathcal{P}_n], \qquad \mathcal{P}_i = \langle D_i, \mathcal{M}_i \rangle \tag{2}$$

In result model for the main problem $\mathcal{P}$ is:

$$m = [m_1, \ldots, m_n], \tag{3}$$

and the model space gets the form

$$\mathcal{M} = \mathcal{M}_0 \times \ldots \times \mathcal{M}_n. \tag{4}$$

The solution constructed by decomposition is often much easier to find.

The model m_i solving the subproblem $\mathcal{P}_i$, is the result of learning process

$$\mathcal{A}(\mathcal{L}_i) : \mathcal{K}_{\mathcal{L}_i} \times \mathcal{D}_i \to \mathcal{M}_i, \quad i = 1, \ldots, n, \tag{5}$$

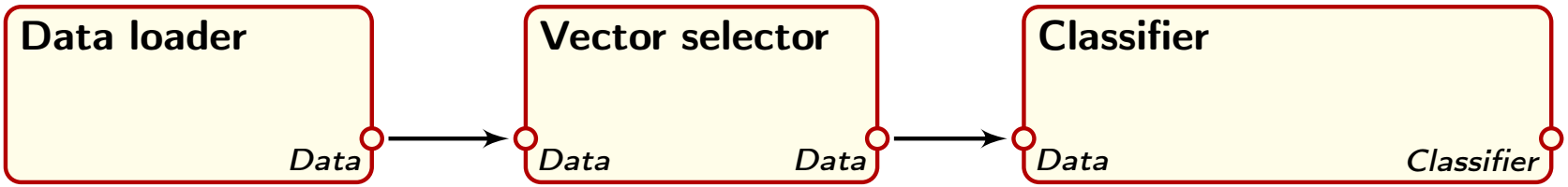
where

$$\mathcal{D}_i = \prod_{k \in K_i} \mathcal{M}_k, \tag{6}$$

and $K_i \subseteq \{0, 1, \ldots, i-1\}$, $\mathcal{M}_0 = \mathcal{D}$.

The main learning process $\mathcal{L}$ is decomposed to the vector

$$[\mathcal{L}_1, \ldots, \mathcal{L}_n]. \tag{7}$$

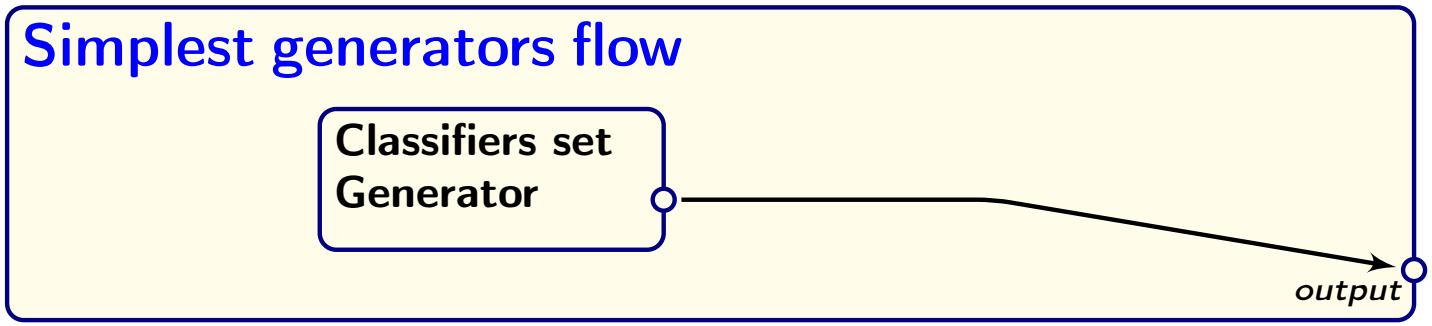


- Such **decomposition is often very natural**: a standardization or feature selection naturally precedes classification, a number of classifiers precede committee module etc.

Search space of MLA

Machine Configuration Generators & Generators Flow

- Fixed set of LM inside MLA is not flexible and strongly limits MLA!
- The goal of **machine configuration generators (MCG)** is to provide/produce machine configurations.
- The goal of **generators flow** is to provide machine configuration to MLA.
- Generators flow is a graph (DAG) of machine configuration generators.
- Each MCG may base on different meta-knowledge and may reveal different behavior, which in particular may even change in time (during the meta-learning progress).

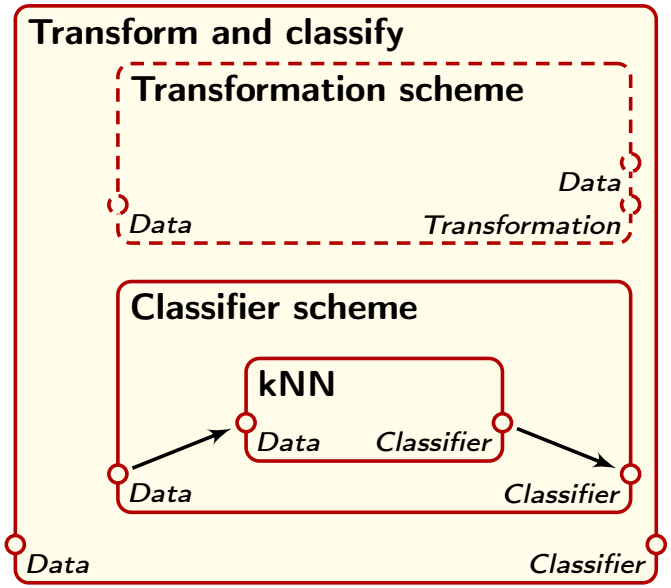


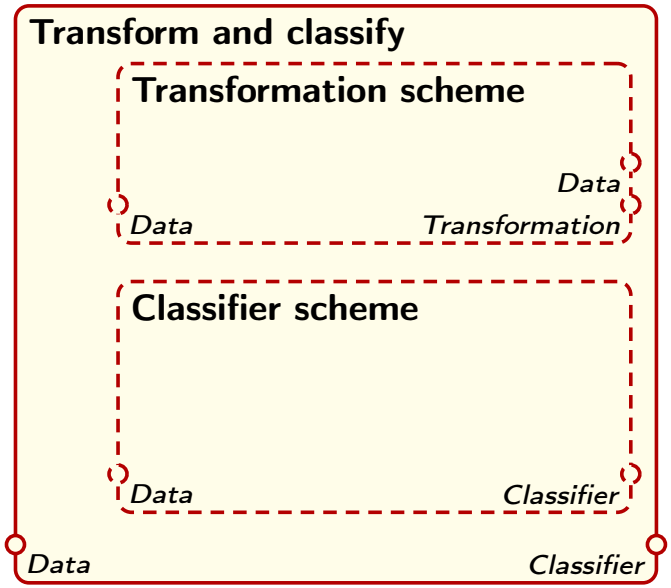
Set-based Generators

- The simplest and very useful machine generator is based on the idea to provide just an arbitrary (initial) sequence of machine configurations.
- Usually, it is convenient to have a few set-base generators in single generators flow
  - set-base generator of simple classifiers
  - set-base generator of classifiers
  - set-base generator of feature ranking
  - set-base generator of prototype selectors
  - set-base generator of committees ...

