Evolving Decision Strategies for Computational Intelligence Agents

Martin Šlapák

Department of Theoretical Computer Science
FIT CTU in Prague, Czech Republic
slapamar@fit.cvut.cz

13. 3. 2014

What is an agent?

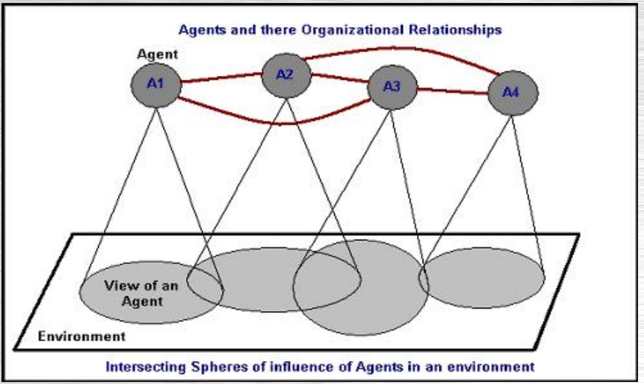
A formal definition by M. Wooldrige

An agent is a computer system that is situated in some environment, and that is capable of autonomous action in this environment in order to meet its design objectives.

- Main benefits of agents:
  - encapsulation (local data, knowledge, ...)
  - persistence
  - autonomy
- Agent's inteligence is usually hardwired.
- An agent can have own "inteligence", it can evaluate some behaviors.
- My work is focused on adaptive approach.

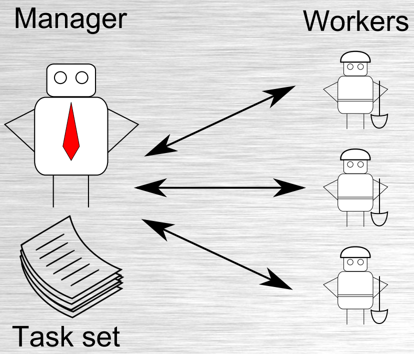
Multi-agent system

- Multiple (intelligent) agents interacting within an environment.
- Cooperative vs. competitive.
- Many other points of view: autonomy, decentralization, local information, type of environment (phys./virtual, discrete, continuous)



Problem definition and main objectives

- Multi-agent system for machine learning.
- Distributed solving of given task set.
- Optimize some criteria (mean error, time, ...) for one task.
- Manager is "dumb" (send tasks randomly).
- Workers have to determine (localy) whether accept or reject the offered task in order to fit criterium.



Agent types

- Manager agent's behaviours:
  - task sending (an offer to worker)
  - receiving back solved or rejected tasks
  - communicating with experimental enviroment (sending results back to evolution – fitness counting)
- Worker agent's behaviours:
  - **decision making** (about task accepting)
  - task solving
- Type of datamining models in workers:
  - **Radial–basis function** (RBF)
  - **Naive bayes**
  - **Multilayer perceptron** (MLP)

Tasks

- Classification
- Tasks come from UCI Machine learning repository[1]
- We selected: **car**, **breast-cancer**, **iris**, **lung-cancer**, **tic-tac-toe** and **weather**
- Number of tasks:
  - 30 for evolution of decision making systems
  - 300+ for performance test of the best evolved DM systems
- Task metadata:
  - # of attributes
  - # of classes
  - # of instances
  - *used in prediction of results (explained later)*

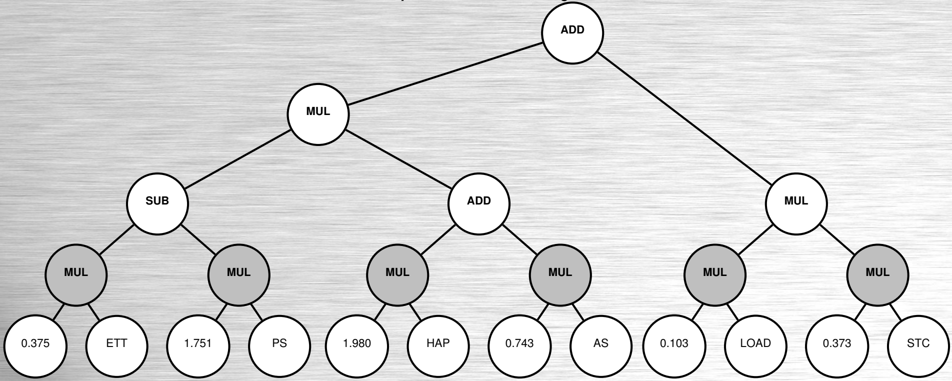
[1] Available at `http://archive.ics.uci.edu/ml/datasets.html`

