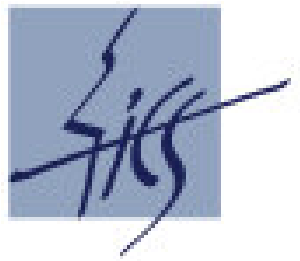


Autonomous Behaviour of Computational Agents (Part I)

Roman Neruda
roman@cs.cas.cz

http://www.cs.cas.cz/bang/

Institute of Computer Science,
Academy of Sciences of the Czech Republic,
Prague, Czech Republic

Objectives

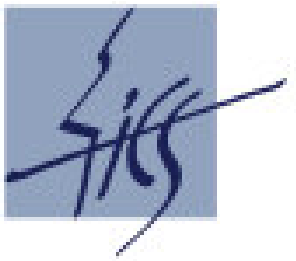
Formal description of computational agents

- to reason about agents
- to reason about MAS
- to configure MAS
- to evolve MAS
- to interact with ontology based knowledge systems

Outlines

- Computational MAS
- Agents description
  - gates, interfaces, properties
- MAS descriptions
  - agents, connections, characteristics
- Implementation
- Future work

Computational intelligence

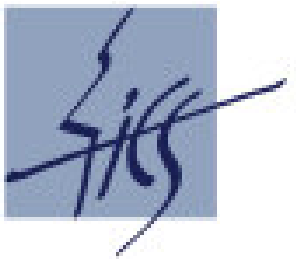


- Soft computing (L.Zadeh): creative fusion of ANNs, EAs, FLCs, ...
- Benefits over individual methods
- No one underlying theory
- Importance of heuristics, experiments
- Practical skills required
- ... and we don't have to focus on the SC only (statistics, numerical analysis, ...

Multi-agent systems (MAS)

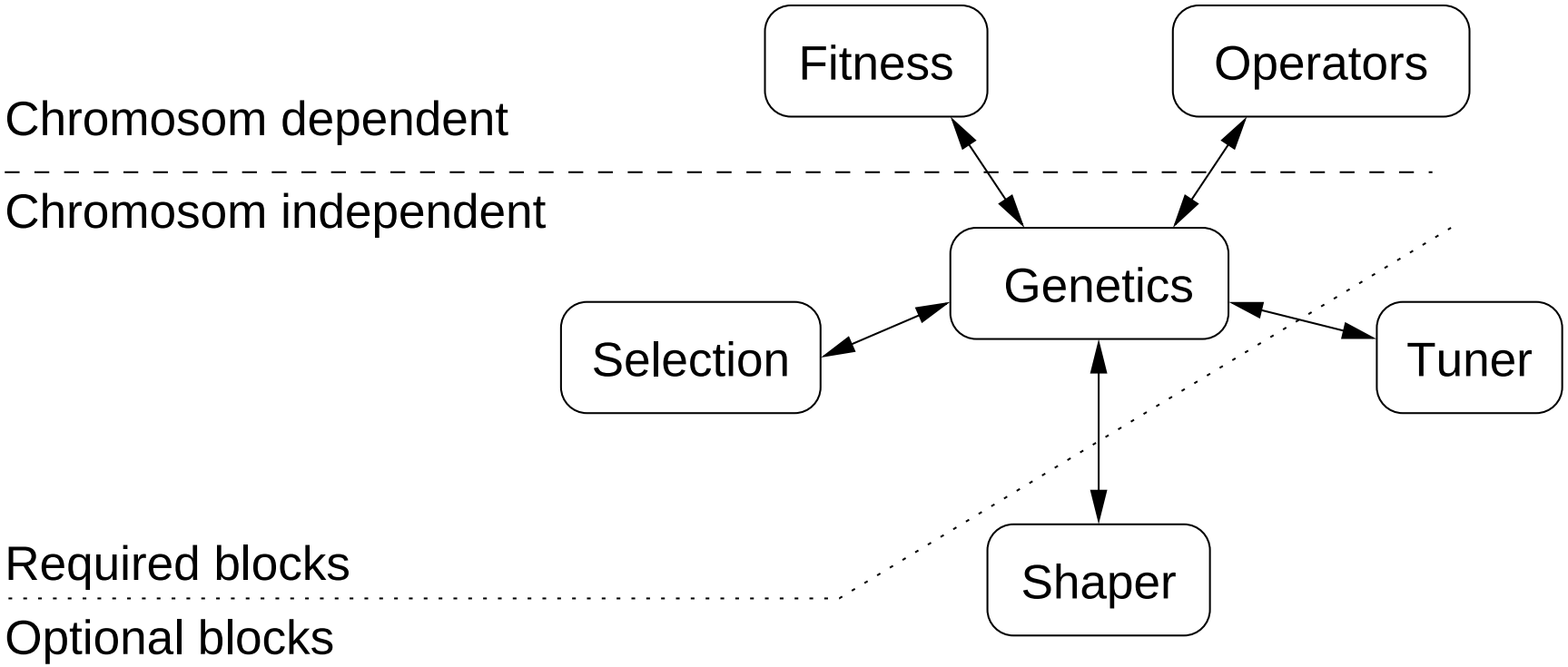
- Agents encapsule computational algorithms
- Distributed execution (cluster of workstations)
- Interchangeability of agents/methods
- Autonomous behavior (connection, negotiations, pro-activity)
- Emergence (more compelx models, evolution)