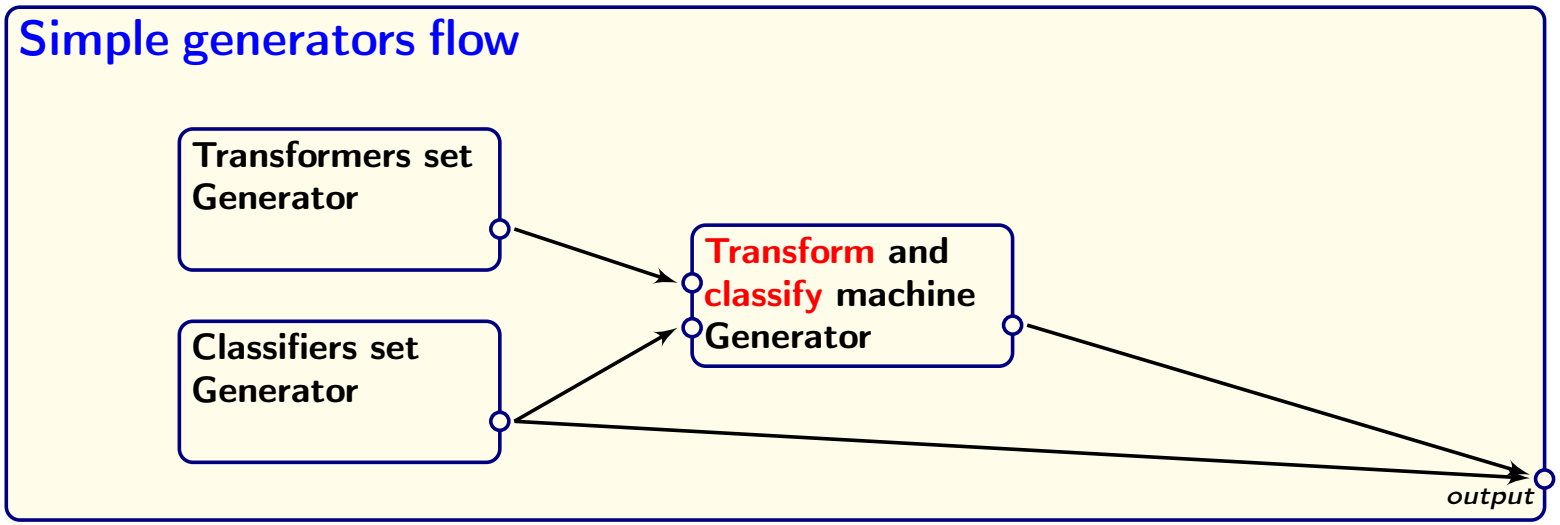
Template-based Generators

Templates

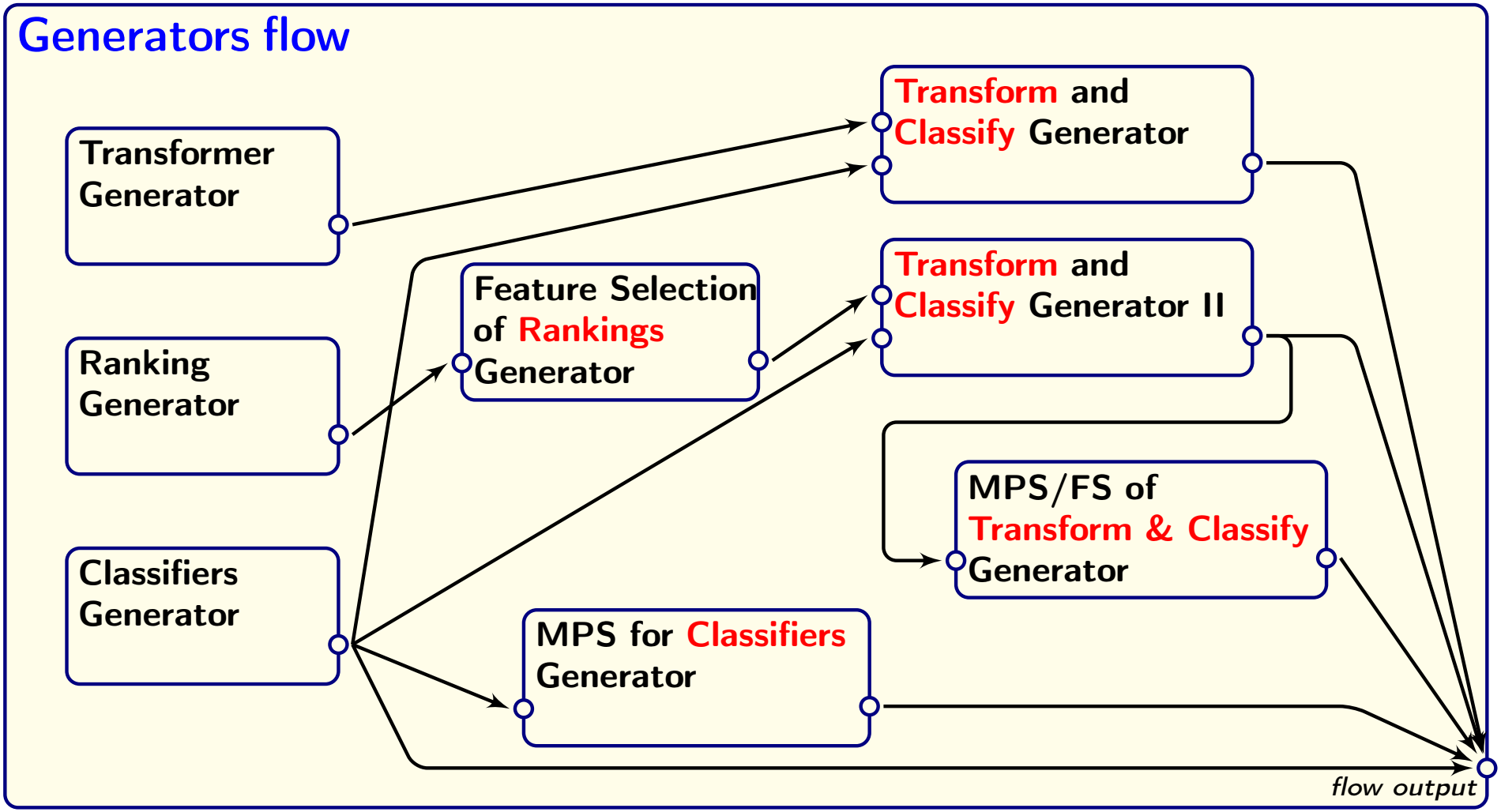




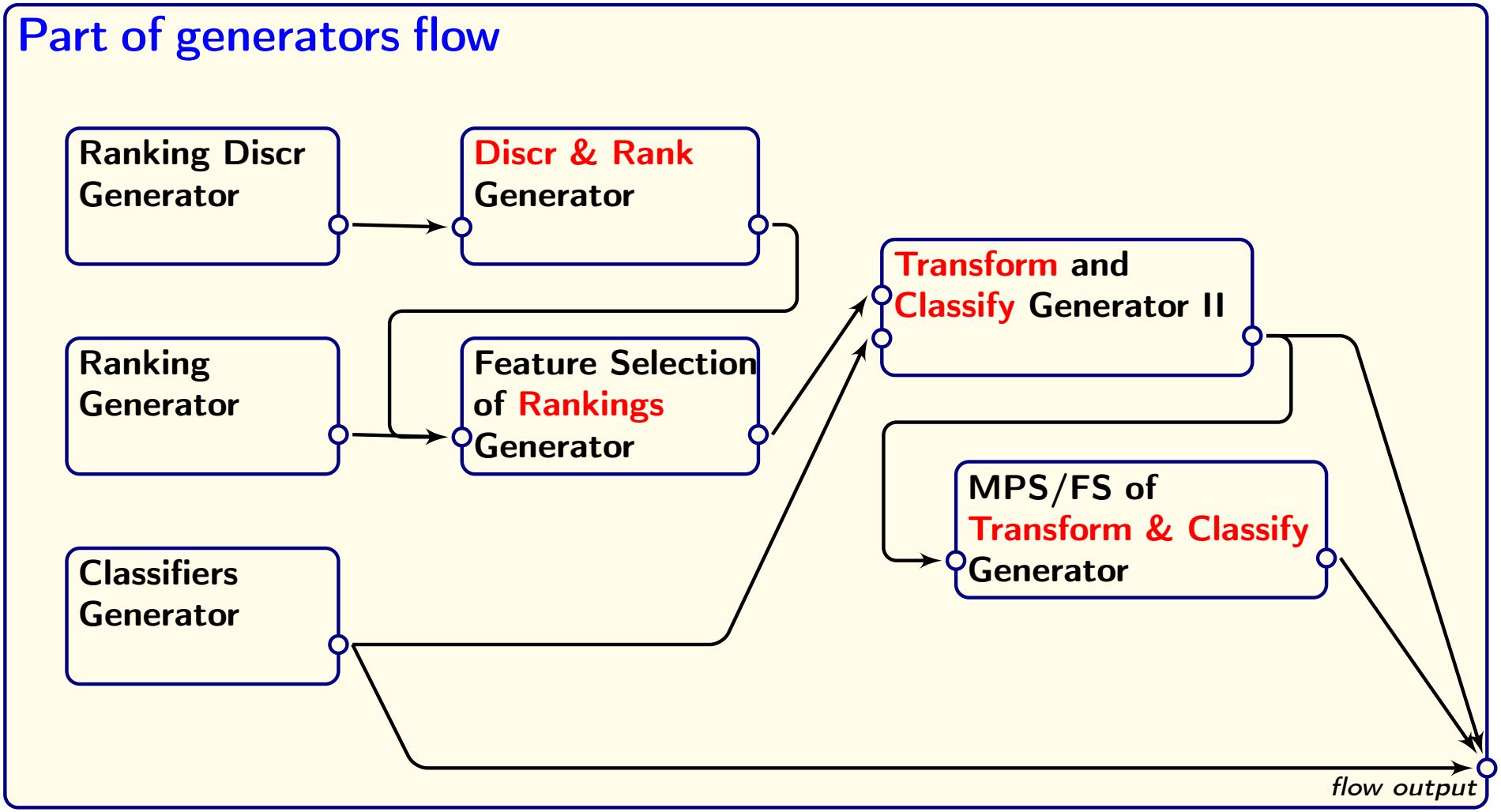
