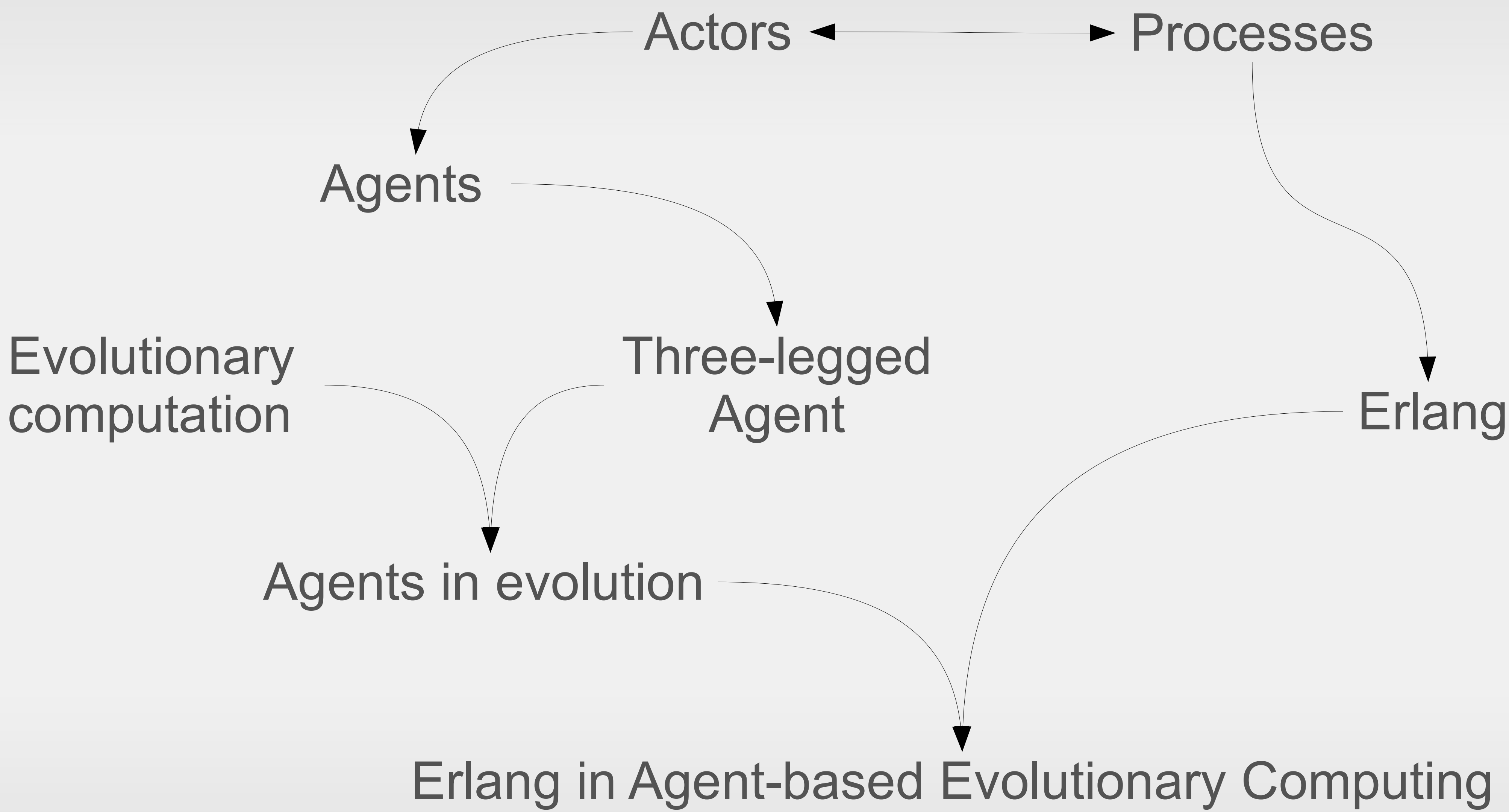
Agent-based Evolutionary Computing

Wojciech Turek

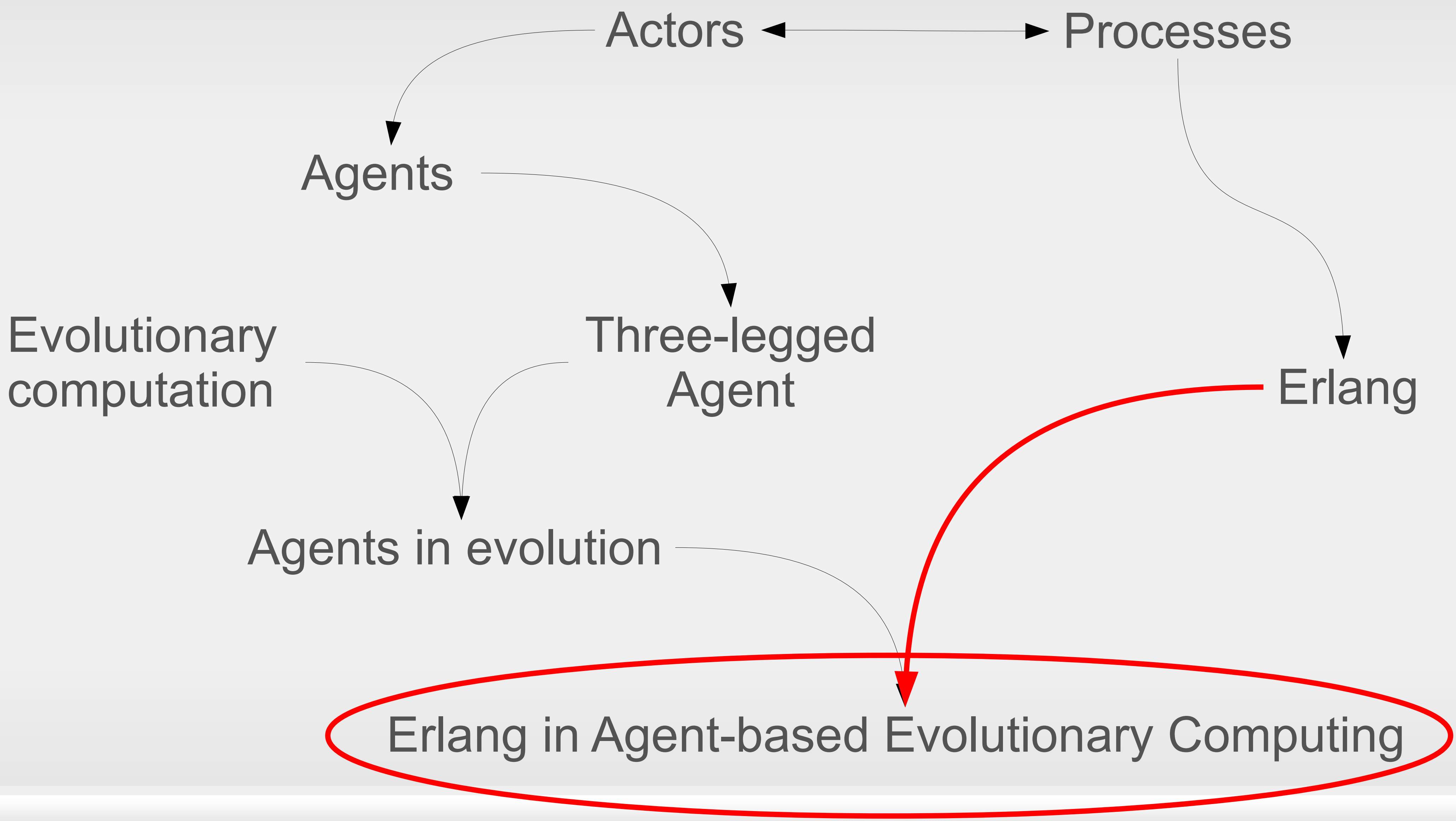
AGH University of Science and Technology
Department of Computer Science
Kraków, Poland

Agenda



Actors
Processes
Agents
Evolutionary computation
Three-legged Agent
Erlang
Agents in evolution
Erlang in Agent-based Evolutionary Computing

Agenda


Actors
Processes
Agents
Evolutionary computation
Three-legged Agent
Erlang
Agents in evolution
Erlang in Agent-based Evolutionary Computing

Actor model

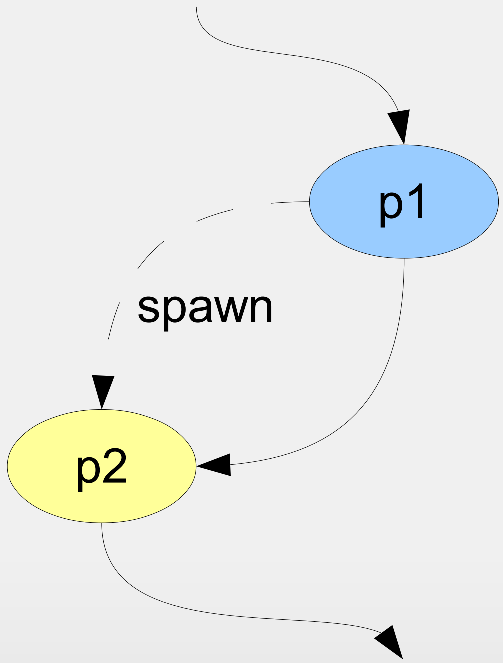
- Formal model of concurrent computation
- Defines an “actor” – a universal primitive, which:
  - Sends messages to other, known actors
  - Receives messages and reacts appropriately
  - Creates more actors

