Bang: A Computational Multi-Agent System

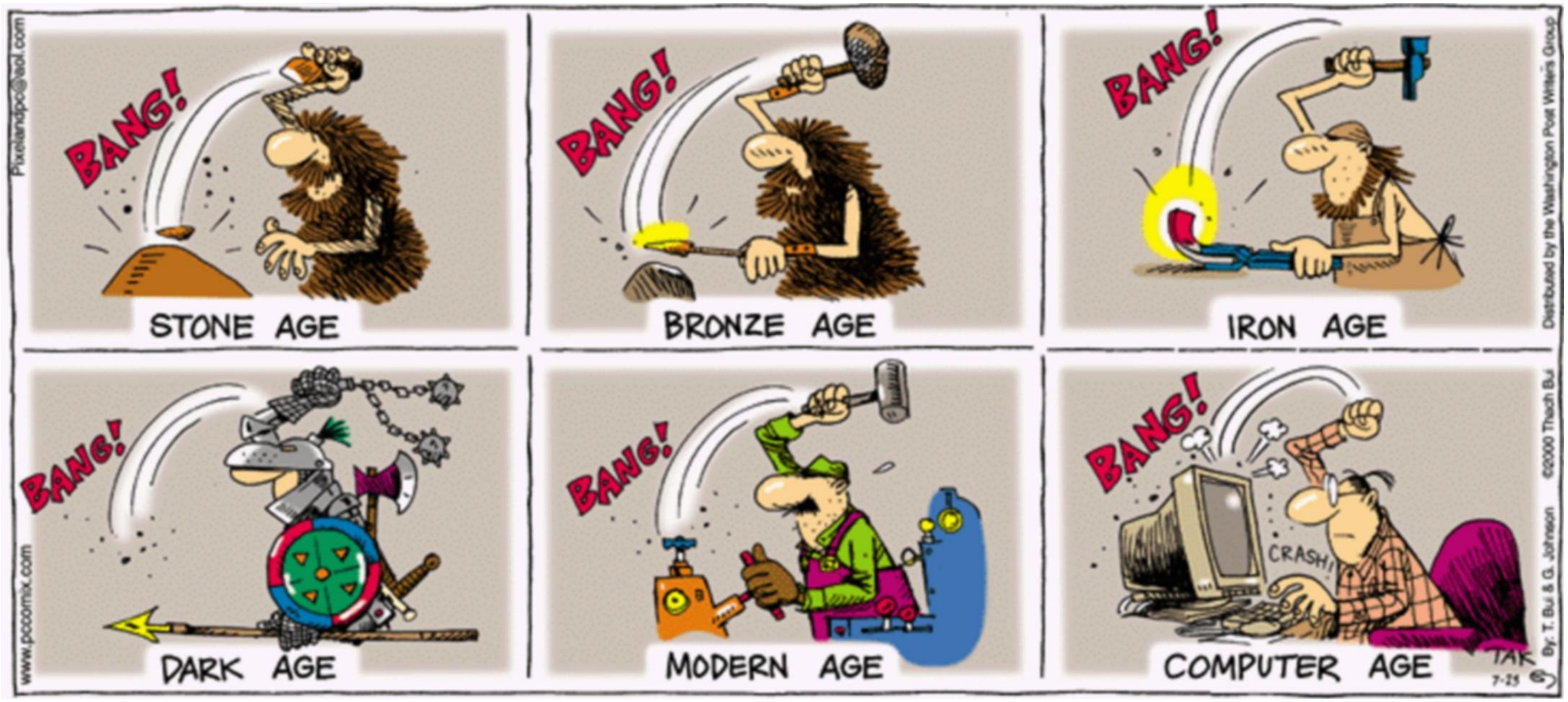
Creating hybrid AI models should be easy (or at least easier)

Roman Neruda
roman@cs.cas.cz

http://bang.sf.net

Institute of Computer Science, ASCR
Prague, Czech Republic

bang


Pixelandpo@aol.com
BANG!
STONE AGE
BANG!
BRONZE AGE
BANG!
IRON AGE
Distributed by the Washington Post Writers Group
www.pccomix.com
BANG!
DARK AGE
BANG!
MODERN AGE
BANG!
CRASH!
COMPUTER AGE
©2000 Thach Bui
By: T. Bui & G. Johnson
TAK
7-25

What?

Hybrid computational models

- 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, ...)

How?

