Agent-based Evolutionary Computing

Wojciech Turek

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

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

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

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...

\[
\text{spawn} \quad \text{p1}
\]

\[
\text{p2}
\]
Actors descendants

- 1973: Actor model defined
- 1991: Erlang released
- 2006: Symmetric multiprocessing in EVM

Erlang supremacy!
Actors descendants

- Actor model defined: 1973
- ???
- Erlang released: 1991
- Symmetric multiprocessing in EVM
- Erlang supremacy: 2006

Agent-based Evolutionary Computing
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

![Diagram showing time of agent creation versus number of agents already in the system for different agent systems: Magentix2, JADE, Erlang. The x-axis represents the number of agents, and the y-axis represents the time in milliseconds.]
Agent-based Evolutionary Computing

Basic performance

- Agent messaging vs Erlang messaging – single node

![Graph showing performance comparisons](image-url)
### Basic performance

- Agent messaging vs Erlang messaging – multiple nodes

<table>
<thead>
<tr>
<th>Number of Computers</th>
<th>Magentix2</th>
<th>JADE</th>
<th>Erlang</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>6.4065</td>
<td>0.7978</td>
<td>0.1539</td>
</tr>
<tr>
<td>8</td>
<td>11.5978</td>
<td>0.9296</td>
<td>0.19874</td>
</tr>
<tr>
<td>12</td>
<td>25.0104</td>
<td>1.1846</td>
<td>0.21998</td>
</tr>
<tr>
<td>16</td>
<td>54.9346</td>
<td>1.388</td>
<td>0.23188</td>
</tr>
<tr>
<td>20</td>
<td>139.6564</td>
<td>1.993</td>
<td>0.22842</td>
</tr>
</tbody>
</table>
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

Global optimum
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
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
• 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
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 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 – tests

- Optimization of Rastrigin function
- 1000 dimensions
- 64 islands

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

The graph shows the fitness over reproductions for different configurations of AgE in Erlang. The x-axis represents the number of reproductions, and the y-axis represents the fitness on a logarithmic scale. The configurations include hybrid and concurrent setups with varying numbers of individuals, as indicated by the legend on the right side of the graph.
AgE in Erlang – results

The graph shows the fitness over reproductions for two different configurations: hybrid 64 and concurrent 64. The fitness decreases as the number of reproductions increases, indicating an improvement in performance or adaptability. The hybrid 64 configuration shows a slightly lower fitness than the concurrent 64 configuration at higher reproduction counts.
AgE in Erlang – further work

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