- **Carl Hewitt** et. al. **1973**

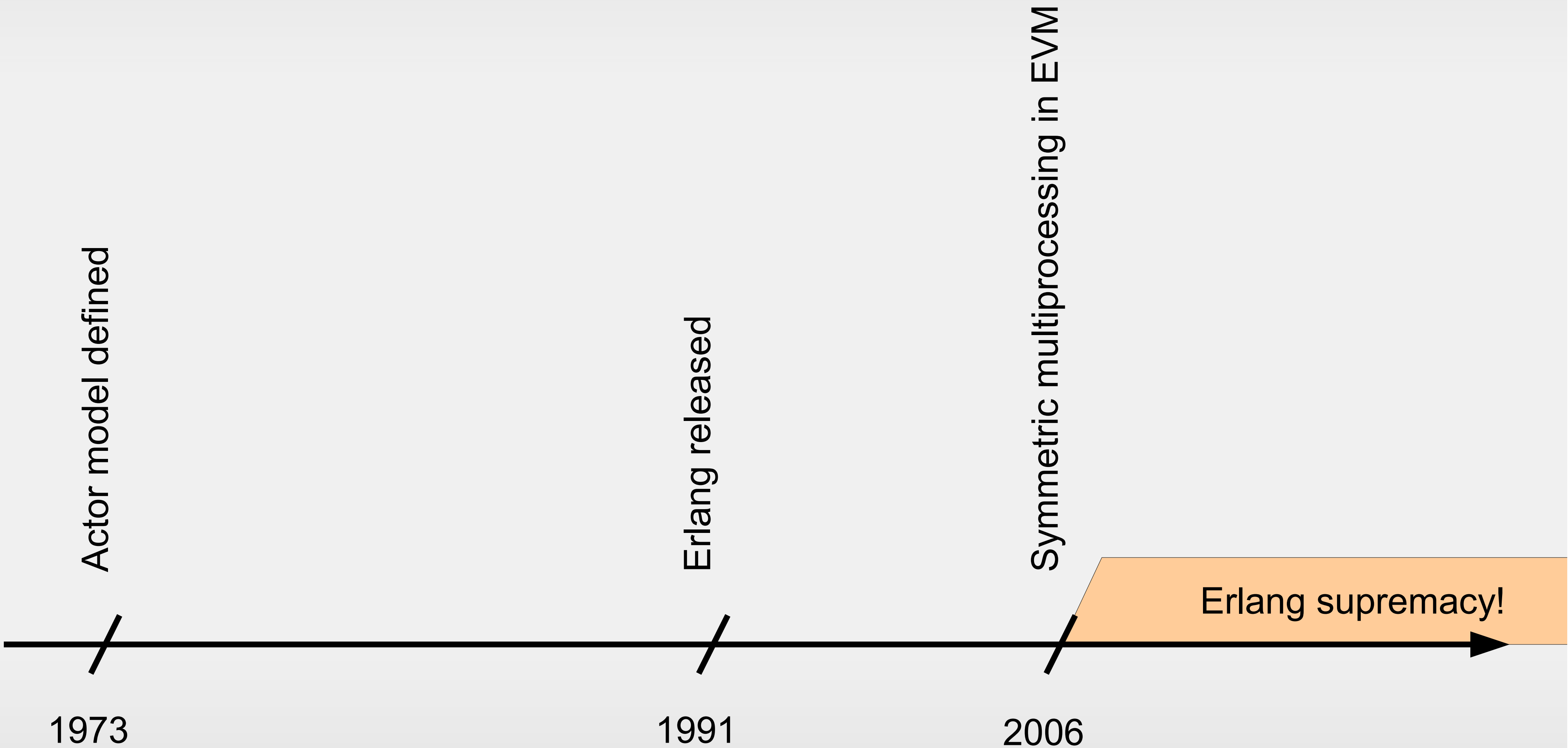
Processes in Erlang

- Sends messages to other, known actors
- Receives messages and reacts appropriately
- Creates more actors

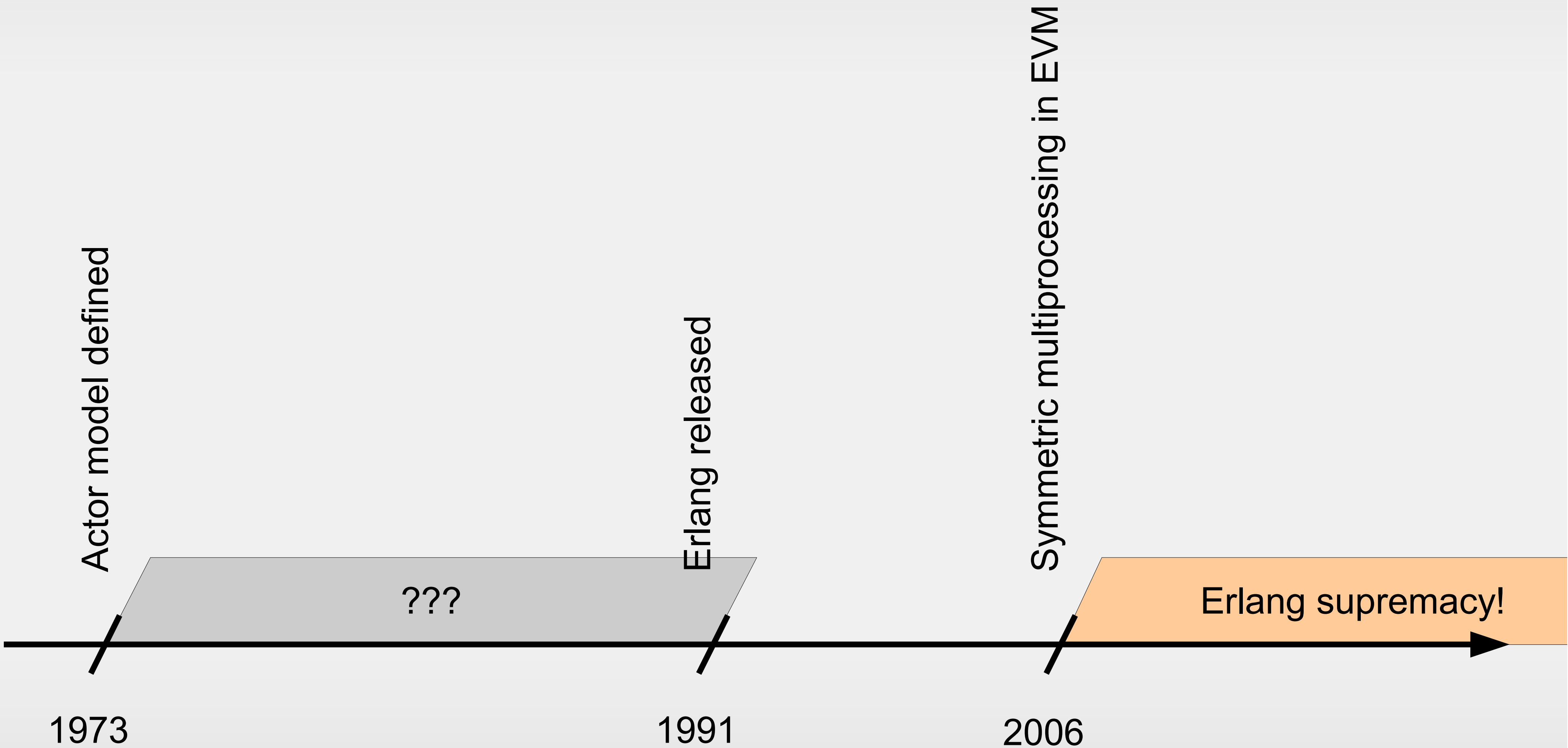
→ Erlang processes do so...



Actors descendants



Actors descendants



Agent model

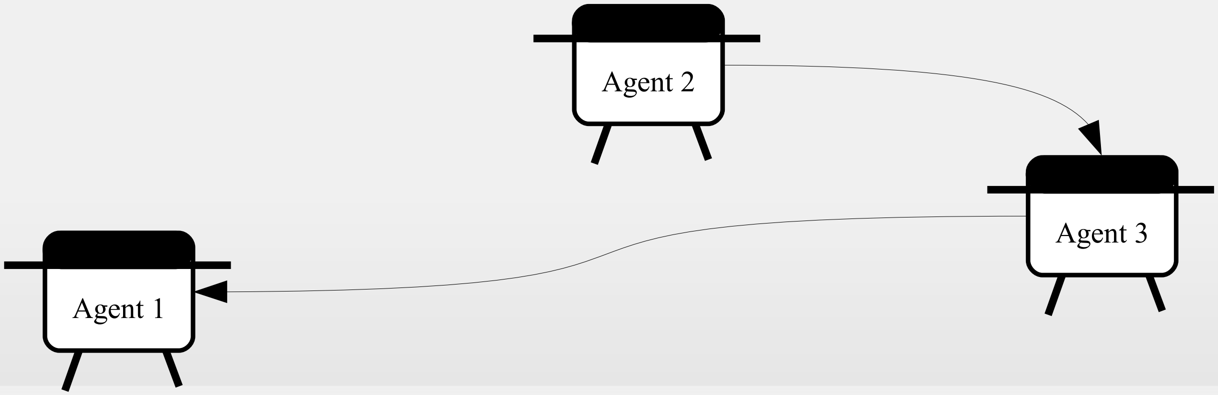
- Computational model for simulating behaviours of autonomous beings
- Defines an “agent” – a universal primitive, which...