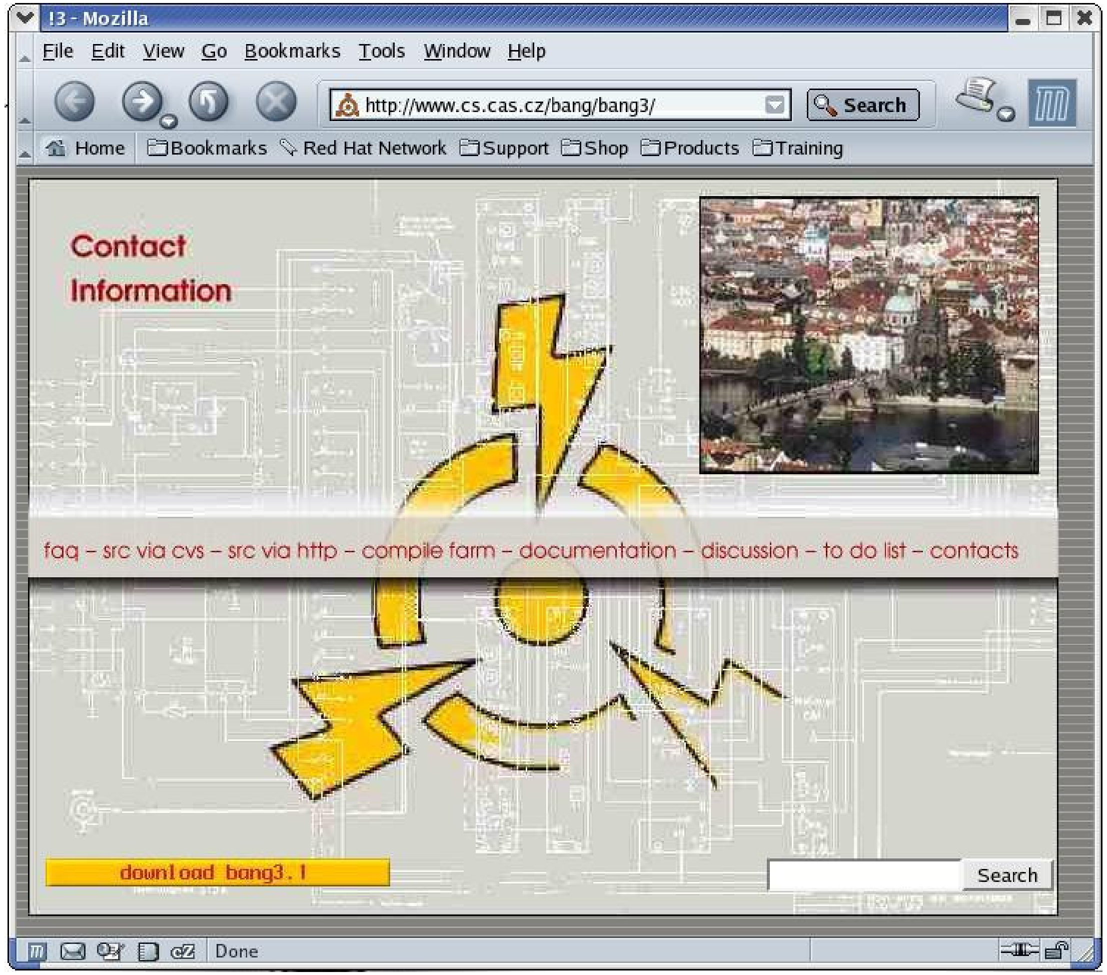
Multi-agent systems (MAS)

- Agents encapsule computational algorithms
- Distributed execution
- Interchangeability
- Autonomous behavior
- Emergence

Where?

Bang:

- tool for creating multi-agent computational systems
- creation, distributed run, performance analysis


!3 - Mozilla
File Edit View Go Bookmarks Tools Window Help
http://www.cs.cas.cz/bang/bang3/
Search
Home
Bookmarks
Red Hat Network
Support
Shop
Products
Training
Contact Information
faq – src via cvs – src via http – compile farm – documentation – discussion – to do list – contacts
download bang3.1
Search
Done

Why?

- combinations rather than individual methods
- complexity estimation and real-time analysis
- distributed execution (clusters of workstations)
- as autonomous/automated as possible
- for researchers and users

Who?

FACULTAS MATHEMATICA PHYSICAQUE

UNIVERSITÄT
KOBLENZ · LANDAU

CALIFORNIA STATE UNIVERSITY, CHICO · 1887 · ARS PROBAT ARTIFICEM

Talk outlines

- Agents and MAS
- Agents that socialize
- Agents that are clever
- Agents that evolve

PART I

- Agents and MAS
  - agent definition
  - computational agents
  - Bang as a ’middleware’
- Agents that socialize
- Agents that are clever
- Agents that evolve

Autonomous agent

- a system situated within,
- and a part of an environment,
- senses that environment,
- and acts on it, over time,
- in pursuit of its own agenda,
- and so as to efect what it senses in the future.

[S. Franklin: Is it an agent or just a software?]

Intelligent agent

- **pro-activeness:** able to exhibit goal-directed behavior by taking the initiative in order to satisfy their design objectives;
- **reactivity:** able to perceive their environment, and respond in a timely fashion to changes that occur in it in order to satisfy their design objectives;
- **social ability:** capable of interacting with other agents (and possibly humans) in order to satisfy their design objectives.

Agents in Bang

- **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

Bang as a middleware

- **support for agents life-cycle:** creation, migration, persistence,
- **communication:** message encoding, delivery
- **resource allocation:** memory, processor, disk
- **complexity analysis:** parallelization profiling
- **airport** on each computer, TCP/IP
- **agent granularity:** monolithic system / 1 or more threads per agent / processes
- **user interface**

Bang as a software

- written in C++, gcc 3.x,
- POSIX, curses, X, Tcl/Tk, prolog, PAPI
- runs on Linux, SGI, Solaris, CygWin
- base code : 0.6MB of C++
- agents: .3MB of augmented C++
- custom data types (XML-izable)
- in house memory management (Objective C-like)

PART II

- Agents and MAS
- Agents that socialize
  - agent communication language
  - messages, gates and interfaces
  - multiagent schemes
  - ontologies
- Agents that are clever
- Agents that evolve

Agent Comm. Language

- superior to e.g. RPC/RMI/CORBA (actions or propositions with **semantics** rather than just object, **declarative** rather than method invocation)
- **message layer:** sender, recipient, subject, conversation id
- **communication layer:** qeury, inform, request
- **content layer:** encoded neural network, what time is it?