Generator



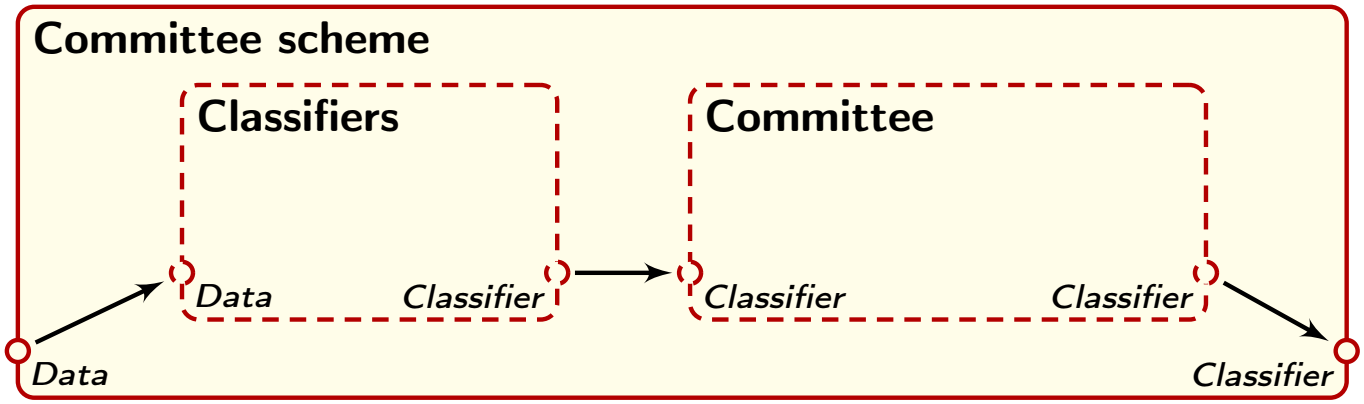
Generator flow example

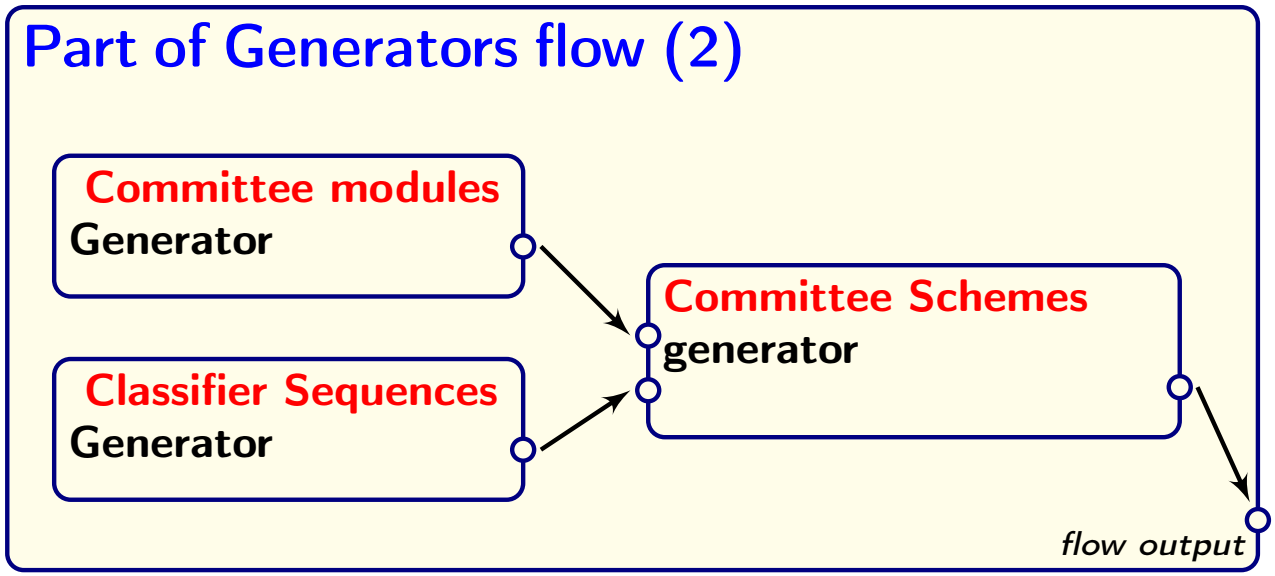


A part of generator flow (1) ...