> *...is a computer system, **situated** in some environment, that is capable of **flexible** **autonomous** action in order to meet its design objectives.*
>
> *NR Jennings, K Sycara, M Wooldridge, 1998*

- Early development: 1971... 1980

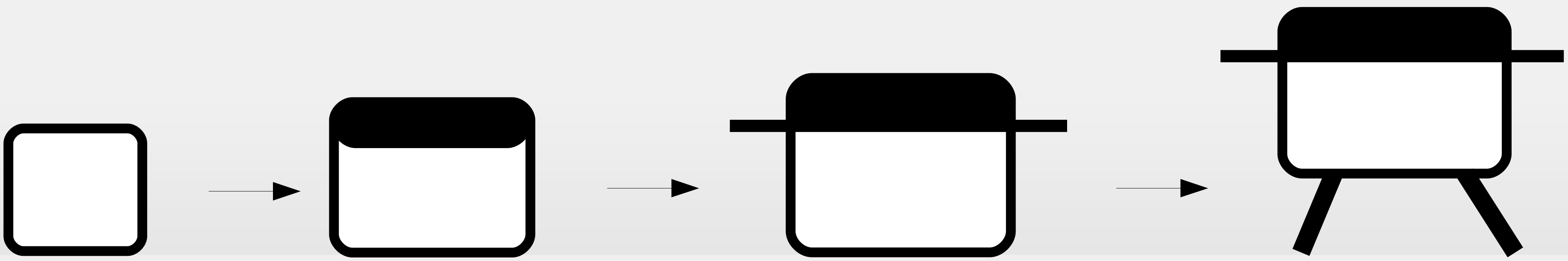
Multi-Agent Systems

- Modelling interactions between autonomous beings:
  - Computation of each agent is asynchronous
  - Data is decentralized
  - No global control system
  - Each agent has its own aim and knowledge
  - Agents can communicate with asynchronous messages



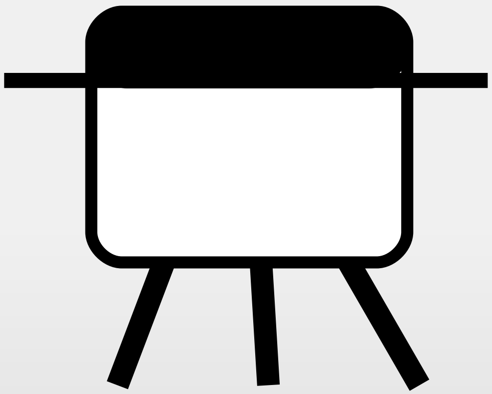
Make it complex

- Many agent definitions, many visions, different aims
  - Asynchronous computation – autonomy
  - Heterogeneous systems – agents able to communicate
  - Agents mobility – code and state migration
  - Knowledge representation, understanding, exchanging
  - Knowledge using
  - Physical agent representation, simulation



Make it complex

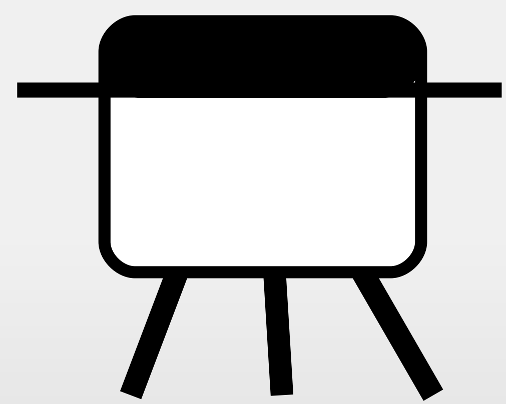
- Three basic views of the agent paradigm:
  - Actor model of computation
  - Heterogeneous systems integration
    - The Foundation for Intelligent Physical Agents
  - Distributed Artificial Intelligence



Make it complex

- Three basic views of the agent paradigm:
  - Actor model of computation
  - Heterogeneous systems integration
    - The Foundation for Intelligent Physical Agents
  - Distributed Artificial Intelligence

This could not walk
very fast or far...



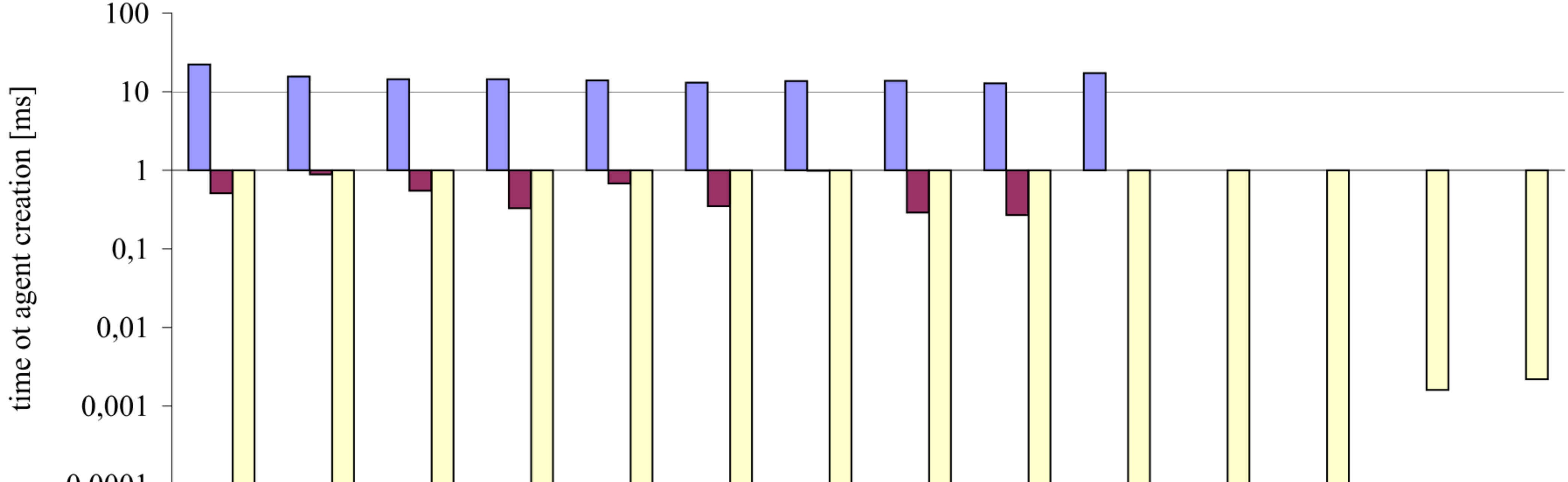
Implementations

- JADE
- Jadex
- Magentix
- Mason
- Repast
- Cougaar
- …

→ mostly Java
→ hundreds of agents

Basic performance

- Agent creation time vs Erlang process creation time

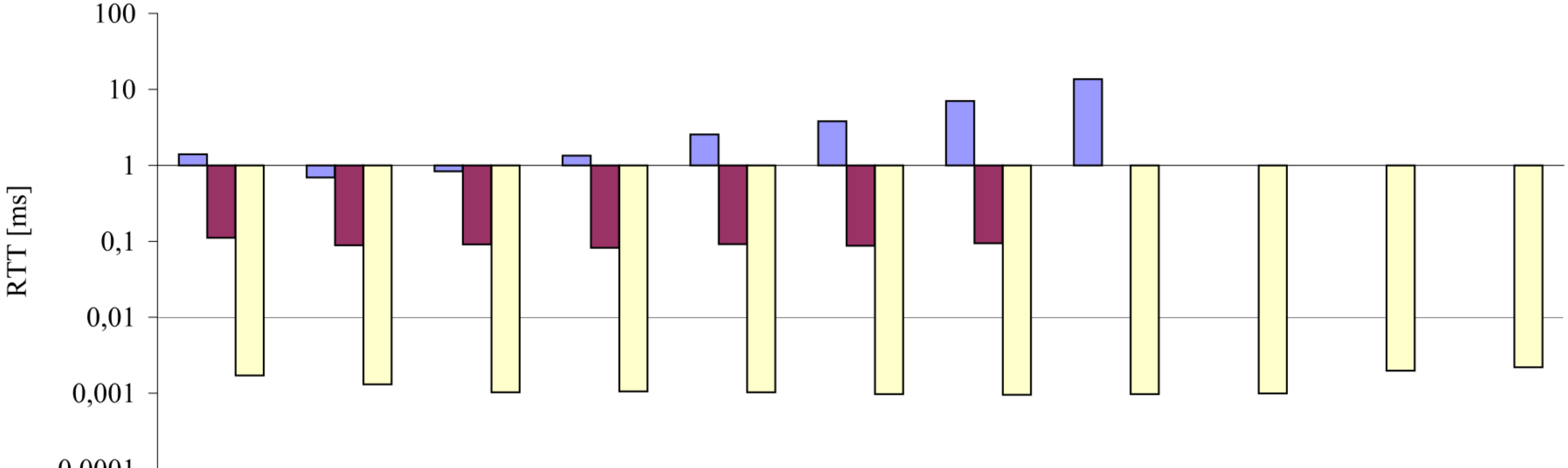


	0	100	200	300	400	500	600	700	800	900	1000	10000	1E+05	1E+06
Magentix2	22,19	15,61	14,43	14,44	13,94	13,03	13,71	13,72	12,76	17,23				
JADE	0,51	0,89	0,55	0,33	0,68	0,35	0,99	0,29	0,27					
Erlang	0,0001	0,0001	0,0001	0,0001	0,0001	0,0001	0,0001	0,0001	0,0001	0,0001	0,0001	0,0001	0,0016	0,0022