ACL in Bang

- message and communication layer based on FIPA ACL (based on KQML)
- XML instead of LISP
- content layer inspired by DMG PMML and Caltec XSIL
- support for building, parsing, catching the messages
- synchronous/asynchronous message sending

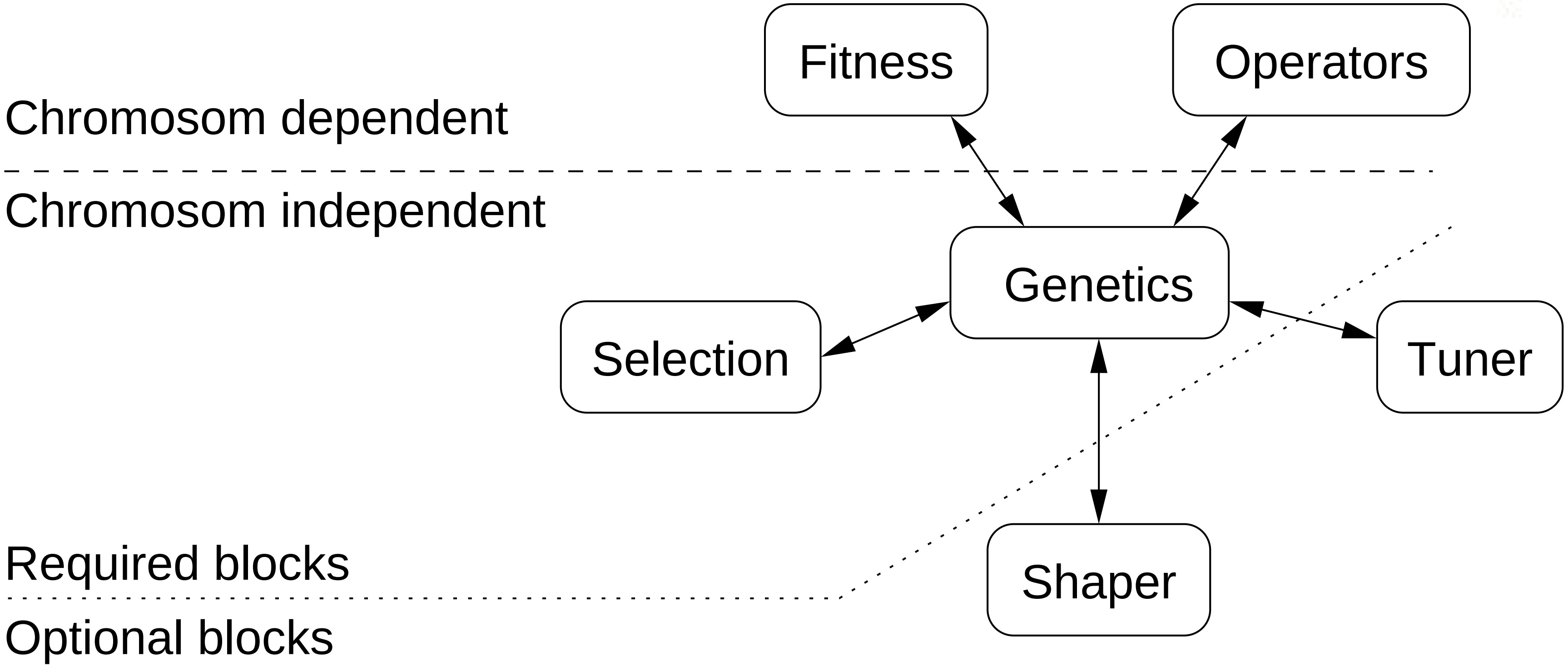
Gates and interfaces

In order to connect agents into MAS, define:

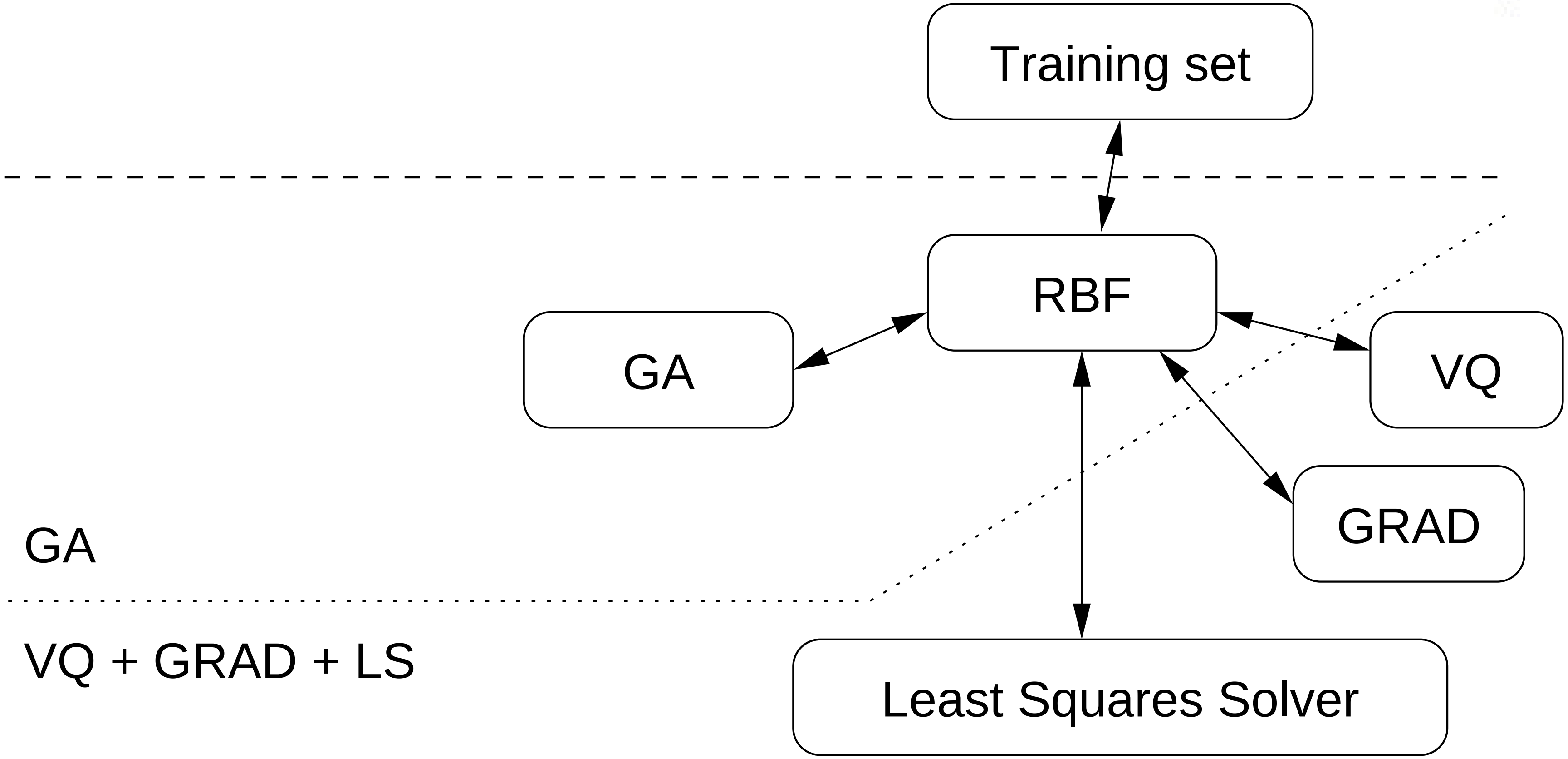
- **gate:** channel for outgoing messages
- **interface:** channel for incomming messages
- their **types:** named set of messages with clear semantics (data source communication, computation control, GUI,. . .)