Agents in the Bang system

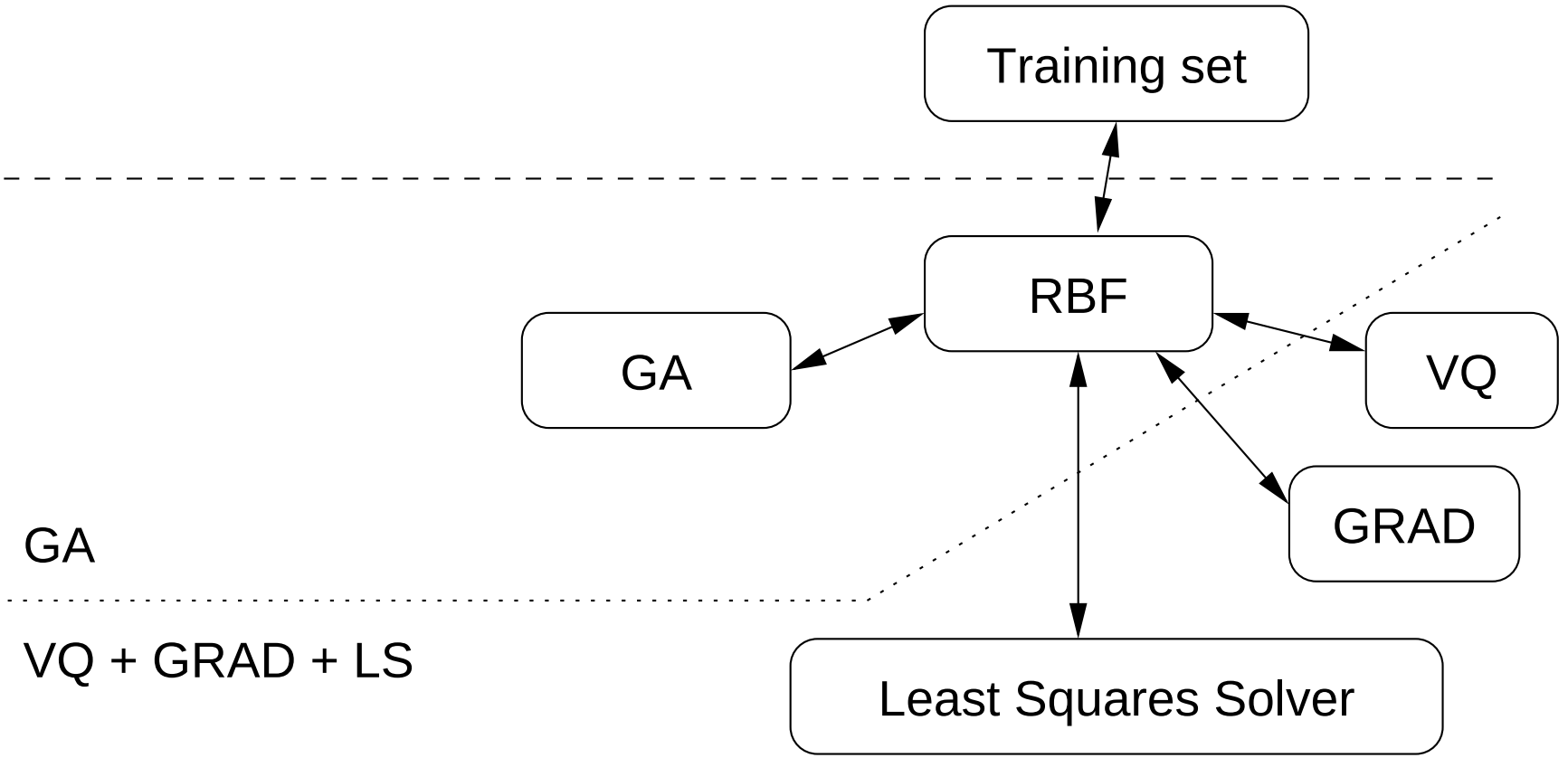


- **computational agents:** neural nets (MLP, RBF), GA suite, Kohonen maps, vector quantization, decission tree
- **computational helpers:** linear system solver, gradient descent optimization
- **task-related:** data source, task manager, file system wrapper
- **system:** launcher, yellow pages, ontology services, debugger, profiler
- **other:** MASman, console, GUI