number of agents already in the system

Basic performance

- Agent messaging vs Erlang messaging – single node

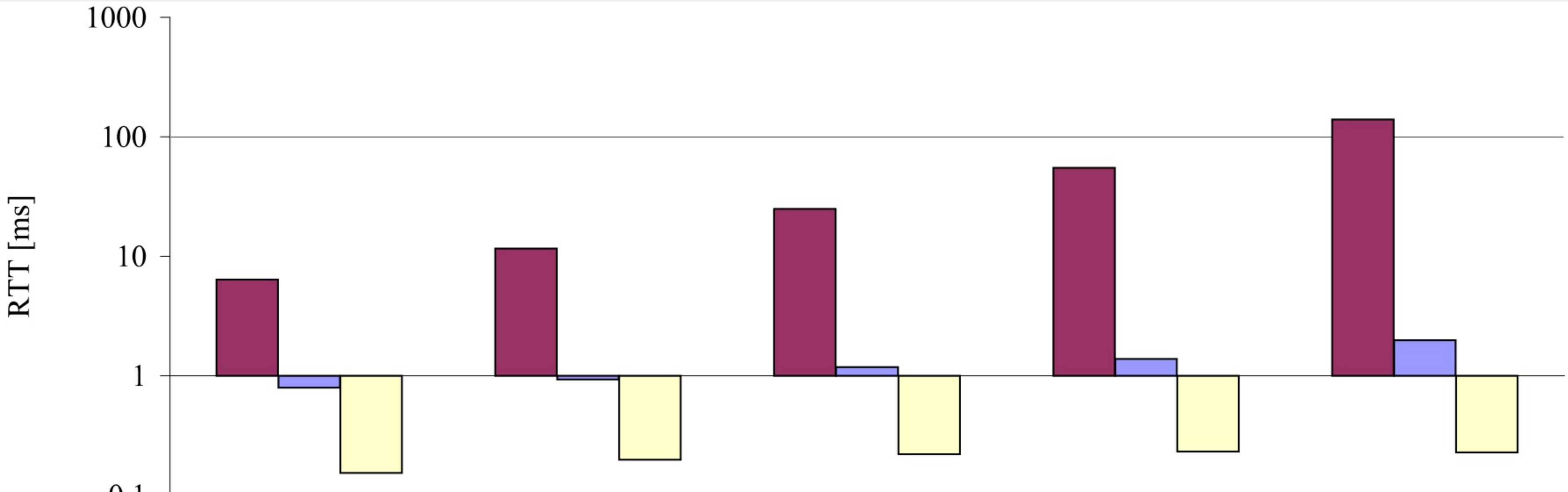


	2	20	100	200	400	600	800	1000	2000	20000	2E+05
Magentix2	1,395	0,6904	0,83304	1,34112	2,5488	3,81372	7,0236	13,62496			
JADE	0,11166	0,088792	0,090812	0,08235	0,09192	0,08722	0,09432				
Erlang	0,001716	0,00131	0,00103	0,001061	0,00103	0,000977	0,000951	0,000973	0,000992	0,00198	0,002204

number of agents

Basic performance

- Agent messaging vs Erlang messaging – multiple nodes



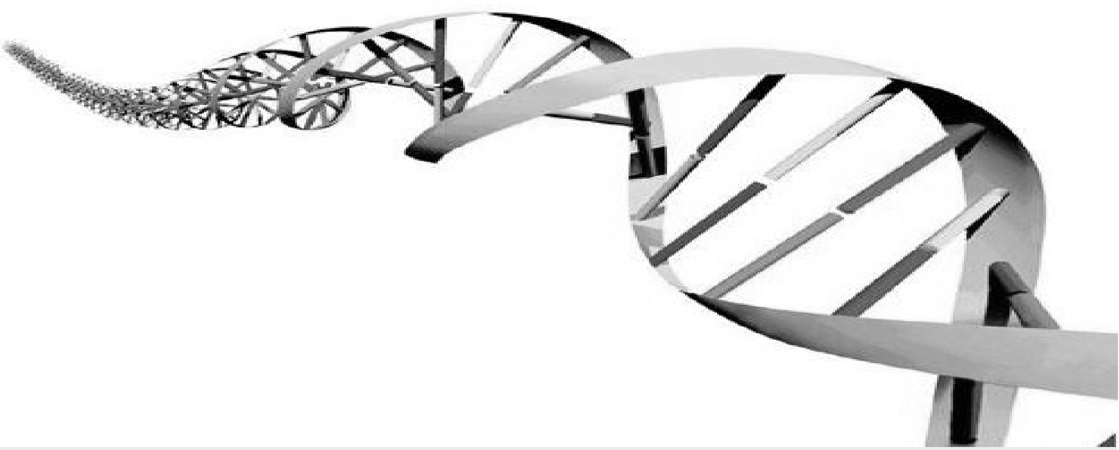
	4	8	12	16	20
Magentix2	6,4065	11,5978	25,0104	54,9346	139,6564
JADE	0,7978	0,9296	1,1846	1,388	1,993
Erlang	0,1539	0,19874	0,21998	0,23188	0,22842

number of computers

eXAT

- The erlang eXperimental Agent Tool
- Developed between 2005 – 2012
- FIPA compliant, AI libraries integrated
- Low performance, compared to pure Erlang

Evolutionary Computation



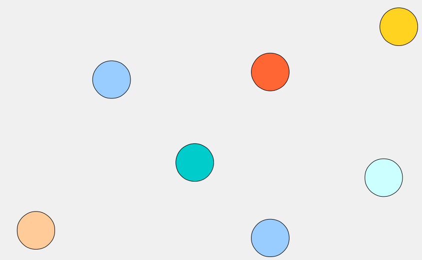
Evolutionary computation

- Group of computational intelligence methods
- Inspired by biological evolution
- Suitable for solving some optimization problems

  - Genetic algorithms
  - Evolution strategy
  - Ant colony
  - Particle swarm optimization
  - Bee Colony
  - ...

Genetic algorithm

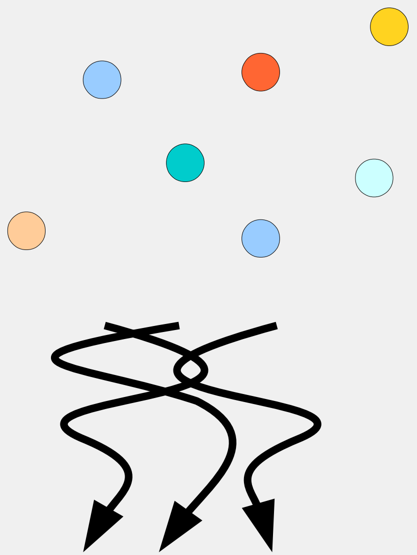
- Search heuristic inspired by mechanism of natural selection
- Population of solutions
  - Initial random set



Genetic algorithm

- Search heuristic inspired by mechanism of natural selection

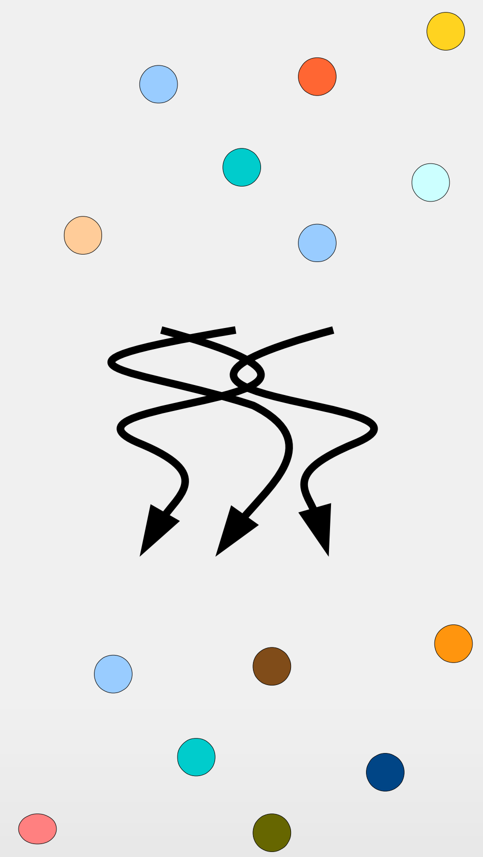
- Population of solutions
  - Initial random set
- Genetic operators
  - New solutions from old



Genetic algorithm

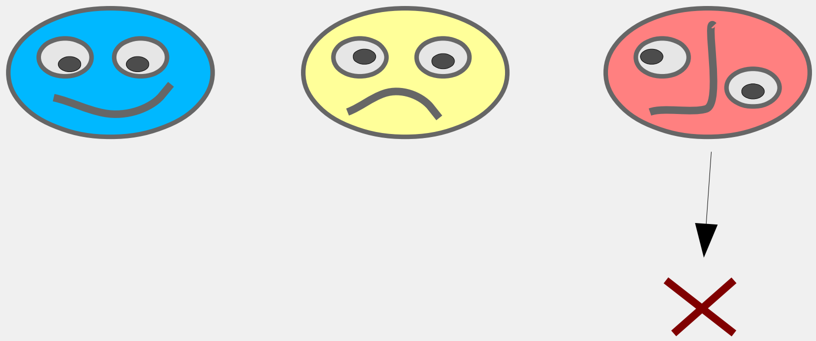
- Search heuristic inspired by mechanism of natural selection