Then, **MAS scheme** is set of agents with defined connections (and some gates/interfaces to the outer world).

Example: GA as MAS



Example: RBF as MAS



Ontologies

[T.Gruber: An ontology is a specification of a conceptualization.]

- agreement to use a vocabulary (i.e., ask queries and make assertions)
- agents commit to ontologies, can share knowledge
- hiearachy of agents, gates/interface types, tasks, agent properties
- description logics formalism (basis for DAML+OIL)

Ontologies example

```
atomic_concept(’igData’);
atomic_concept(’requestData’); % init/open/close/rewind/get info/next/
message_type(’igData’, ’requestData’);
atomic_concept(’DataSource’);
  interface(’DataSource’, ’igData’);
atomic_concept(’DataSourceConsumer’);
  gate(’DataSourceConsumer’,’igData’);
atomic_concept(’IterativeComputation’);
IterativeComputation is Computation;
interface(’IterativeComputation’,’igIterativeCompControl’);
gate(’IterativeComputation’,’igIterativeToMonitor’);
hide(’IterativeComputation’,’igToMonitor’);
atomic_concept(’aRbfNetwork’);
aRbfNetwork is NeuralNetwork and IterativeComputation
  and classInBang and SimpleTaskManager and Father;
  gate(’aRbfNetwork’,’igSolveRepresentatives’); % ALloyd VQ
  hide(’aRbfNetwork’,’igCommonCompControl’);
  gate(’aRbfNetwork’ ,’igSolveLinEqSystem’);    % LinearSystemSolver
```