Problem's criteria

- mean error – average error of the whole task set
- time – average time to solve one task
  - real time – seconds
  - virtual time – ticks (steps)
  - aspects of worker's buffer
- multicriteria – mean error, virtual time

Decision making

- Polynomial (tree structure) connects together all local attributes of an agent and tasks.
- **How to make a decision?**
  1. Substitute all variables in a tree by actual values of attributes.
  2. Evaluate tree as polynomial and obtaing one real number R.
  3. **If** $R > 0$ **then** *accept task* **else** *reject task*.

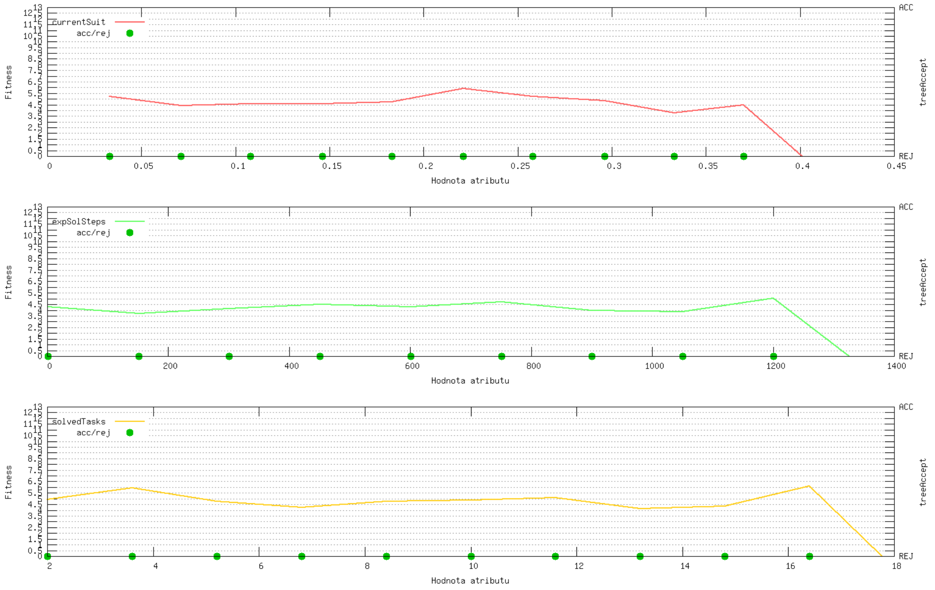


Decision making – attributes

The tree connects together **N** local **attributes of an agent**:

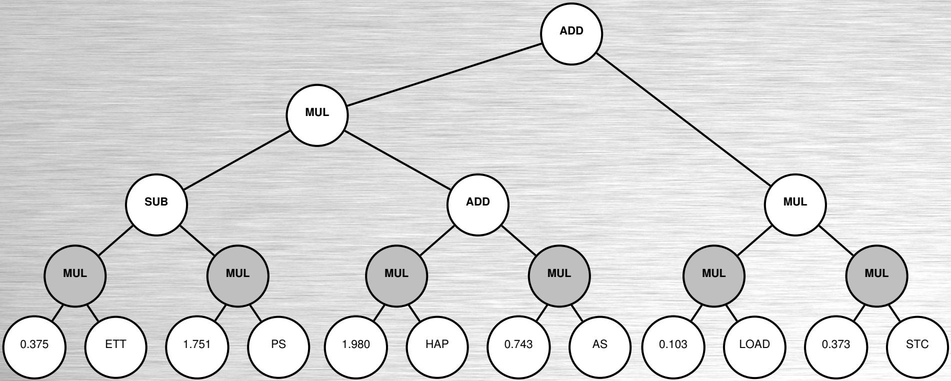
- **solvedTasks** is number of tasks solved by an agent,
- **expSolSteps** is expected time (steps) to solve actual task,
- **stepsSolved** is number of steps of actual task ever done,
- **currentSuit** represents agent–task compatibility of actual task,
- [deprecated] **avgTaskTime** is average expected solving time per task in agent's task buffer,
- [deprecated] **avgBuffSuit** which means an average compatibility of enqueued tasks in agent's buffer.
- [new] **offeredSuit** represents agent–task compatibility of new offered task,
- [new] **percentSSol** is *stepsSolved* in percentage,
- [new] **ticksToEnd** represents how many ticks left to solve actual task