- Population of solutions
  - Initial random set
- Genetic operators
  - New solutions from old
- Generations
  - Repeat until acceptable solution found



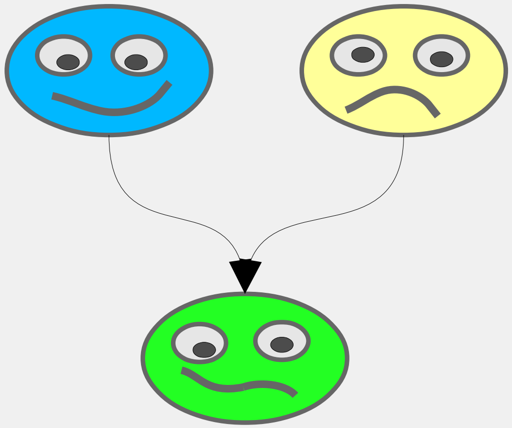
Selection

- Each solution is evaluated using fitness function
- Selected number of worst solutions is removed



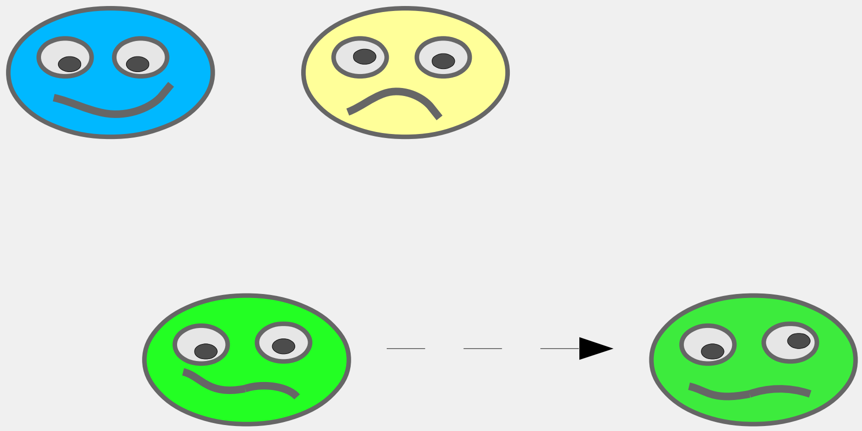
Crossover

- Remaining solutions are joined in pairs and used for creating new solution



Mutation

- Solution can be randomly modified
- With very low probability

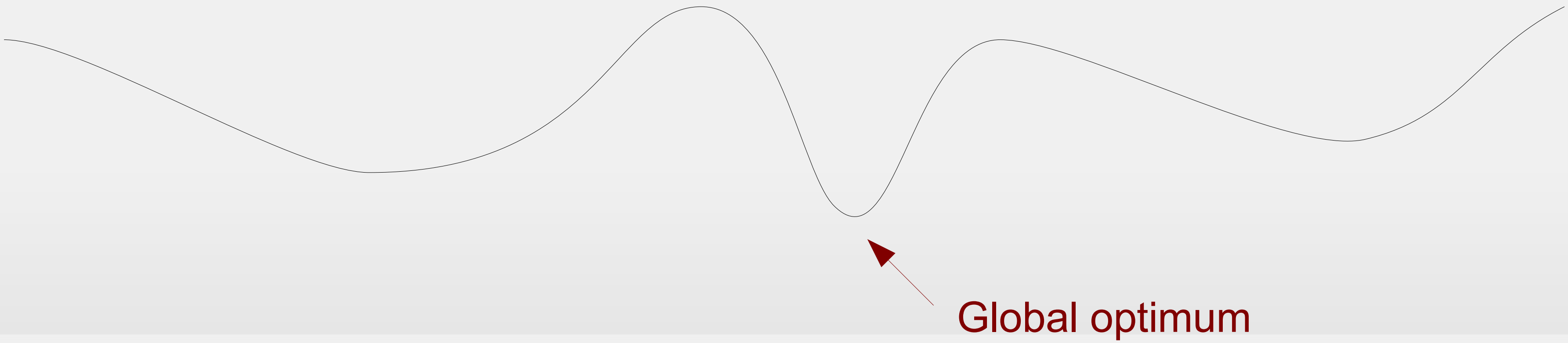


New generation

```
var population = createPopulation
while(!stopCondition)
     population = transform(selected(population ))
```

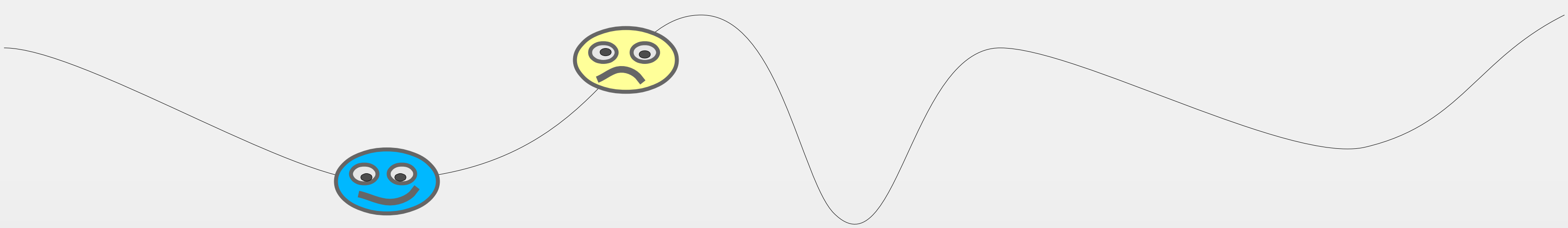
Genetic algorithm

- Constant number of solutions in each generation
- Fully sequential algorithm
    - Not like in real biological natural selection
- Slow convergence in many cases
- Blocking in local minima



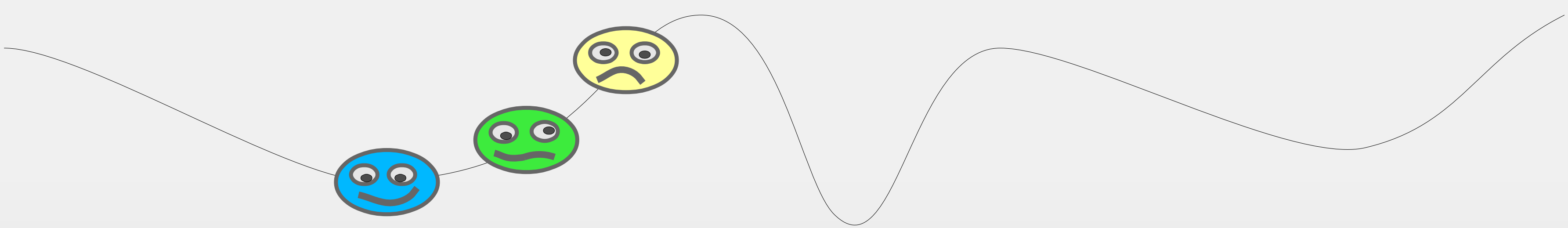
Genetic algorithm

- Constant number of solutions in each generation
- Fully sequential algorithm
  - Not like in real biological natural selection
- Slow convergence in many cases
- Blocking in local minima



Genetic algorithm

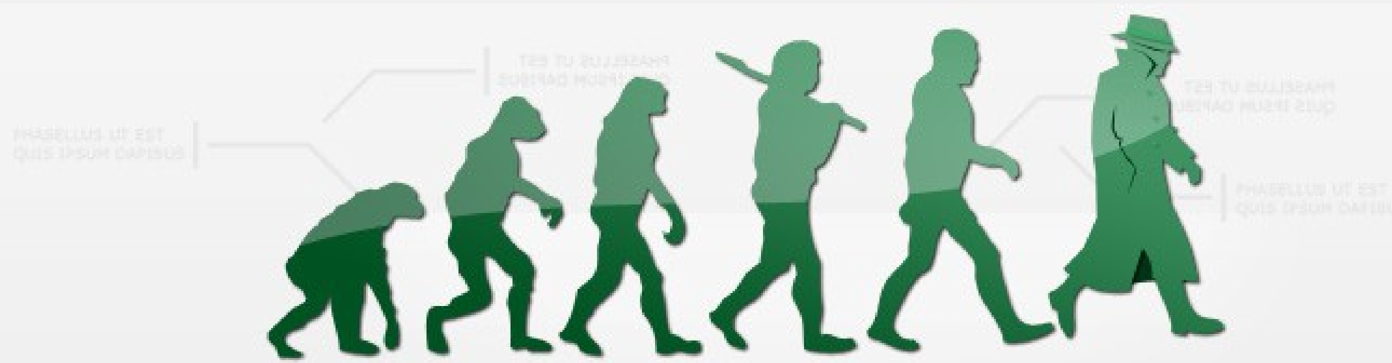
- Constant number of solutions in each generation
- Fully sequential algorithm
  - Not like in real biological natural selection
- Slow convergence in many cases
- Blocking in local minima



Agents + Evolution = AgE

AgE, eMAS

- How to use agent paradigm to overcome the problems?
  - Define a solution agent
  - Population of solution = multi agent system
  - Let the MAS work asynchronously – let the population of agent live and evolve