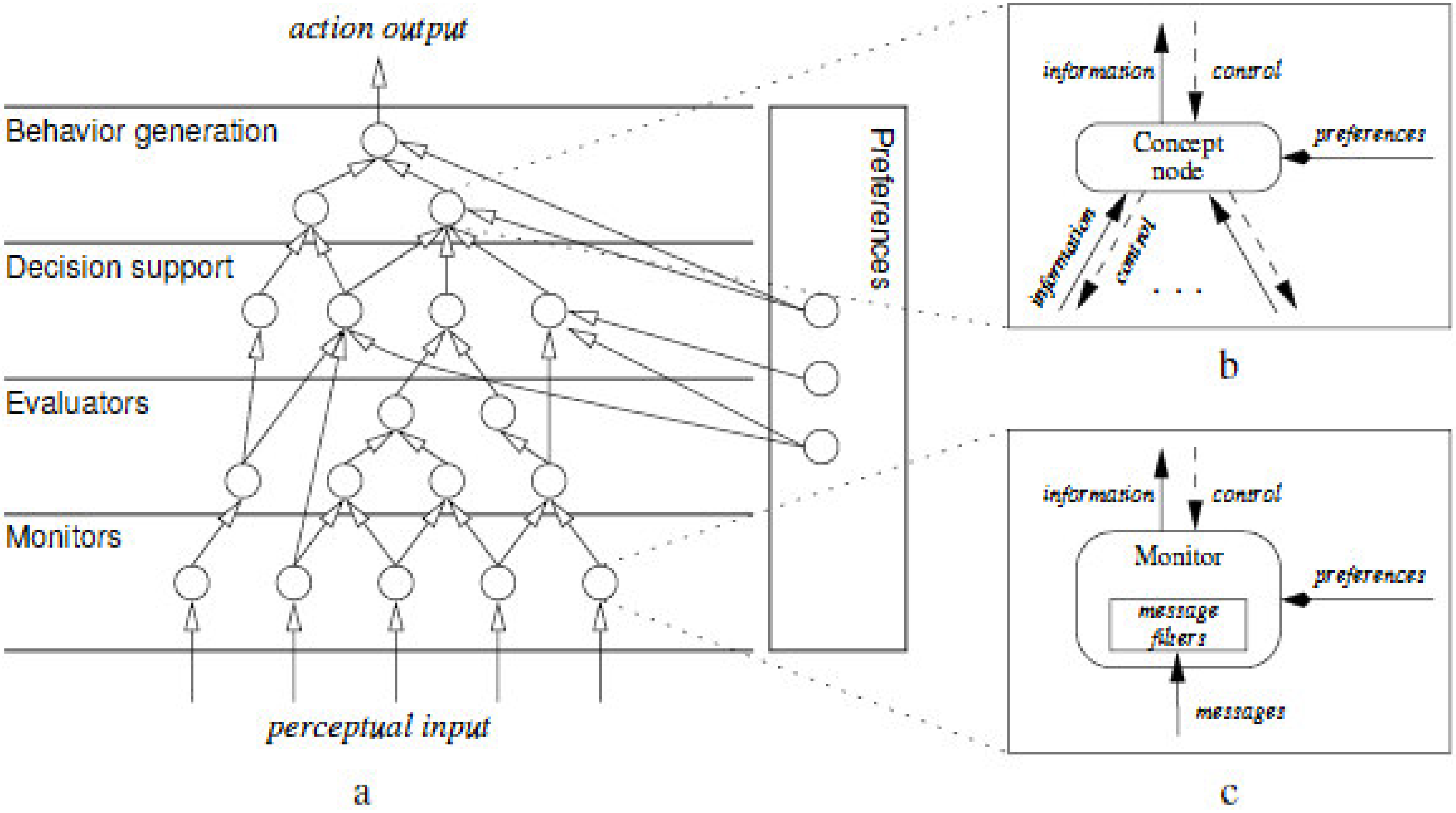
PART III

- Agents and MAS
- Agents that socialize
- Agents that are clever
  - decission support for an agent
  - accept/reject computations
  - cooperation, pro-activness
  - BDI architecture
- Agents that evolve

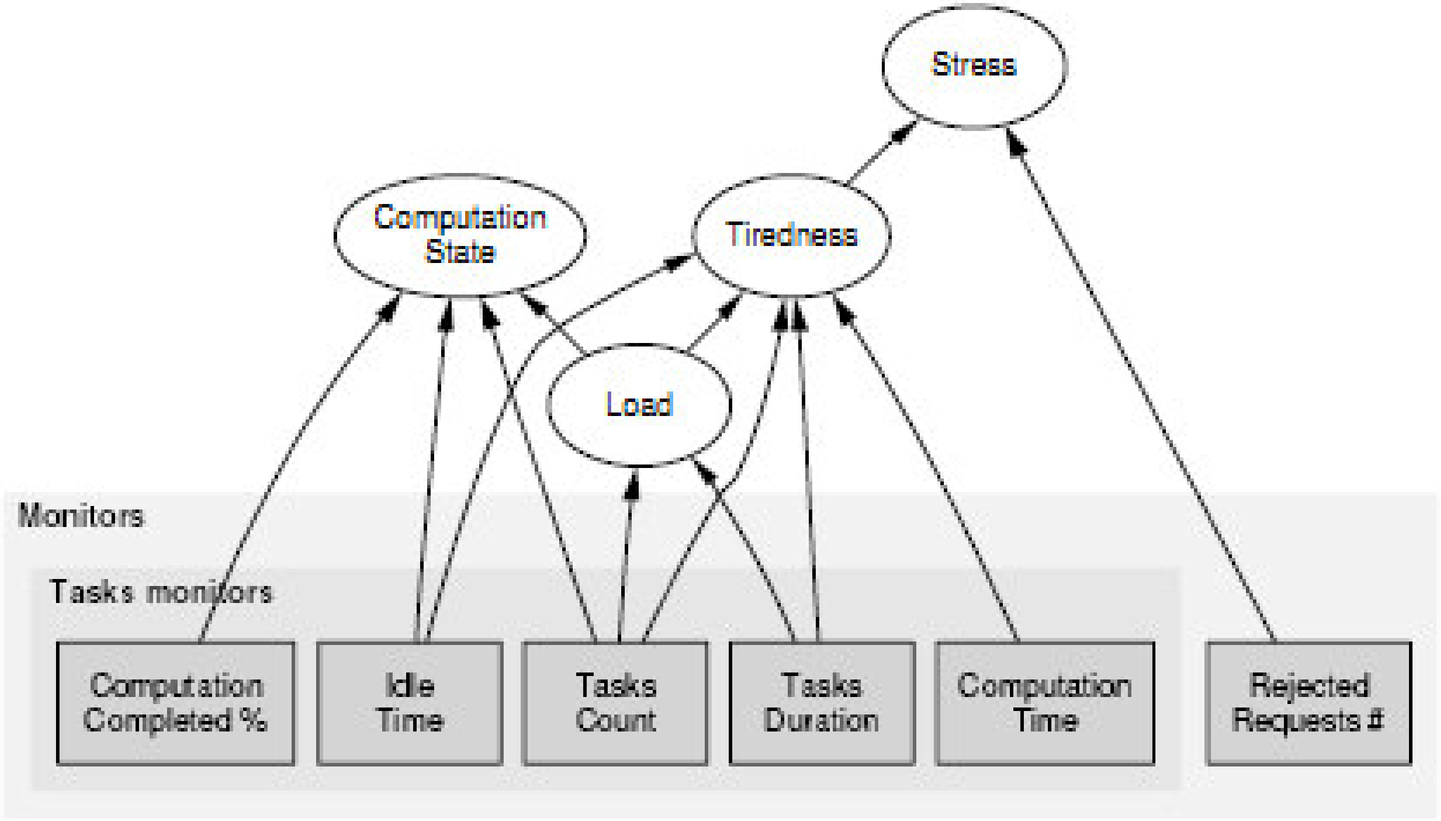
Intelligent agents

- additional brain (not necessary)
- eavesdropping all agent conversation
- internal model of agent state, ...
- can provide advices to agent
- decission support in cooperation, task acceptance
- generation of agent behavior, plans, ...
- adaptive