Decision making – sensitivity analysis



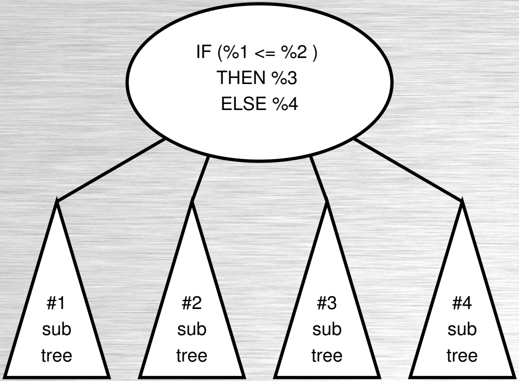
Decision making – combination of attributes by a polynomial

- inner nodes: **ADD**, **SUB**, **MUL**
- leaf nodes: *double* constant, *double* / *integer* attribute



Decision making – from polynomial to general tree structures

- inner nodes: new ternary operator **IF**
- possibility to make several "smaller" decisions based only on some subset of attributes



Decision making – general tree structures (*IF* operator)

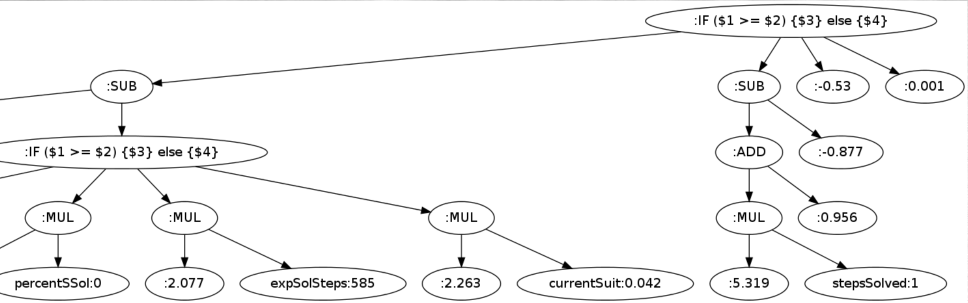


Figure: Example of an evolved tree with *IF* operator

Evolution (Genetic Programming)

- Common genetic algorithm is used.
- Evolved for 200–500 generations.
- Tournament selection (tournament size was up to 20 %).
- Size of population is only about 20–30 individuals.
- Individual is polynomial structure represented as **N-ary tree**.
  - encoding: natural $\sim$ Java objects
- Calculation of the fitness function is extremely expensive (time ascpect).
  - Run whole experiment for given individual.
  - Use of precomputed results.
- Island model for parallel evolution.
- Elitism is used.

Evolution (GP) – mutation operators

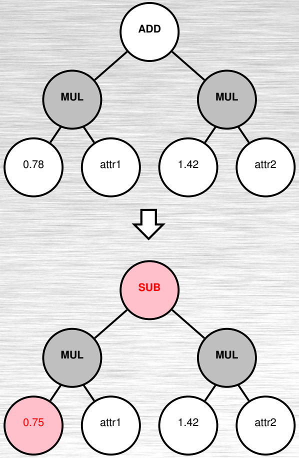
- Probability of mutation 10 %.
- Inner nodes: **change node operation**
- Leaf nodes: **change constant**
  1. Add δ to the leaf value.
  2. Change slightly this δ:

$$\delta_{T+1} = \delta_T \cdot \left(1.1 - \frac{\mathrm{rnd}()}{5}\right)$$

  - initialization of δ:

$$\delta = (-1)^r \cdot \frac{1}{3} v$$

where r is chosen at random from set $\{0, 1\}$ and v is initial δ value.