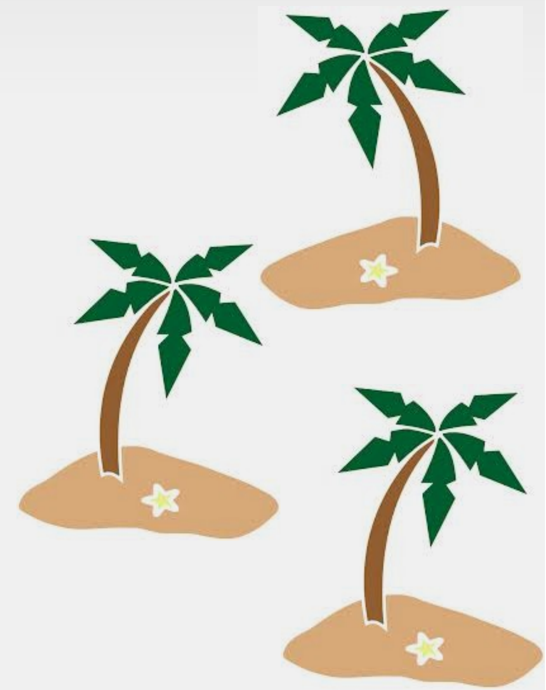


AgE, eMAS

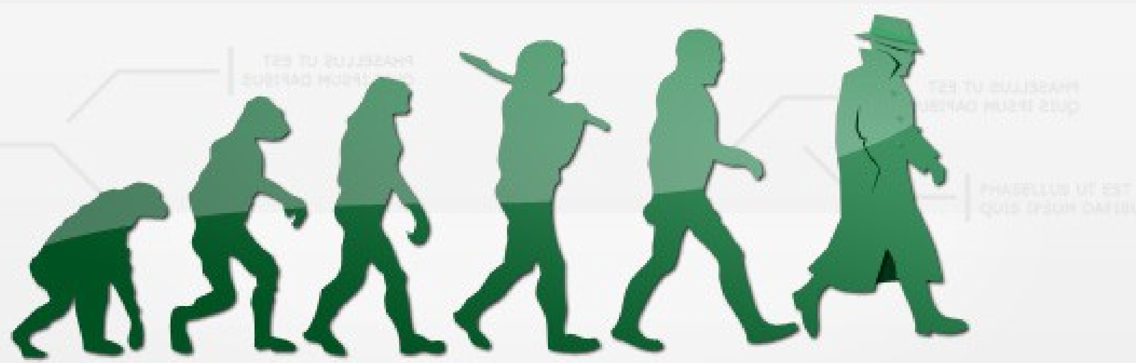
- Which agent should reproduce, which should die?
  - Define energy in the system, split between agents
  - Define actions depending on agent energy
    - Actions pass energy between agents
  - Number of agents vary, total energy is constant

AgE, eMAS

- How to solve local minima problem?
- Bio-inspire again!

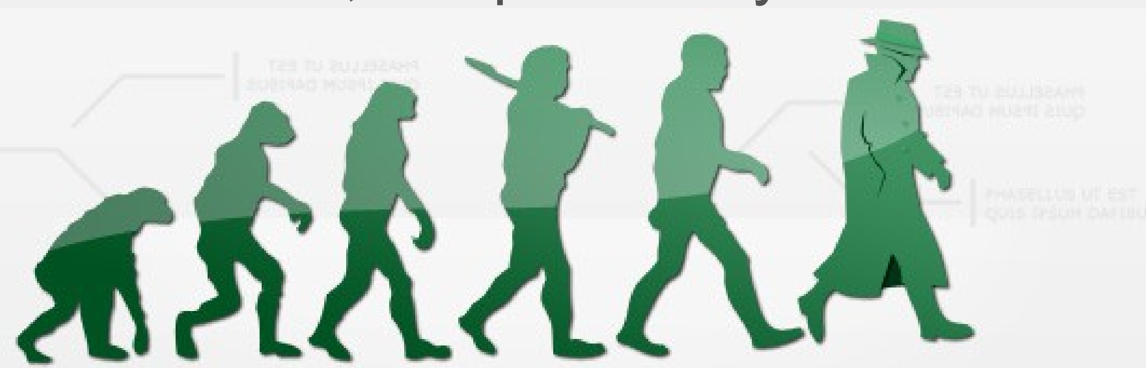


- Define the concept of **islands** – separated MASes
    - Agents can interact within own island only
- Define an action of **migration**



AgE, eMAS

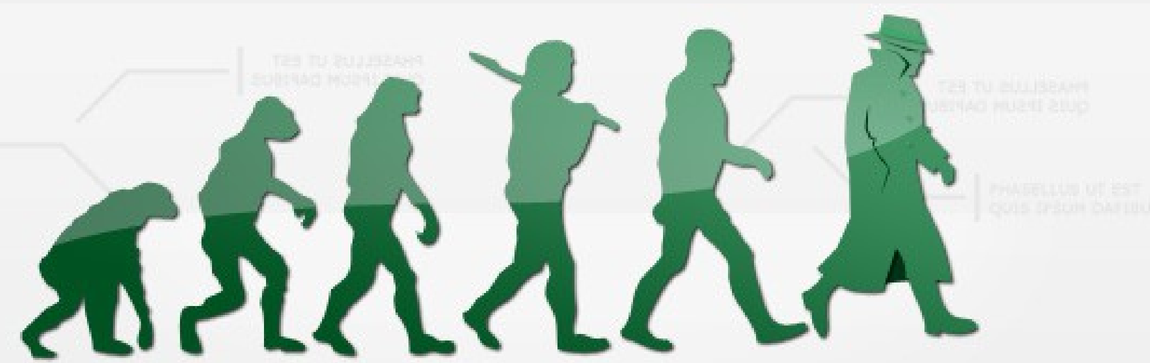
- AgE algorithm:
  - Each agent independently decides what to do, based on energy
  - Highest energy allows crossover, which results in passing some energy to children
  - Child can mutate on its birth, low probability
  - Medium energy allows fighting – better agent overtakes some energy
  - Zero energy causes agents death
  - New action: migration between islands, low probability



AgE, eMAS

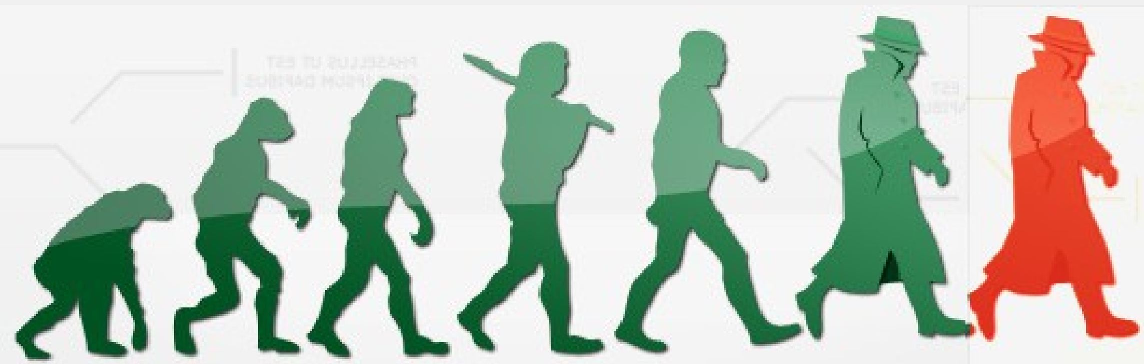
- AgE implementation:
  - In Java... some time ago
  - Impossible to make all agents asynchronous
  - Asynchronous islands, synchronous processing within an island