Schemes of a committee & generator flow with committee

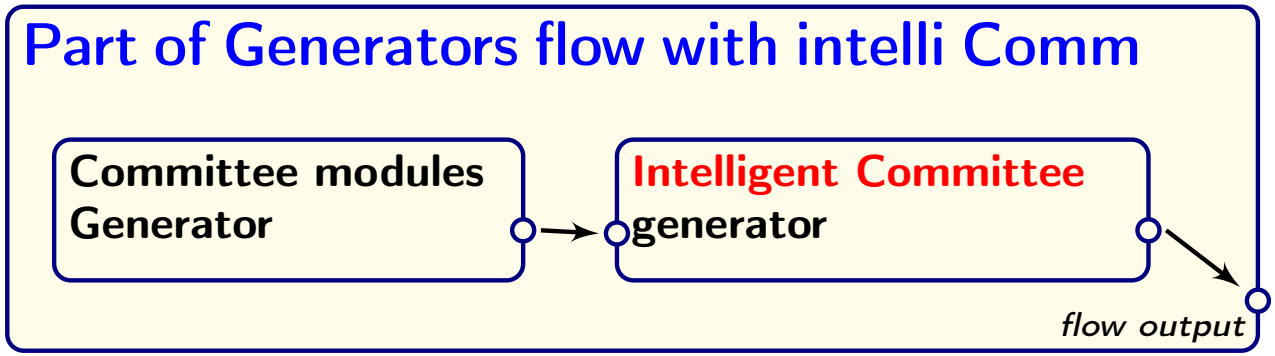




Advanced Generators

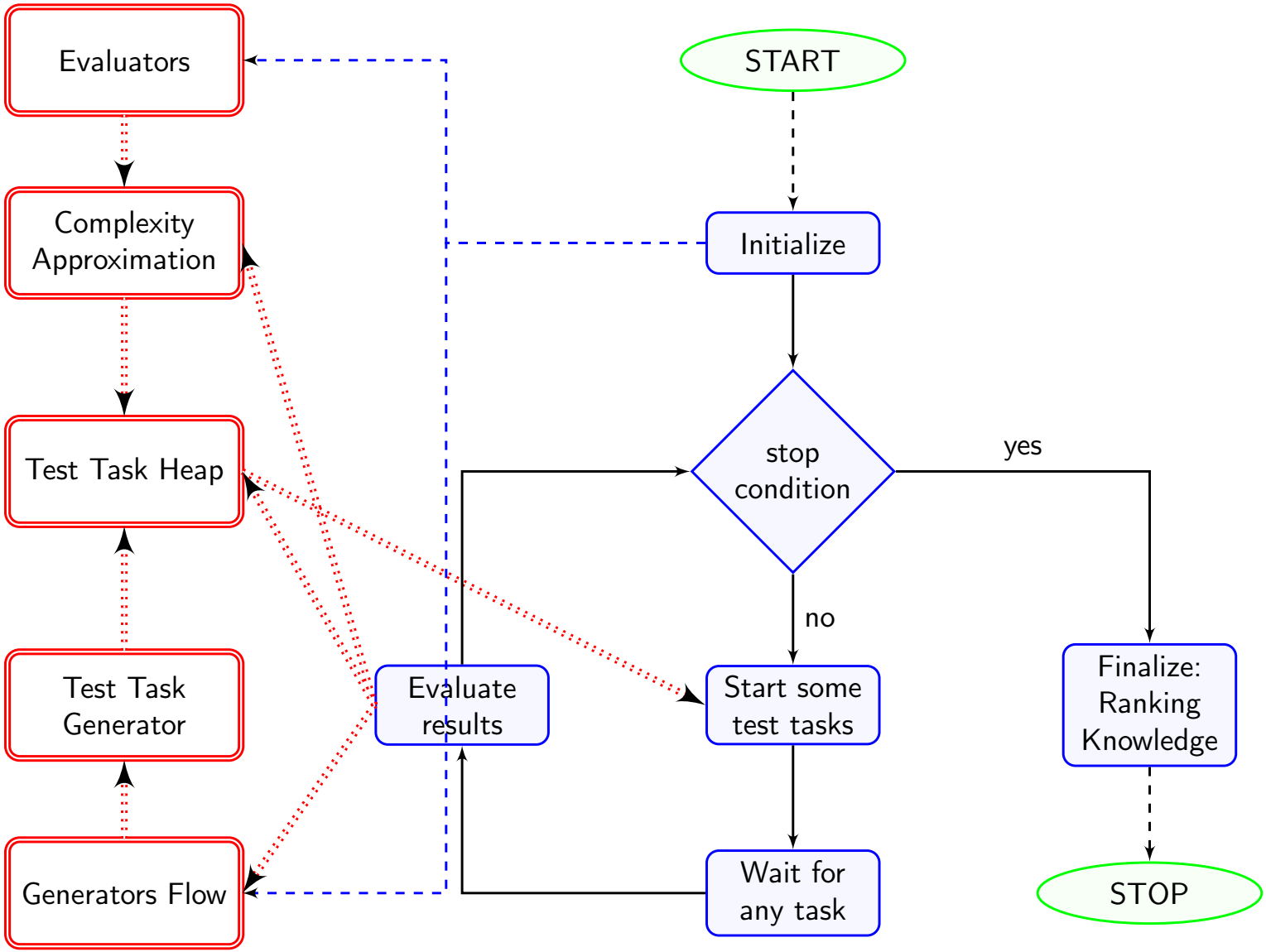
- **Advanced generators are informed each time a test task is finished**. The generators may read the results of test task and the current machine configuration ranking.
- **Advanced generators can learn from meta-knowledge** and build their own specialized meta-knowledge.
- **A meta-knowledge is used by generator to provide new machine configurations.**
- **Different generators $\equiv$ different *strategies***. No single perfect strategy.

Generation of intelligent committees



- Intelligent committee generator observe progress of meta-learning.
- Natural Model diversity (!!!) — generators observe which part of data is badly classified/approximated.
- Committees are constructed from machines of different structural complexities.
- McNemmar test — to control diversity.
- Intemi machine cache — almost no additional costs...

Meta-learning loop, complexity and test task ordering



Test task ordering. Which complexity?

$$C_L(P) = \min_p \{c_*(p) : \text{program } p \text{ prints } P \text{ in time } t_p \text{ and halts}\} \quad (8)$$

Solomonoff and **Kolmogorov**

$$c_k(p) = l(p) \quad (9)$$

no time restriction...

Levin

$$c_l(p) = l(p) + \log t_p \quad (10)$$

The problem of using above def — *too small time influence...*
Ex.: a program running **1024 times longer** than another one may have just a little bigger complexity: **just** $+10$.

Modifications of complexity measures and their optimality

- Let's consider **Restricted learning problem** with time limit t
- and assume that **each Learning machine is equally promising**
- Let's learning machines $m_1, m_2, \ldots, m_q$ corresponds with testing times of $t_1, t_2, \ldots, t_q$ respectively.
- **Then to maximize** the expected value of the best test result obtained within the time t, we need to run as many tests as possible...
- Therefore, an **optimal order** of the tests is an order of nondecreasing testing times:

$$t_{i_1} \leq t_{i_2} \leq \ldots \leq t_{i_m}, \tag{11}$$

- and the choice of **first m shortest/simplest** tests such that

$$m = \arg \max_{1 \leq k \leq q} \left[\sum_{l=1,\ldots,k} t_{i_l} \leq t \right], \tag{12}$$

is an **optimal choice** of respective learning machines.

- This implies that meta-learning in complexity definition have to base on the time:

$$c_t(p) = t_p \tag{13}$$

This optimal choice assumes that LM are equally promising.