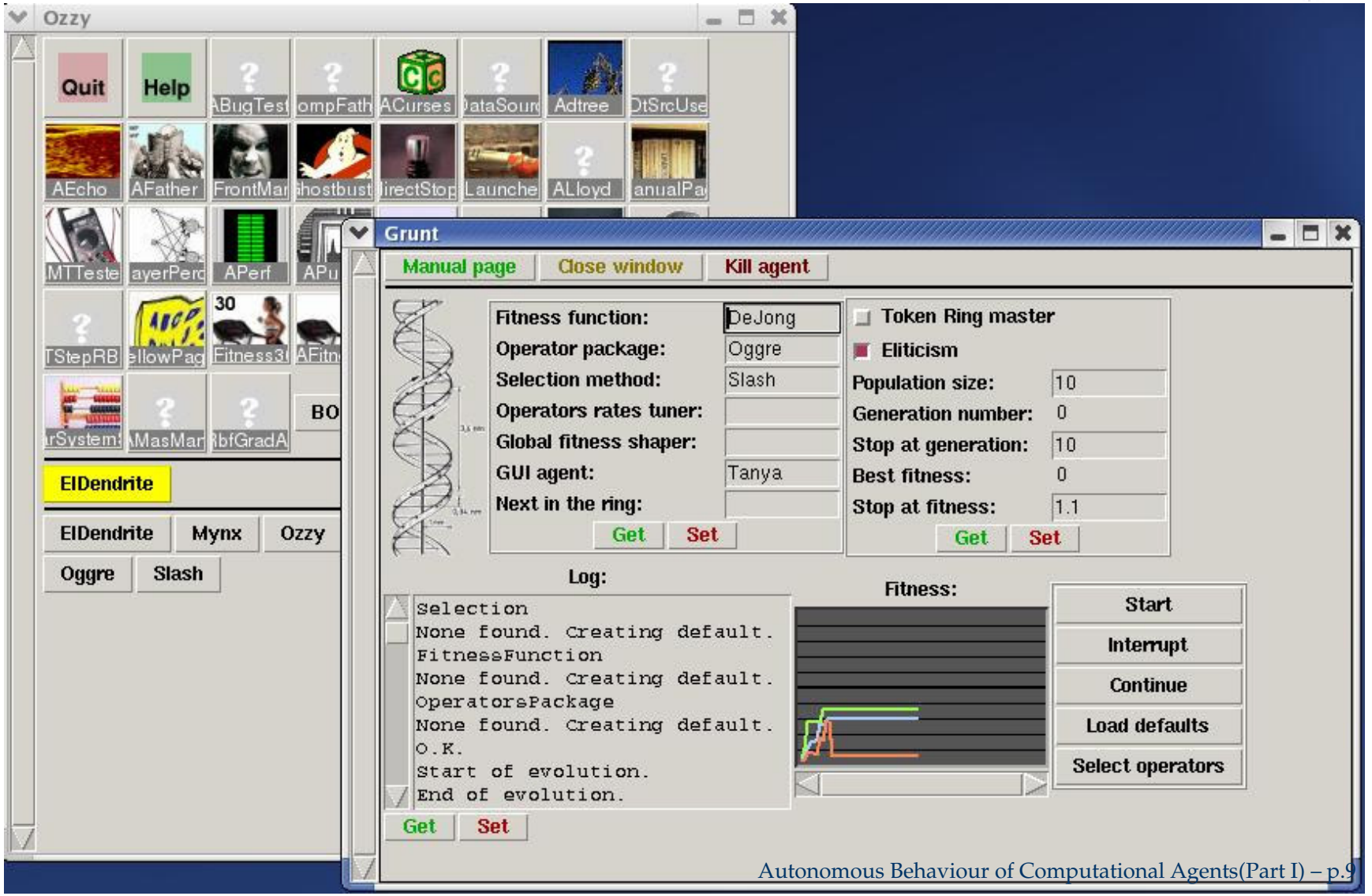
Example: GA as MAS



Example: RBF as MAS



GAs in action



Definitions: Communication

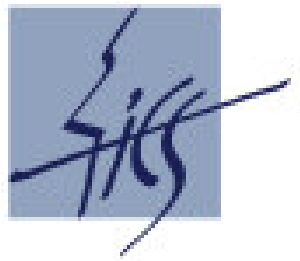
Message type: identifies a category of messages that can be send to an agent in order to fulfill a specific task.

Interface: the set of message types understood by a class of agents.

Gate: a tuple consisting of a message type and a named link.

Connection: a triple consisting of a sending agent, the sending agent's gate, and a receiving agent.

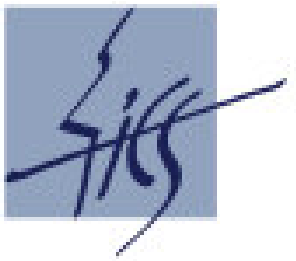
Definitions: Agents and MAS



Agent class: defined by an interface, a set of message types, a set of gates, and a set of types.
Agent: an instance of an agent class. It is defined by its name and its class.

Multi-Agent Systems (MAS): consist of a set of agents, a set of connections between the agents, and the characteristics of the MAS.

Concepts and roles



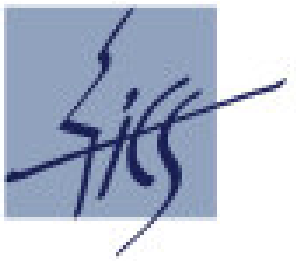
Concepts	
mas(C)	C is a Multi-Agent System
class(C)	C is the name of an agent class
gate(C)	C is a gate
m_type(C)	C is a message type

Roles	
type(X,Y)	Class X is of type Y
has_gate(X,Y)	Class X has gate Y
gate_type(X,Y)	Gate X accepts messages of type Y
interface(X,Y)	Class X understands mess. of type Y
instance(X,Y)	Agent X is an instance of class Y
has_agent(X,Y)	Agent Y is part of MAS X

Computational agent

class(decision_tree)
type(decision_tree, computational_agent)
has_gate(decision_tree, data_in)
gate_type(data_in, training_data)
interface(decision_tree, control_messages)
. . .