AgE
AGENT-BASED EVOLUTION

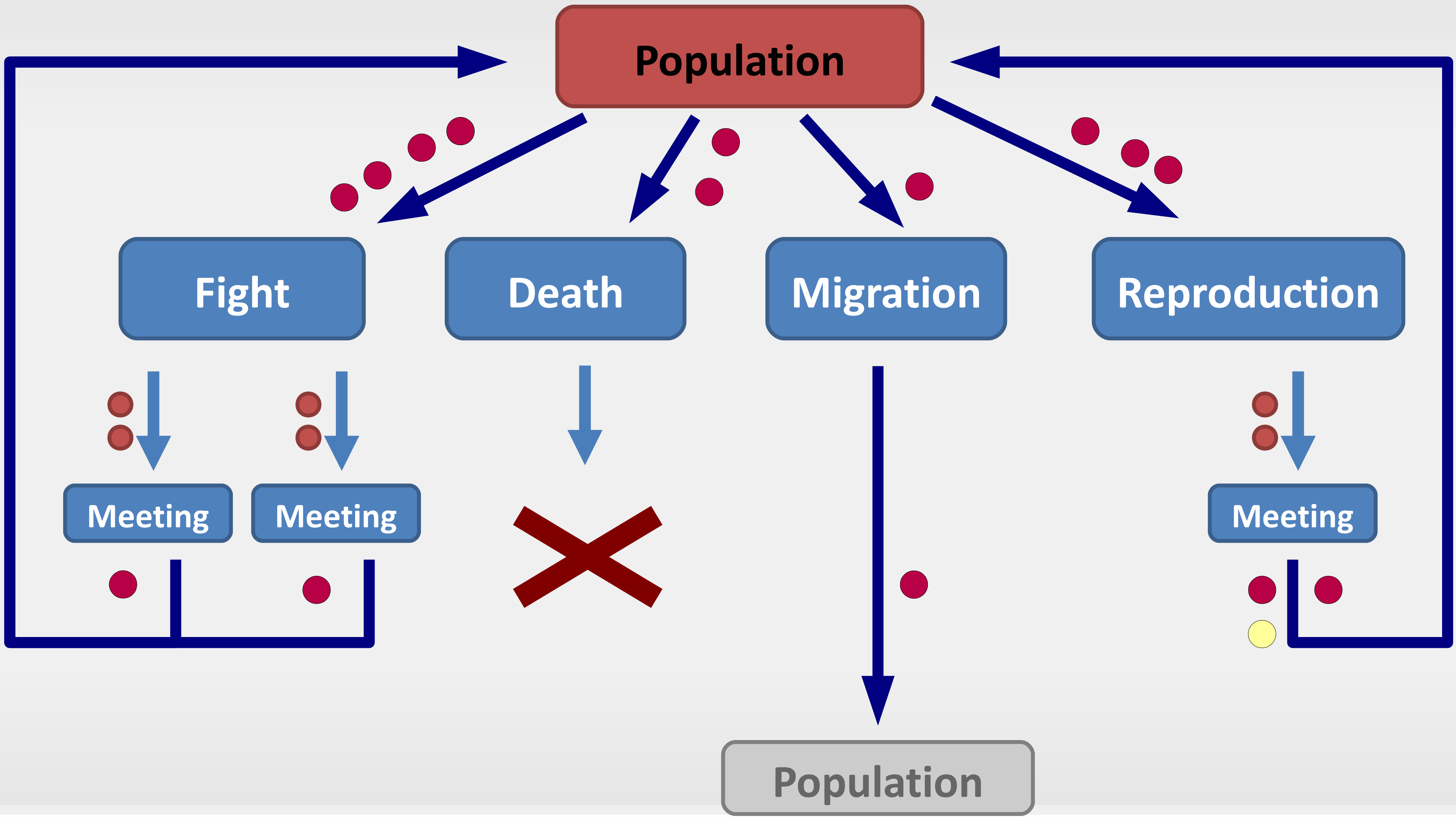


AgE in Erlang

- AgE implementation in **Erlang**:
  - Fully asynchronous, finally possible!
  - Each agent/solution is a process
  - Processes decide what to do
  - EVM scheduler decides which agent acts when
  - Arena processes for performing actions



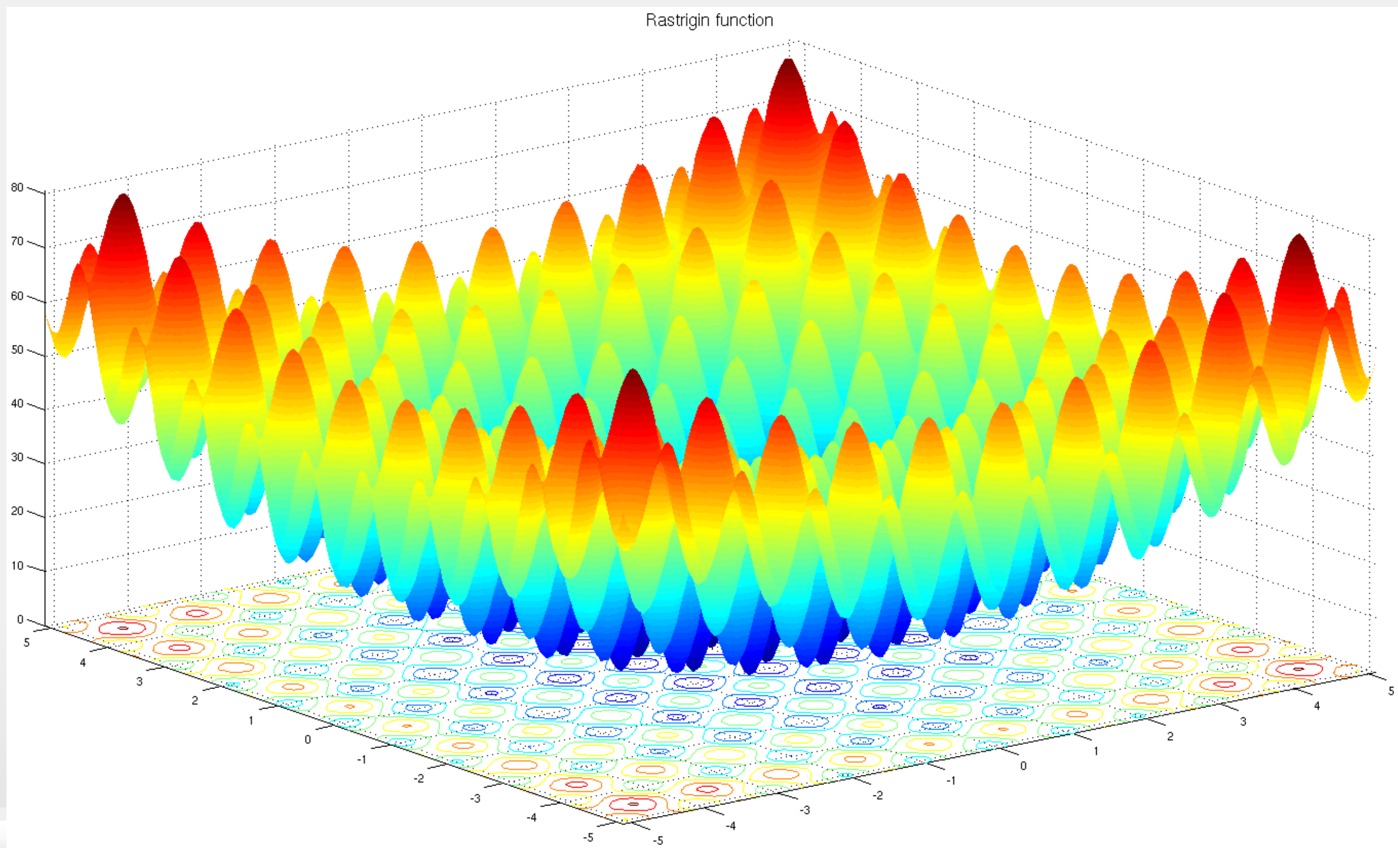
AgE in Erlang – arenas



AgE in Erlang – tests

- Optimization of Rastrigin function
- 1000 dimensions
- 64 islands

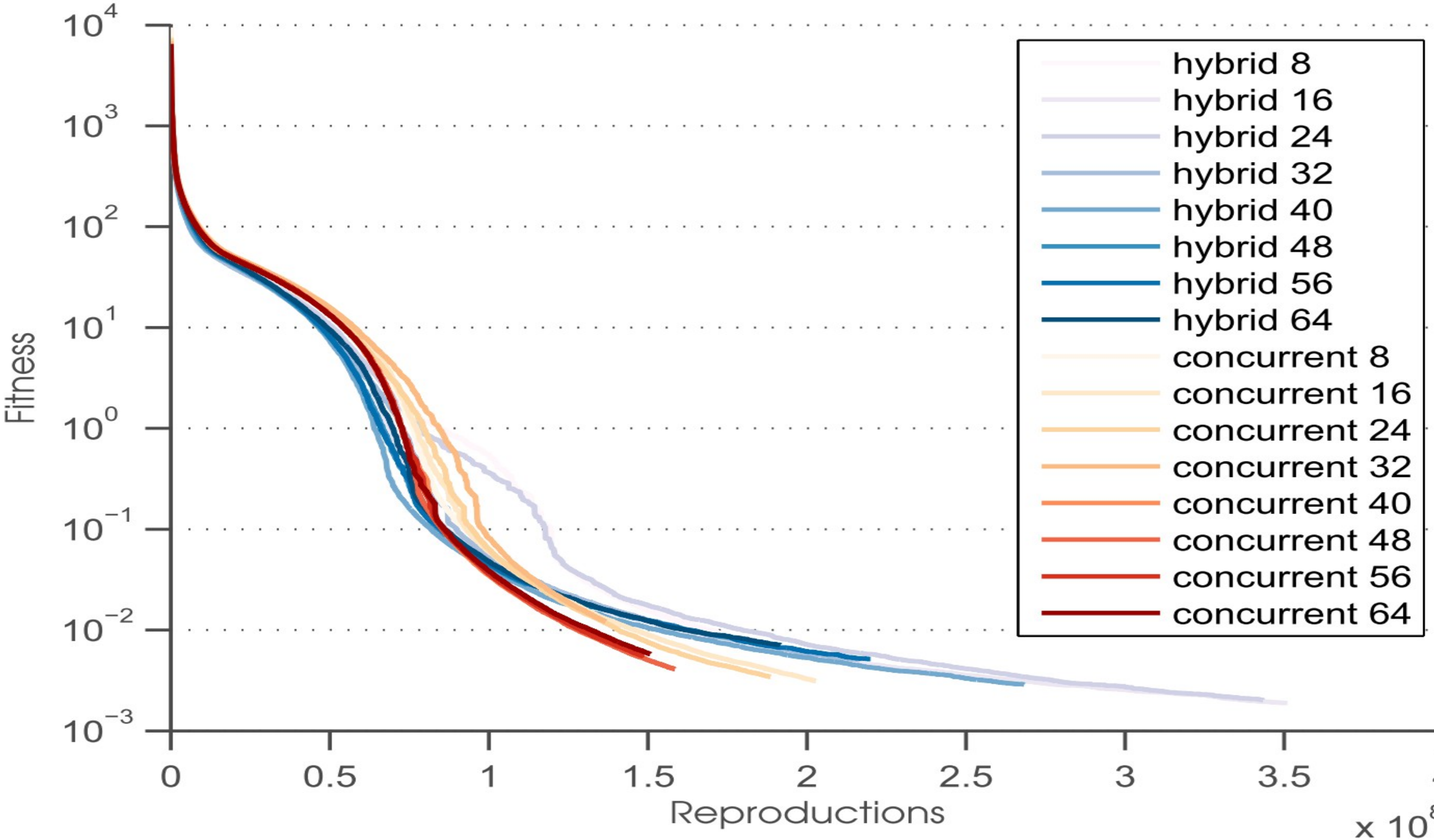
$$f(\mathbf{x}) = An + \sum_{i=1}^{n} \left[x_i^2 - A\cos(2\pi x_i) \right]$$