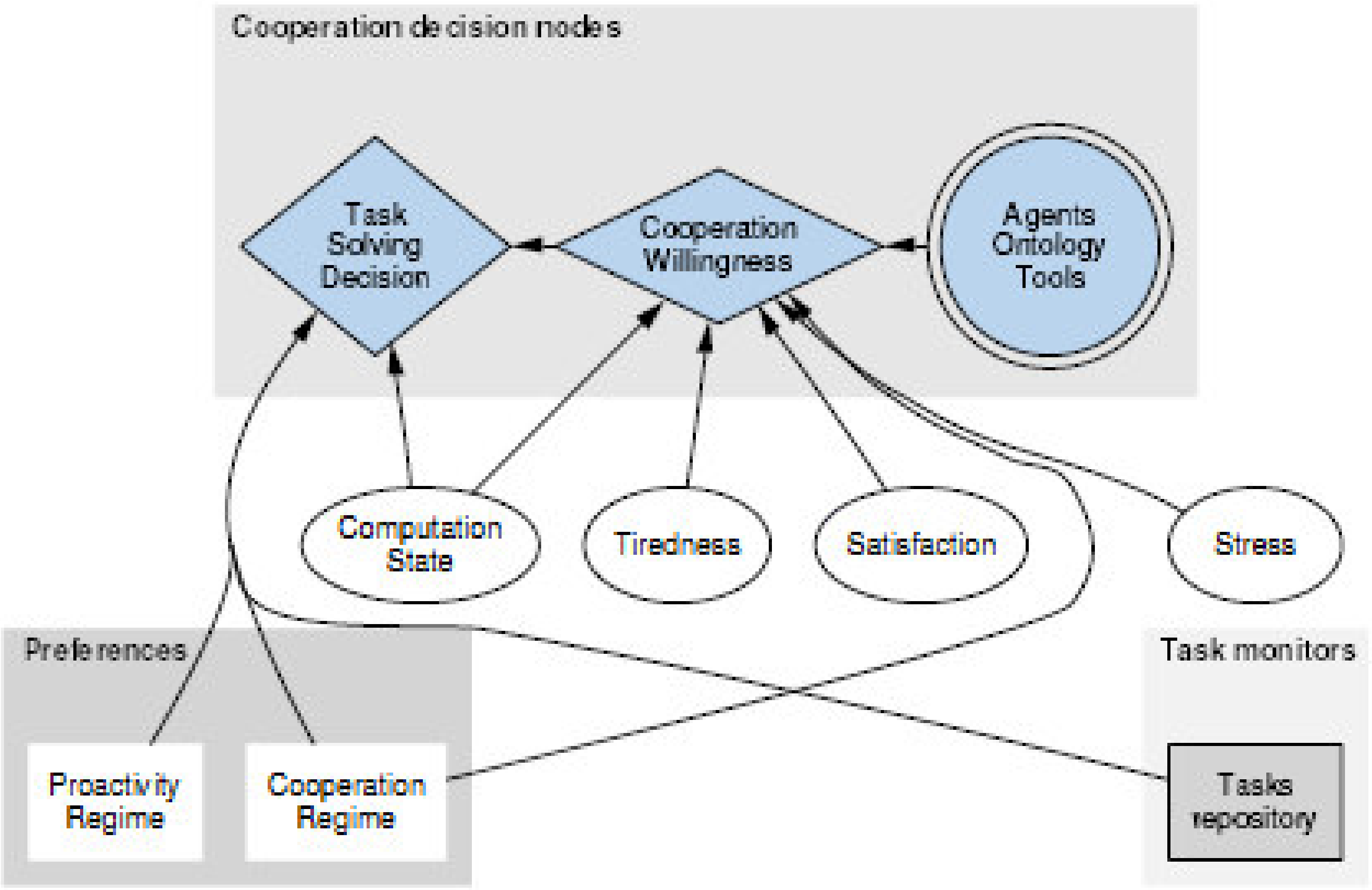
Network of concepts



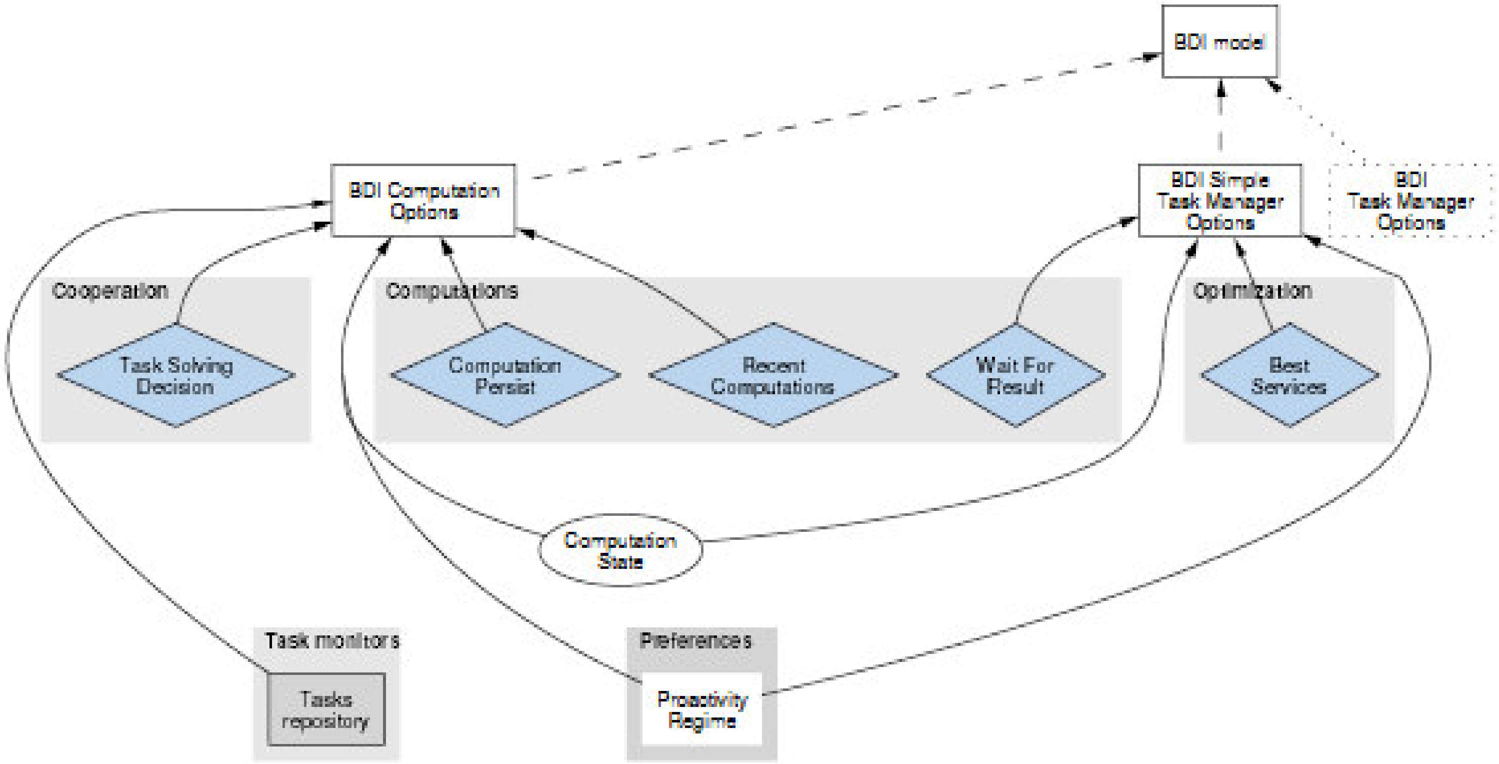
State of agent



Cooperation support