Trusted MAS

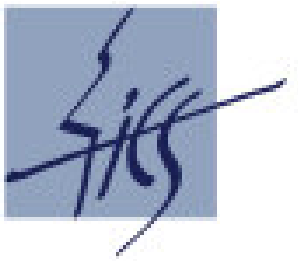


trusted_MAS(MAS) ←
findall(X, has_agent(MAS,X), A))∧
all_trusted(A)
all_trusted([]) ← true
all_trusted([F|R]) ←
instance(F,FC)∧
type(FC, trusted)

A MAS is trusted if all of its agents are trusted.

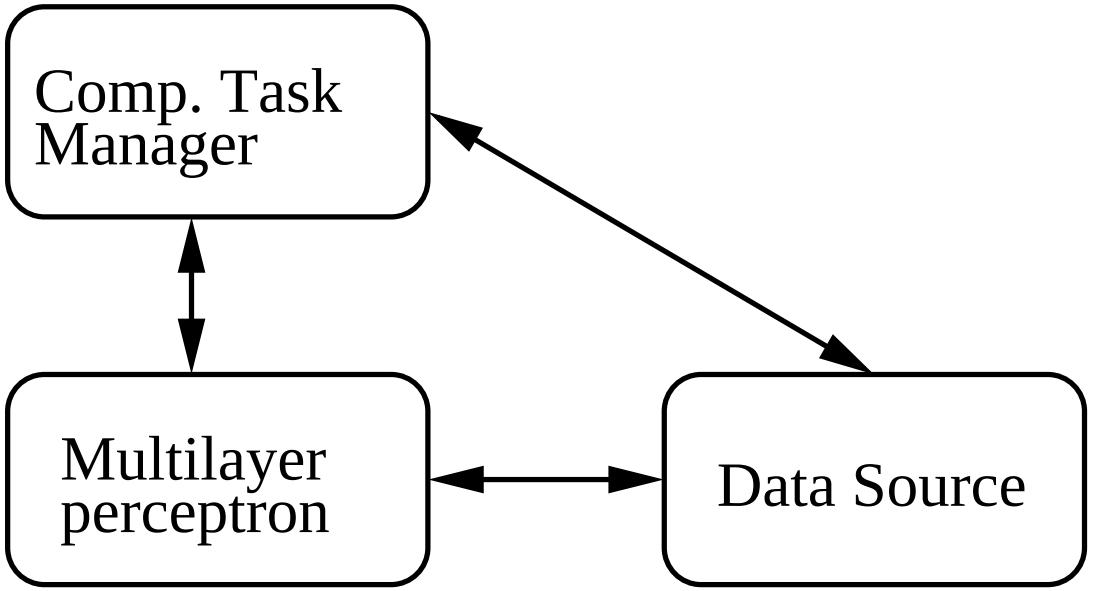
Prolog predicate `findall` returns a list of all instances of a variable for which a predicate is true.