- Negligence of program length in (13) is not recommended because machines always have to ***fit* in the memory**, which is limited too.
- In learning machines, typically the time complexity exceeds significantly the memory complexity. This is why measure of complexity should combine time and memory together:

$$c_a(p) = l_p + t_p/\log t_p. \tag{14}$$

CONCLUSION: Test tasks are started with TIME LIMIT

The quarantine and penalty term

- Complexity Approximation Framework $\Rightarrow$ errors.
- If **test task broke** the time limit, the **reliability decrease** and test task come back **to the machine heap**.
- **Quarantine + penalty term:**

$$c_b(p) = [l(p) + t_p / \log t_p] / q(p) \tag{15}$$

  where $q(p)$ reflect **an estimate of reliability** of p.
- **No halting problem** inside tested machines.
- This mechanism **prevents** from running test tasks for **unpredictably long time**.

Attractiveness

- **NO more assumption of equally promising machines**

- Another extension of the complexity measure is possible thanks to the fact that MLAs are able to collect meta-knowledge during learning.
- The **meta-knowledge** may **influence the order of test tasks** waiting in the machine heap and machine configurations which will be provided during the process.
- The optimal way of doing this, is adding a new term to the $c_b(p)$ to shift the start time of given test in appropriate direction:

$$c_c(p) = [l(p) + t_p / \log(t_p)] / [q(p) \cdot a(p)]. \quad (16)$$

$a(p)$ **reflects the attractiveness** of the test task p.

Computing the complexity — Meta-evaluators

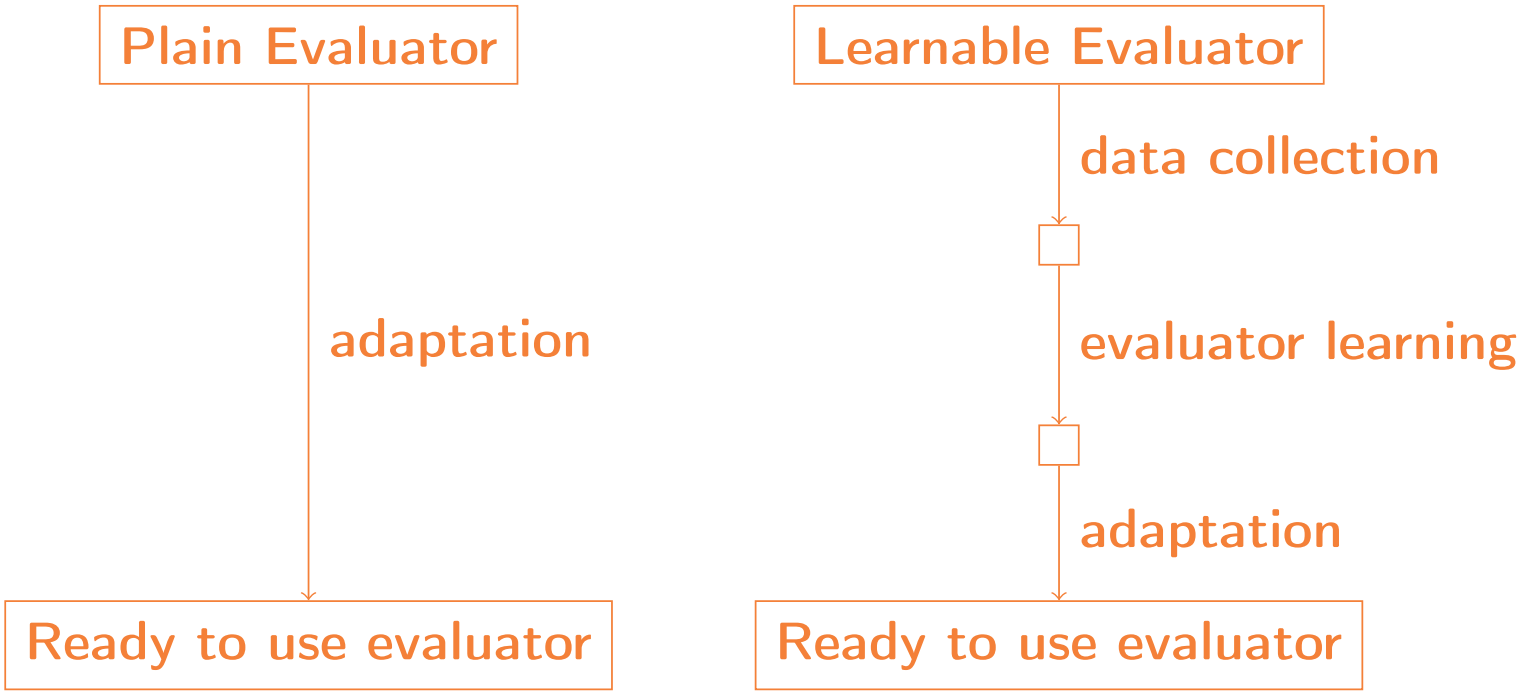
- A function computing the complexity for machine $\mathcal{L}$ should be a transformation:

$$\mathcal{D}_{\mathcal{L}} \ : \ \mathcal{K}_{\mathcal{L}} \times \mathcal{M}_{+} \rightarrow R^2 \times \mathcal{M}_{+}, \tag{17}$$

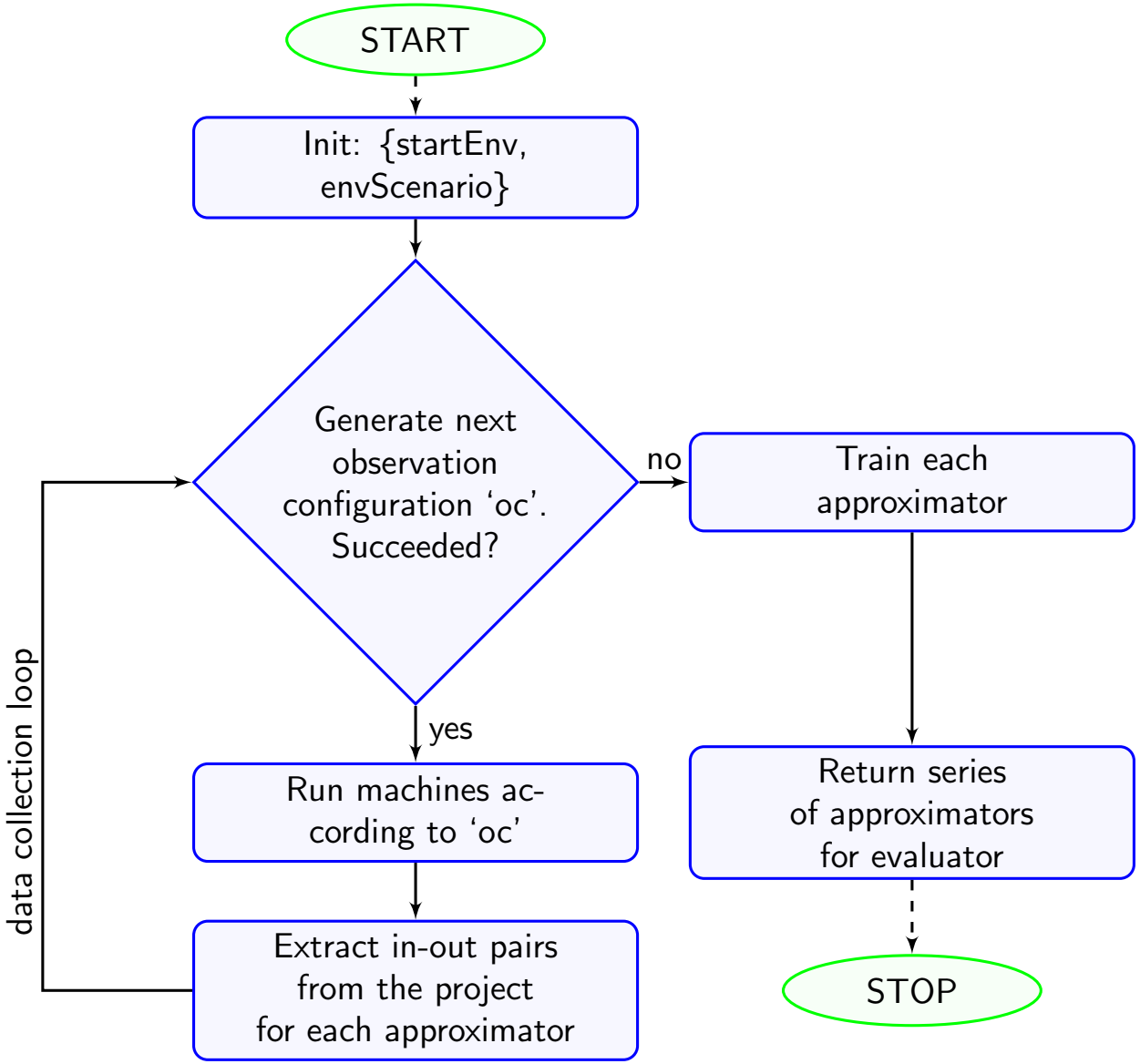
  where $\mathcal{K}_{\mathcal{L}}$ is the configurations space and $\mathcal{M}_{+}$ is the space of meta-inputs (and outputs). R^2 reflects the time complexity and the memory complexity.
- The problem is not as easy as the above form of the function. Finding the right function for given learning machine $\mathcal{L}$ may be impossible.
- **Configuration elements are not always as simple as scalar values.**
- In some cases configuration **elements are represented by functions or by subconfigurations**.