Evolution (GP) – cross operator

- Swap randomly selected distinct subtrees.
- How to deal with **bloating problem**?
  - Ignore it! :-)
  - Omit crossover.
  - Prevent it by generating only valid trees by some grammar. ∼ Kitano
  - Swap *similar* subtrees. ∼ my approach
- Randomly select 1st individual and one of its subtrees, randomly select 2nd individual; for 2nd one generate all possible subtrees and select from them.
- **Similarity metric** – number of identical attributes in leaves of two candidate trees.

Fitness

1. Fitness of individual is **total experiment time** combined with **AVG task time**.

$$f(\text{individual}) = 2 \cdot \frac{R_{experiment}}{T_{experiment}} + \frac{R_{avgTask}}{T_{avgTask}}$$

   R_x ... empiric values obtained by 100 times run experiment with random decision making.

2. Fitness of individual is averaged **mean error** over the whole task set.

$$f_E(\text{individual}) = \frac{\sum_{t=1}^{\text{taskCnt}} E_t}{\text{taskCnt}}$$

3. Fitness of individual is averaged **time** of one task.

$$f_T(\text{individual}) = \frac{\sum_{t=1}^{\text{taskCnt}} T_t}{\text{taskCnt}}$$

4.

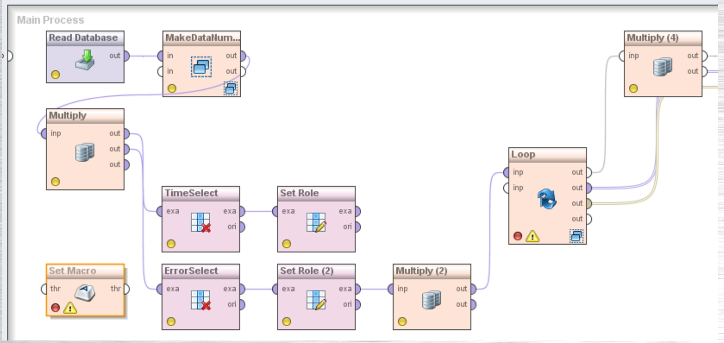
$$f_c(\text{individual}) = \alpha \cdot \frac{1}{f_E} + \beta \cdot \frac{1}{f_T}$$

Precomputed database of results

- Each fitness evaluation means to run the whole experiment.
  - Extremely time expensive.
  - Real time $\Rightarrow$ virtual time $\Rightarrow$ in fact no solving of classifying tasks, using precomputed results (Pikater project).
- 18 combinations of pair *agent–task*
- $\approx$ 105k rows
- Used to obtain:
  - some attributes – eg. an expected number of *ticks/steps* to solve an offered task.
  - final error of the solved task.

Model learning for parameter estimation

- Based on precomputed DB, why not to learn regression models for predicting that attributes?
- **Actual work – not yet tested!** ;-)
- Preliminary: KNN (k=5) seems to be the best for parameter estimation of MLP.



Results – random DM vs. evolved polynomial

Table: Average computation times for non–informed solutions with fixed task accept ratio and decision making based on evolved tree.

decision method	⌀time per task [ms]	time for task set [ms]
accept 2% of tasks	1 004	99 432
accept 10% of tasks	8 593	882 961
accept 50% of tasks	12 450	1 458 572
accept 90% of tasks	16 299	1 648 996
accept 100% of tasks	16 453	1 691 372
best expression	1 013	120 636

Results – best trees evolved with or without crossover

Table: Comparison of different decision making approaches

Decision method	Random 50 % accept. ratio	Best tree without crossover	Best tree with crossover
Real acceptance ratio	0.4845	0.4286	1.0000
Avg. task error	0.1733	0.1375	0.0929
Computational time	5248 ms	1749 ms	1895 ms
Avg. value of polynomial	–	26.4870	90.3724

*) All computed by workers with buffer for incoming tasks.