BDI

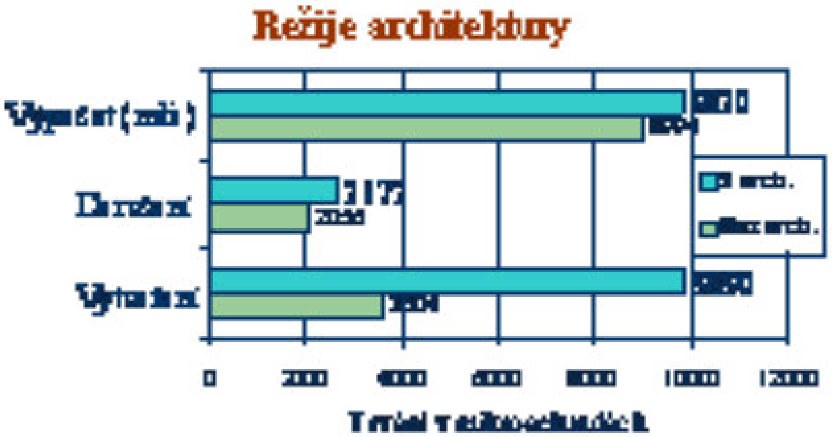


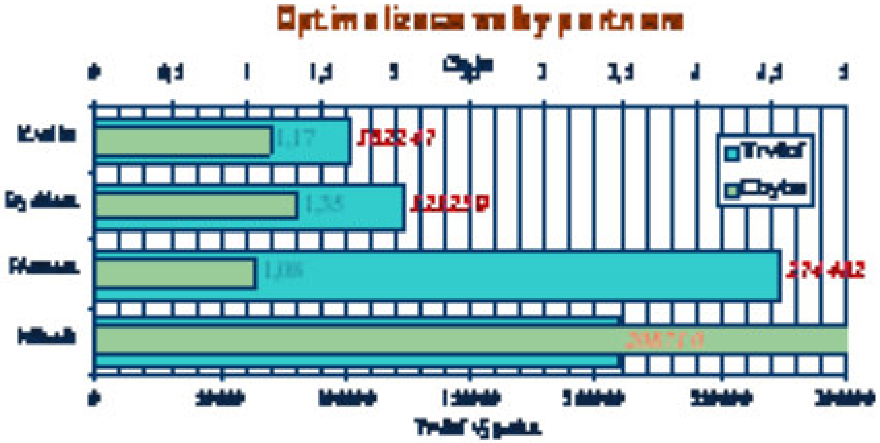
accept/reject/find missing info/search for new info