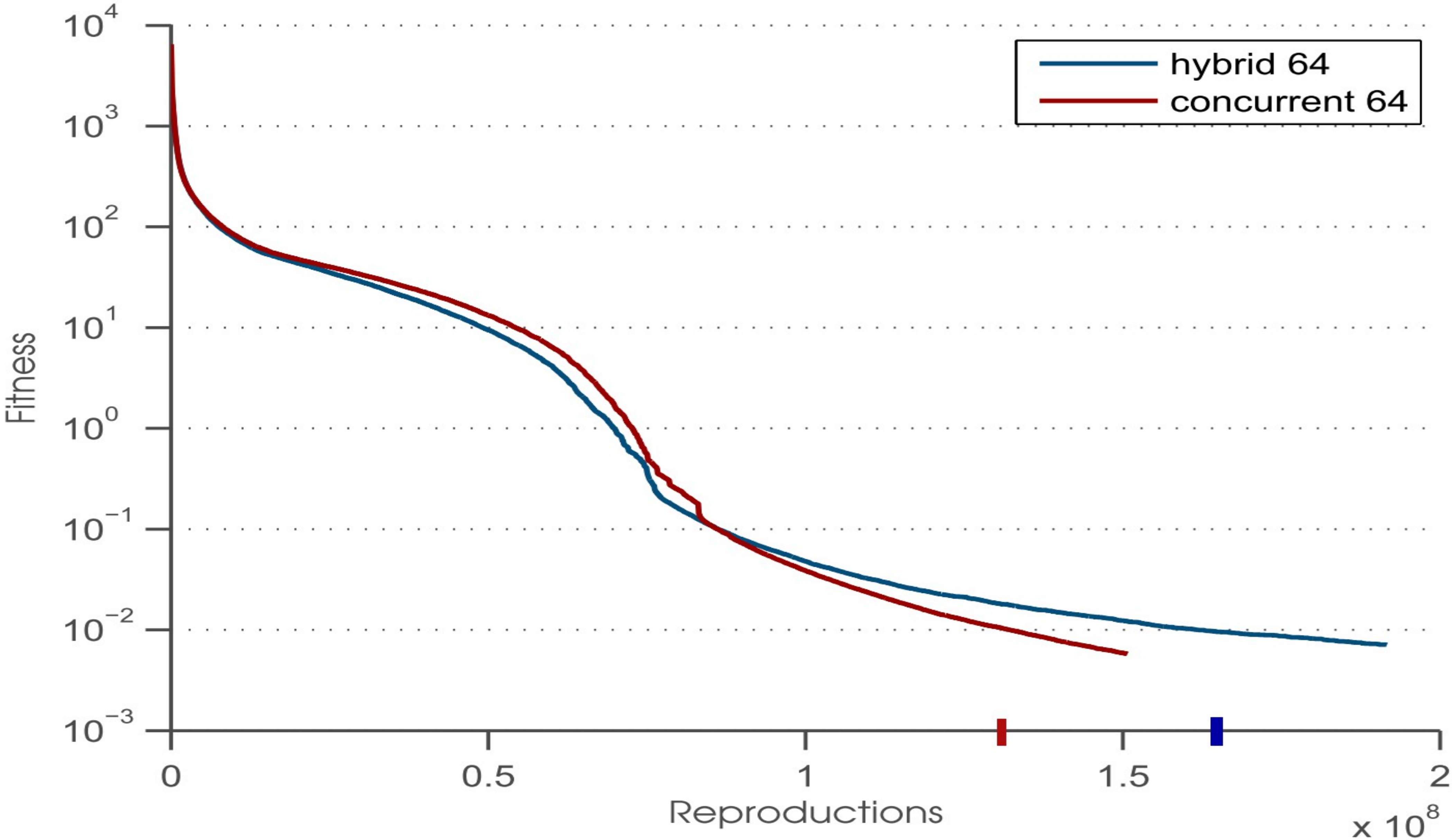
AgE in Erlang – computer

- Academic Computer Centre CYFRONET AGH Kraków, Poland
- Zeus Computing cluster
  - operating system: Scientific Linux
  - processors: Intel Xeon, AMD Opteron
  - cores: 25468
  - RAM: 60 TB
  - computing power: 267 Tflops
- In test:
  - 1 hardware node,
  - 64 cores

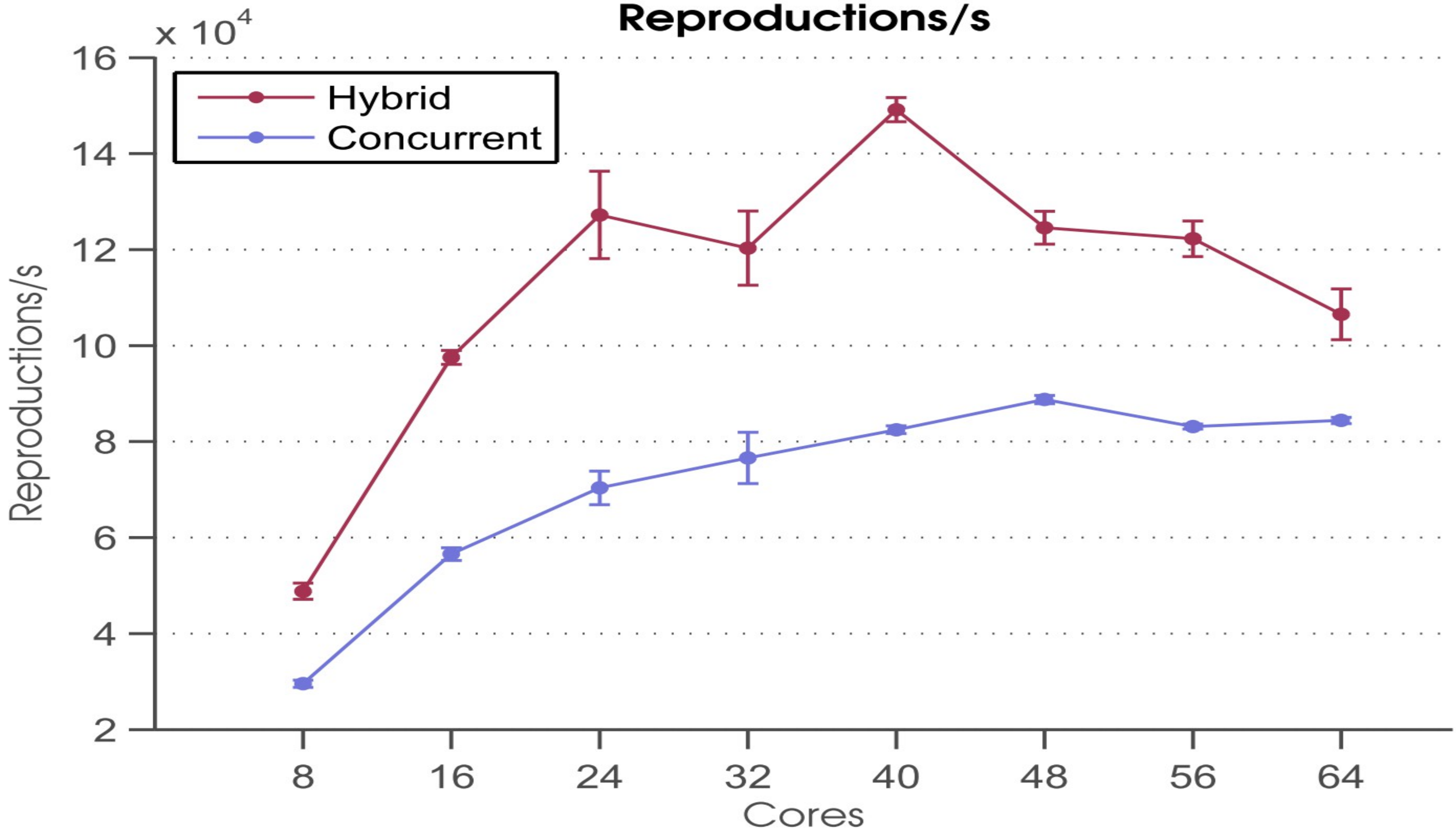
AgE in Erlang – results



AgE in Erlang – results



AgE in Erlang – results



AgE in Erlang – further work

- Paraphrase patterns in Erlang-AgE
- Including GPU
- Different applications / optimization tasks
- Various variants of algorithm on multicore computers