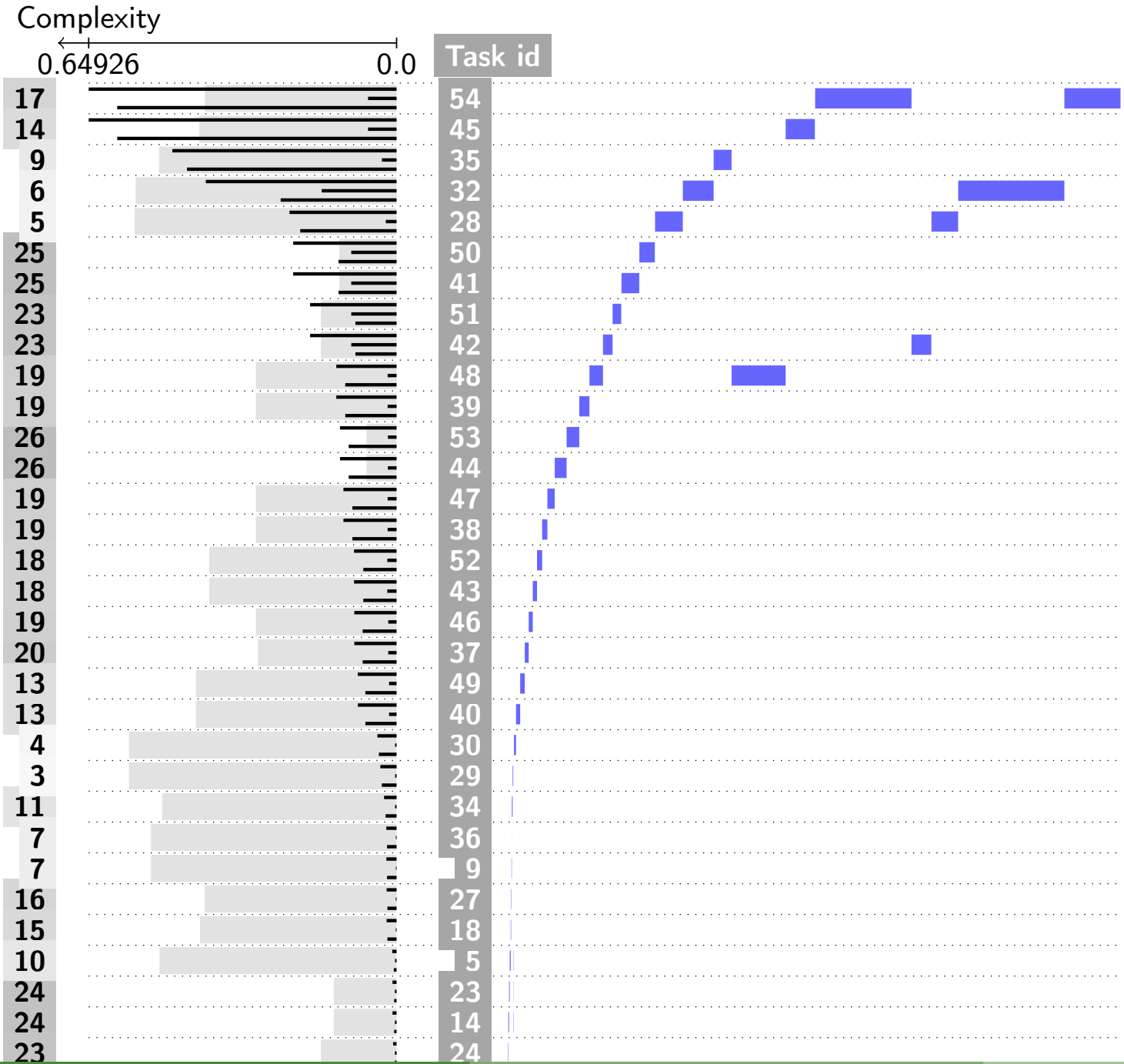
Evaluators Approximation framework

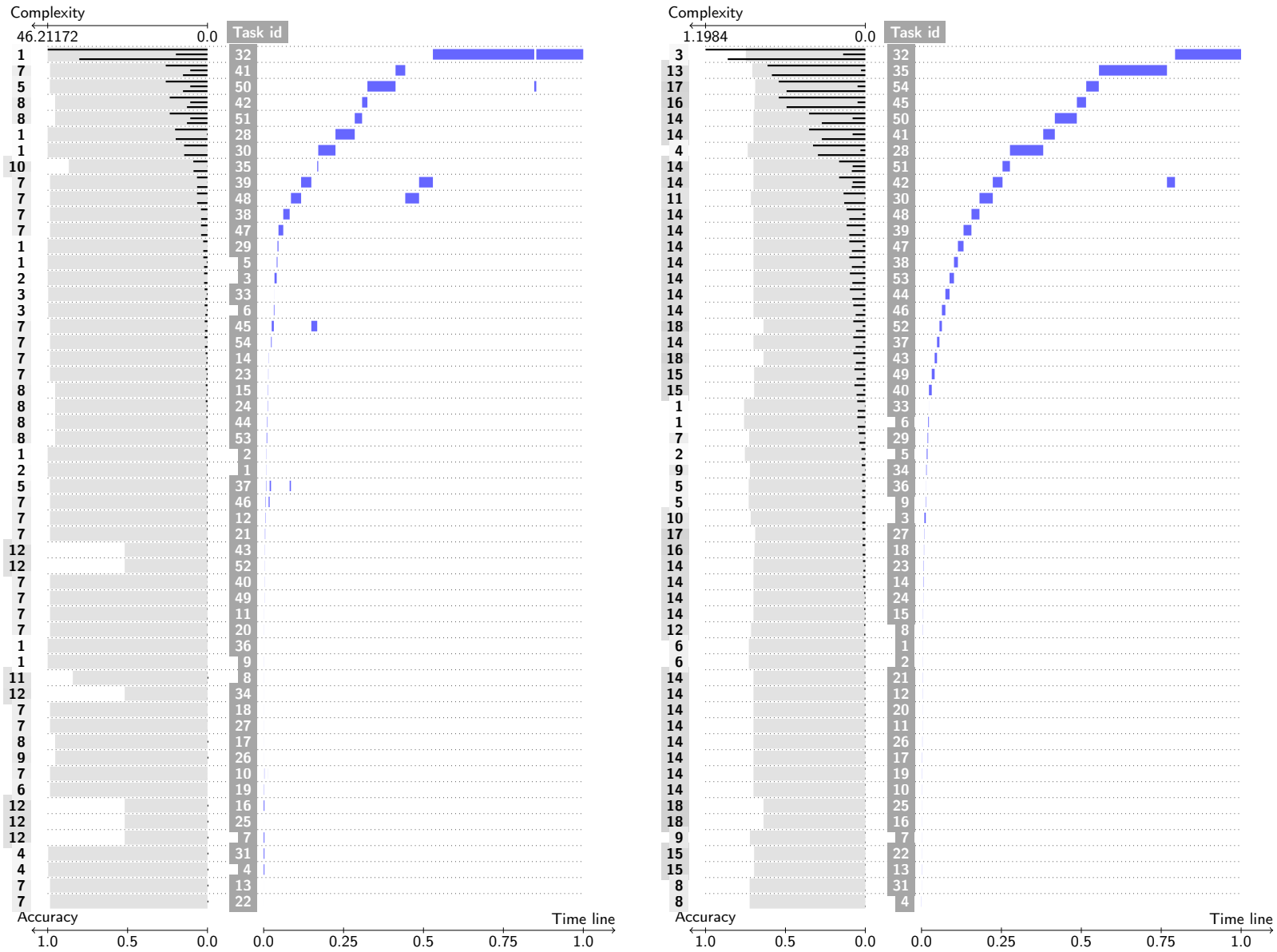
- If evaluators may not be defined in analytical way the **approximation framework** is