Results – multicriteria optimization, impact of *IF*

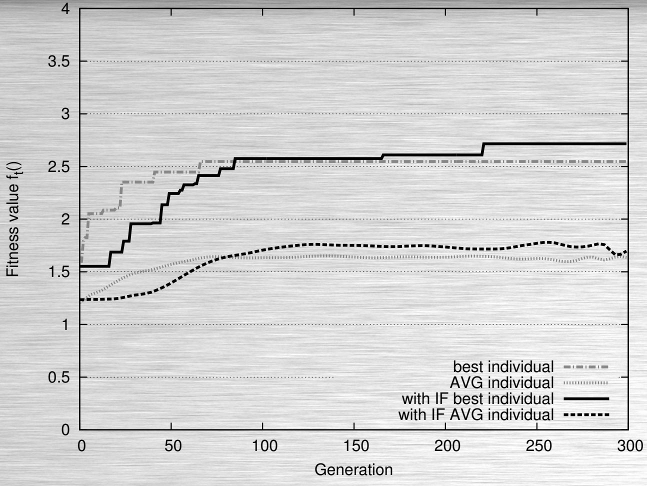


Figure: Impact of adding *IF* operator; combined fitness $f_c = \alpha \cdot \frac{1}{f_E} + \beta \cdot \frac{1}{f_T}$

Results – MCO, SCO, impact of *IF*

Table: Results of experiments with fitness, agent's attrs and tree's ops

old agent's attrs without deprecated[2] attributes

solvedTasks, expSolSteps, stepsSolved, currentSuit

best fitness	ops: MUL, ADD, SUB	ops: MUL, ADD, SUB, IF
$A = 1/normTime$	43.525	36.404
$B = 1/normAvgTaskErr$	2.371	2.219
$[A + B]_{norm}$	1.175	1.521

old agent's attrs without deprecated and with new attributes

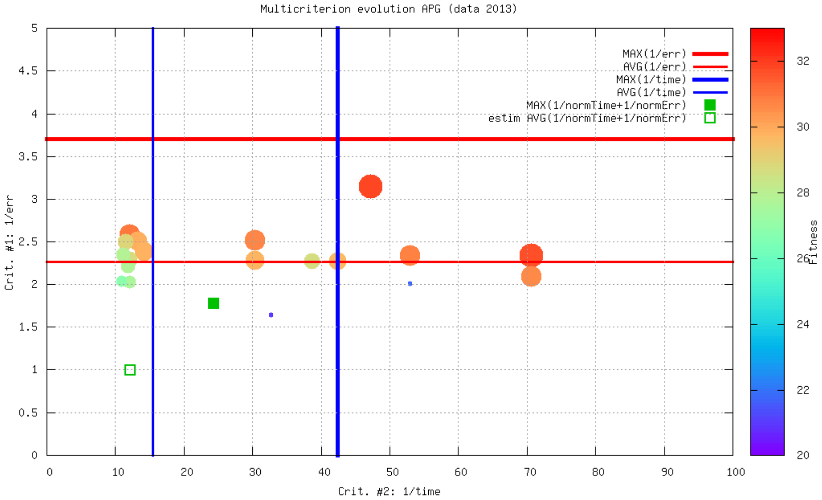
new attributes: *offeredSuit, percentSSol, ticksToEnd*

best fitness	ops: MUL, ADD, SUB	ops: MUL, ADD, SUB, IF
$A = 1/normTime$	48.753	40.583
$B = 1/normAvgTaskErr$	2.185	2.240
$[A + B]_{norm}$	1.635	1.8533

attrs overview

[2] Due to buffer removal: *avgTaskTime, avgBuffSuit*

Results – multicriteria optimization NSGA-II, criterions: f_E, f_T



Results – multicriteria optimization NSGA-II, criterions: f_E, f_T

Table: The summary of the best results of each experiment

experiment	mean squared error	time [ticks]
SCO old attributes	0.3118	1.0120
SCO new attributes	0.3267	1.0117
SCO *if* operator	0.3272	1.0127
MCO all from 1st front	0.3175	1.0215
	0.4273	1.0143
	0.4935	1.0095

Future work & Conclusion

- Replace precomputed DB with trained predicting models.
- Make use of well known general decision trees or any other method (no supervised learning approach – we don't know which combination of agent's attributes is good to ACC/REJ).
- Tune parameters of genetic programming such as impact of crossover etc. (Some ideas from NEAT.)
- Verify the best evolved trees on significantly larger task sets.

Questions?