Computational MAS

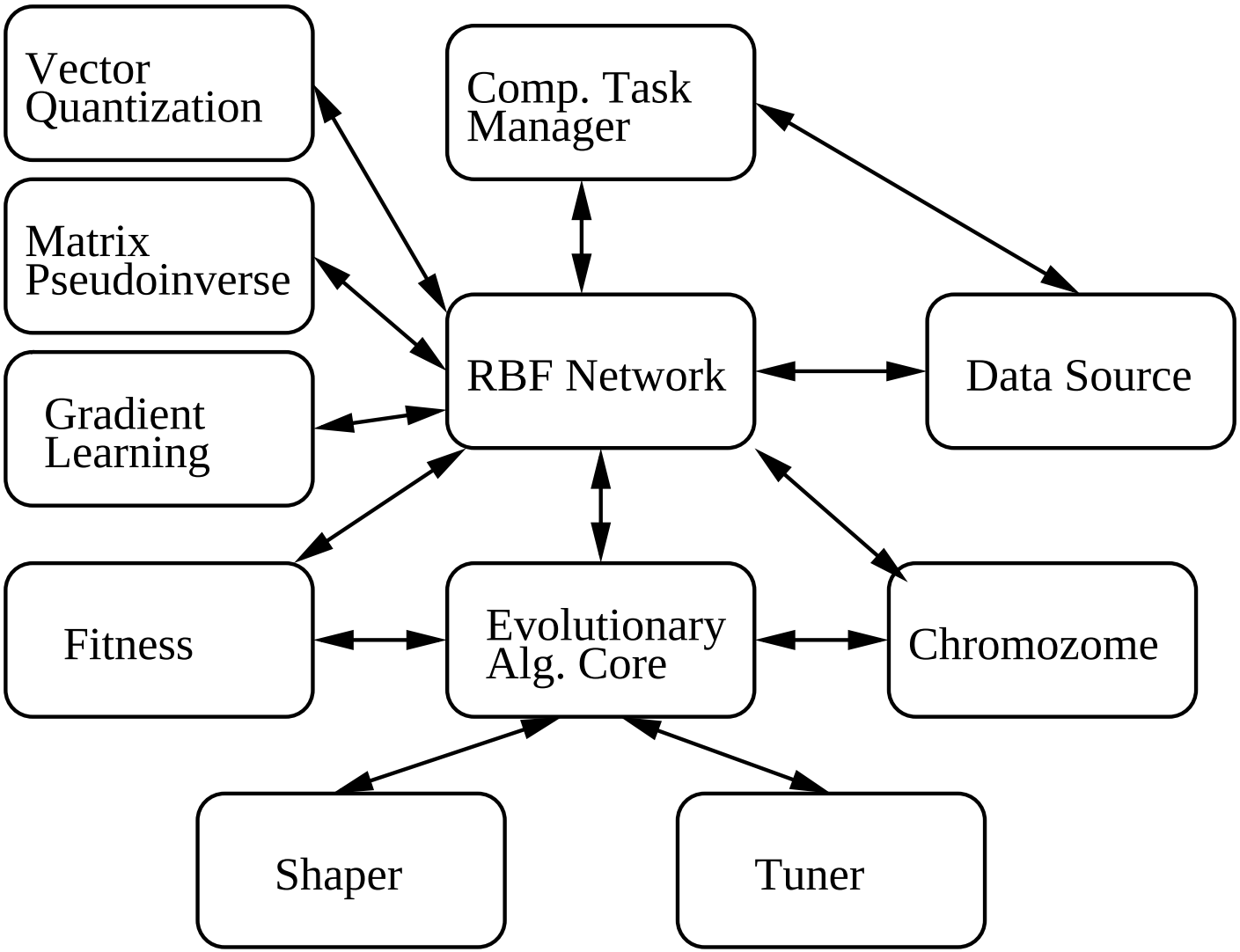


comp_MAS(MAS) ←
type(CAC, computational_agent)∧
instance(CA, CAC)∧
has_agent(MAS, CA)∧
type(DSC, data_source)∧
instance(DS, DSC)∧
has_agent(MAS, DS)∧
connection(CA, DS, G)∧
type(TMC, task_manager)∧
instance(TMC, TM)∧
has_agent(MAS, TM)∧
connection(TM, CA, GC)∧
connection(TM, GC, GD)