used to construct evaluators.



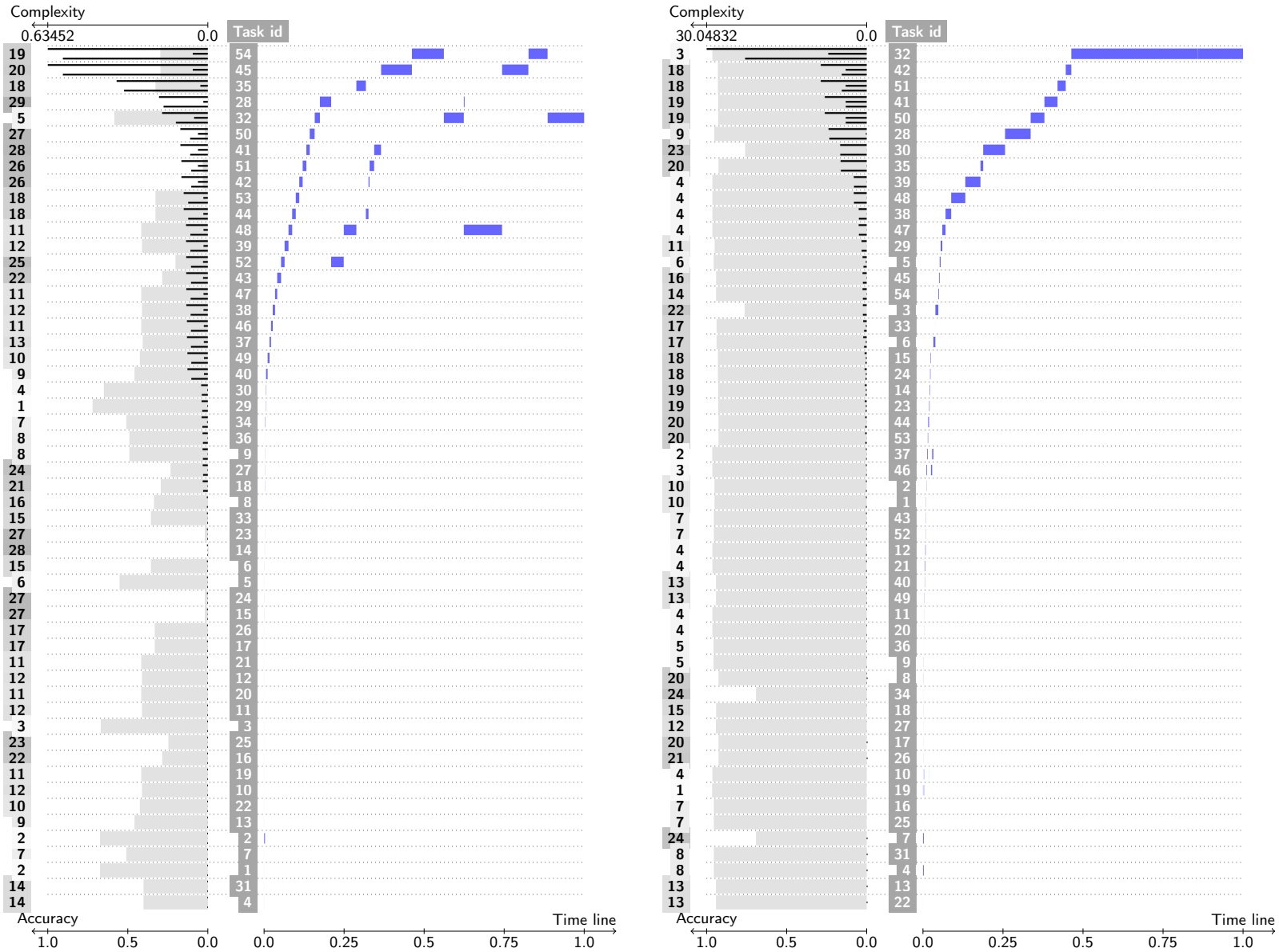
- Before an evaluator is used by a meta-learning process, all its approximators must be trained.
- Each evaluator is learned once.