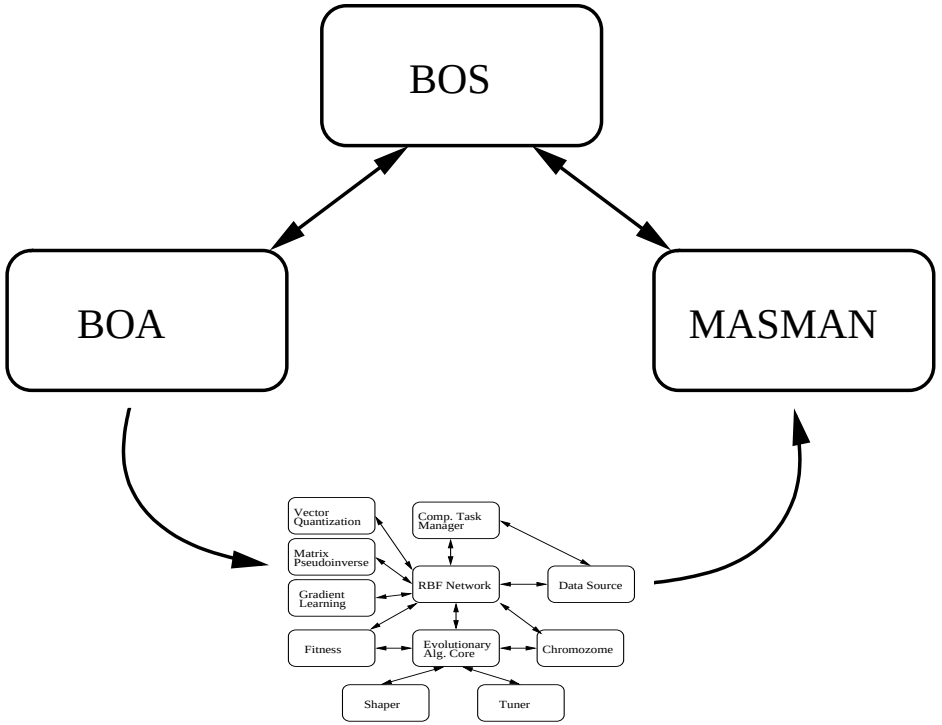
Simple C-MAS



Less simple C-MAS



Implementation



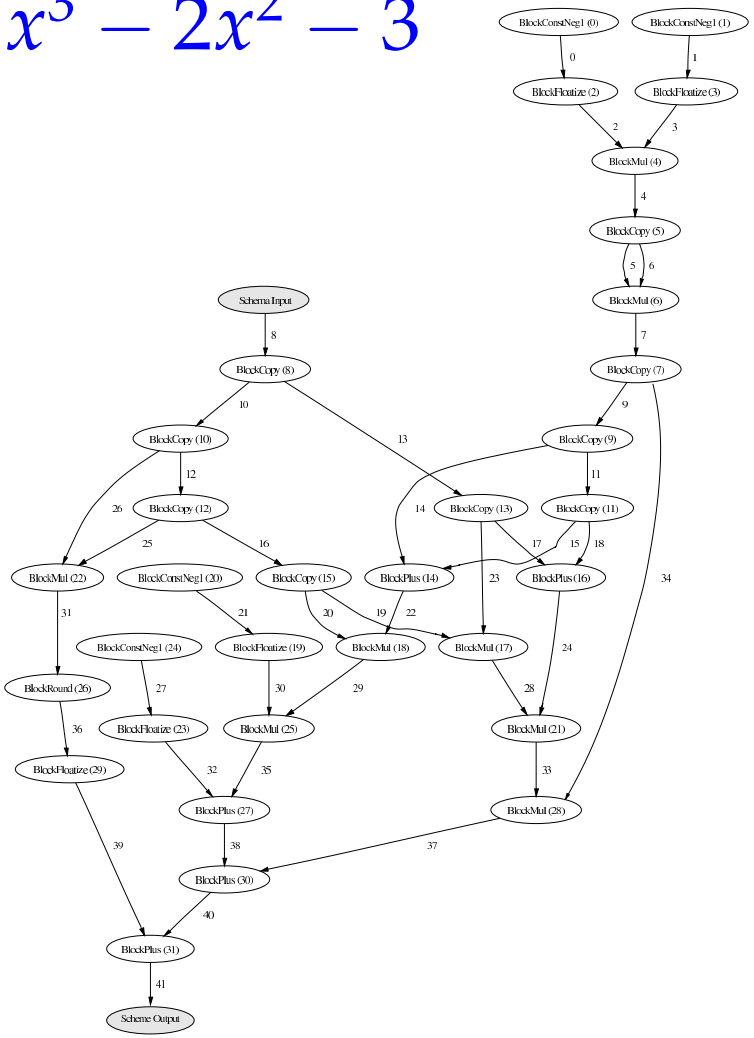
BOA agent generates a MAS configuration description and sends it to the MAS manager agent, which takes care of MAS creation and run. They both query the BOS ontology services agent.