START
Init: {startEnv, envScenario}
Generate next observation configuration 'oc'. Succeeded?
no
Train each approximator
yes
Run machines according to 'oc'
Extract in-out pairs from the project for each approximator
data collection loop
Return series of approximators for evaluator
STOP


Complexity
0.64926
0.0
Task id
17 54
14 45
9 35
6 32
5 28
25 50
25 41
23 51
23 42
19 48
19 39
26 53
26 44
19 47
19 38
18 52
18 43
19 46
20 37
13 49
13 40
4 30
3 29
11 34
7 36
7 9
16 27
15 18
10 5
24 23
24 14
23 24

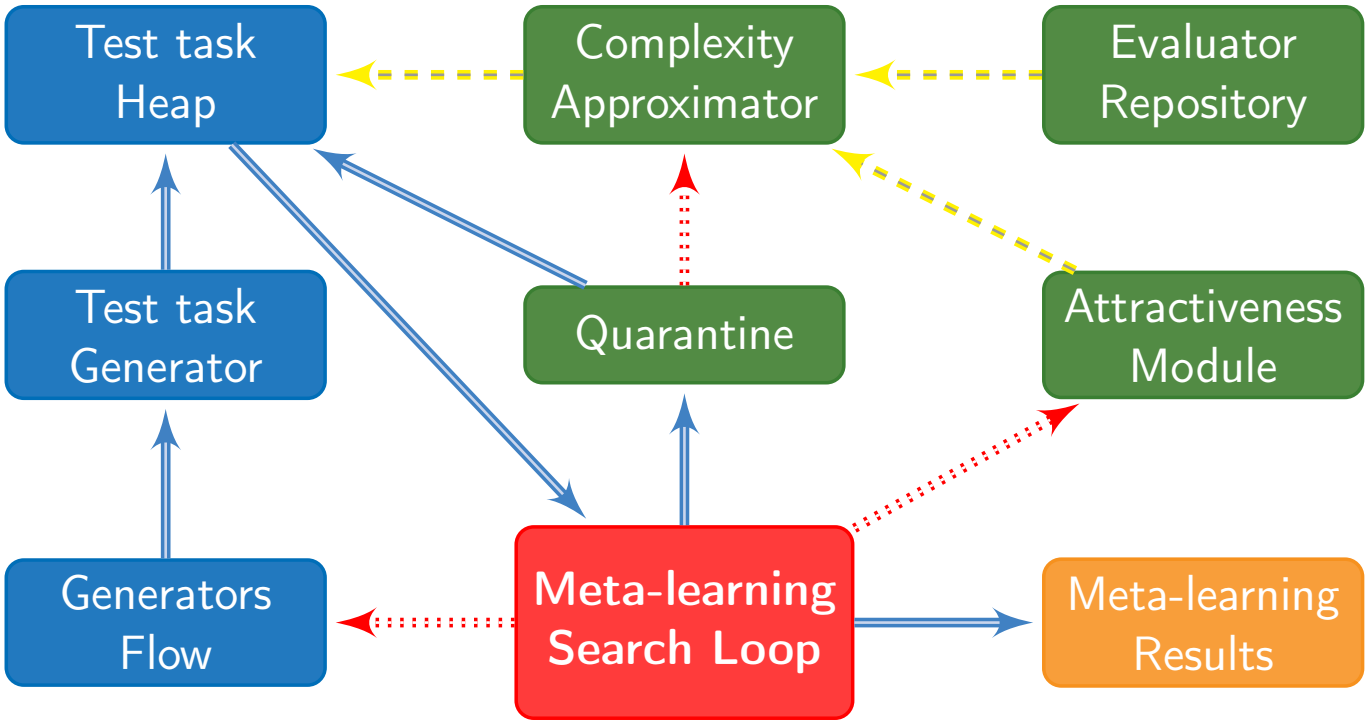


Mushroom & German numeric

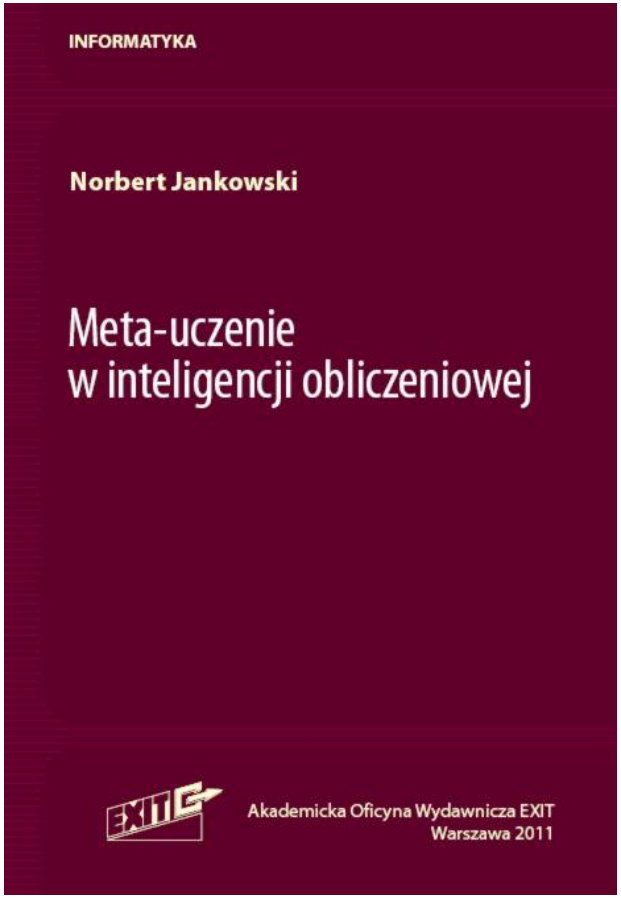


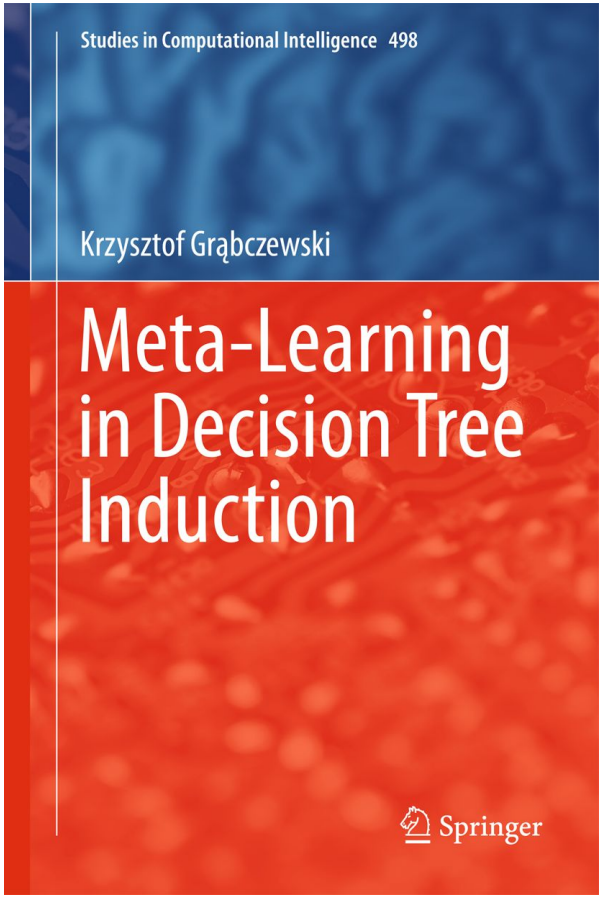
Glass & Thyroid

Summary


Test task Heap
Complexity Approximator
Evaluator Repository
Test task Generator
Quarantine
Attractiveness Module
Generators Flow
Meta-learning Search Loop
Meta-learning Results


Studies in Computational Intelligence 358
Norbert Jankowski
Włodzisław Duch
Krzysztof Grąbczewski (Eds.)
Meta-Learning in Computational Intelligence
Springer


INFORMATYKA
Norbert Jankowski
Meta-uczenie w inteligencji obliczeniowej
Akademicka Oficyna Wydawnicza EXIT
Warszawa 2011


Studies in Computational Intelligence 498
Krzysztof Grąbczewski
Meta-Learning in Decision Tree Induction
Springer