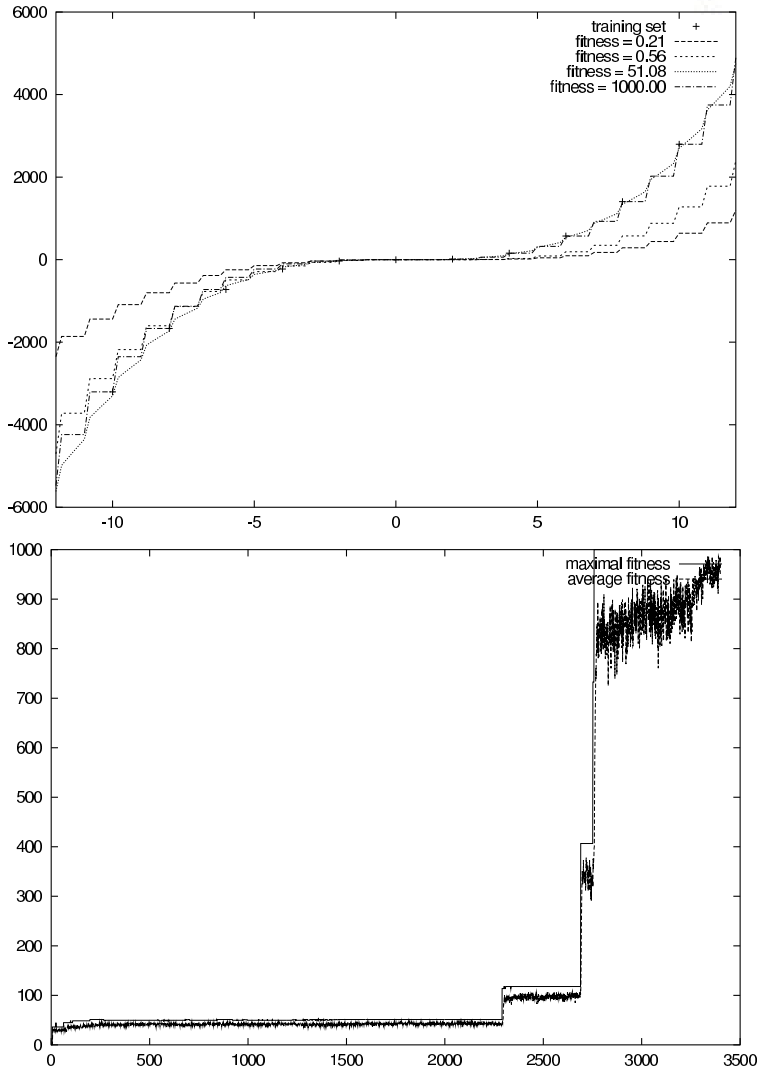
Experiments with evolution I

- **Evolving arithmetical functions**
  - $2x + 1$
  - 0
  - $x^3 - 2y - 3$
  - $\ldots$
- **Success depends on**
  - Function complexity
  - Initial population
  - Operator set and their parameters ($x^2 + y^2$ vs. $x^2 + y^2 + 1$)

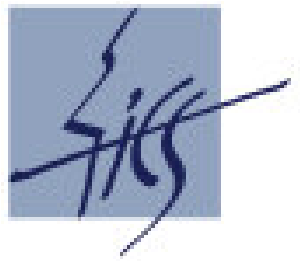
Experiments with evolution II

$$x^3 - 2x^2 - 3$$





Conclusions



- Connection to description logics reasoner (RACER)
- Going to WWW: HTTP/HTML interface
- Advertising services via ontologies
- Interactive MAS generation

http://www.cs.cas.cz/bang/