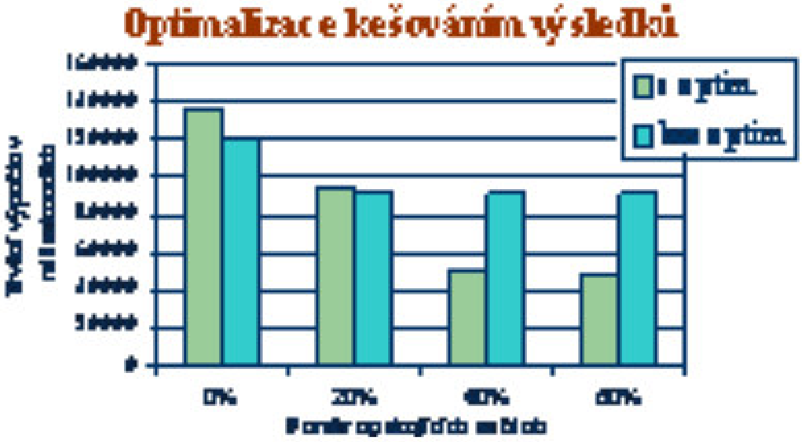
Example

```
Percy(356158647): Set to be cautious
Manager1(356158889): Set to be very persistent
Manager2(356158946): Set to be very cautious
Manager1(356158972): Assigning task 0 to MLP
Percy(356158976): Asking BDI what to do...
Percy(356159038): Computation accepted
Manager1(356159043): Training started, task 0
Manager2(356159043): Assigning task 0 to MLP
Percy(356159046): Asking BDI what to do...
Percy(356159104): Sorry, busy
Manager2(356159106): OOPS, we were rejected. We have to try again: 0
Manager2(356159110): Assigning task 0 to MLP
Percy(356159112): Asking BDI what to do...
Percy(356159170): Sorry, busy
Manager2(356159171): OOPS, we were rejected. We have to try again: 0
Manager2(356159183): Assigning task 0 to RBF
Manager2(356159193): Training started, task 0
Manager2(356159576): Task was finished:
```

Brain helps


Režije architektury




Optimalizace kešováním výsledků

PART IV

- Agents and MAS
- Agents that socialize
- Agents that are clever
- Agents that evolve
  - evolutionary algorithm for MAS schemes
  - reasoning about MAS
  - hybrid search algorithm
  - sci-fi

Evolution of schemes

- MAS Scheme – a directed acyclic graph
- EA similar to Koza's genetic programming:

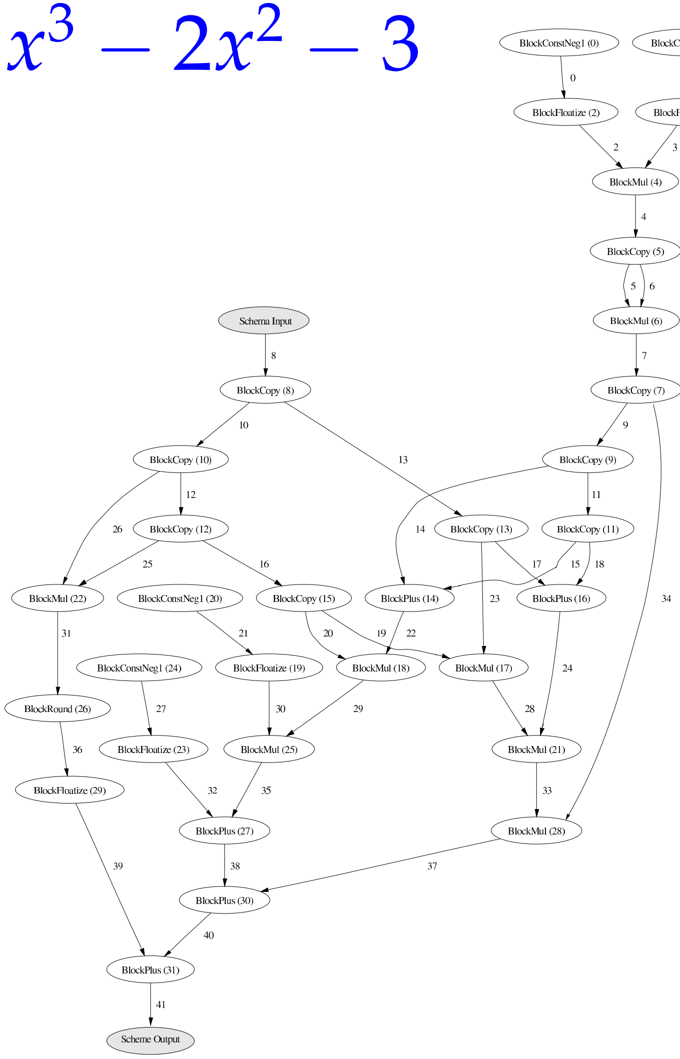
```
♦ randomly create the initial population pop
♦ do {
    • foreach g∈pop
        · create the scheme, run and evaluate
          its fitness
    • using selection and genetic operators
      generate new population
  } until fitness<desired
```

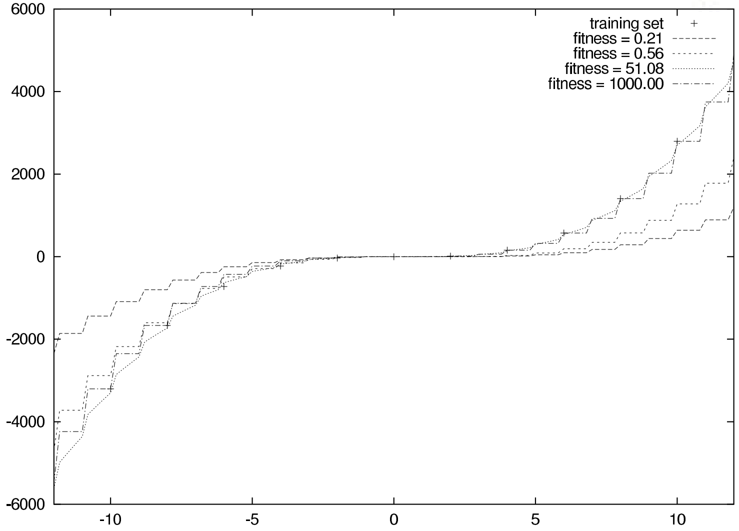
Experiments with evolution I

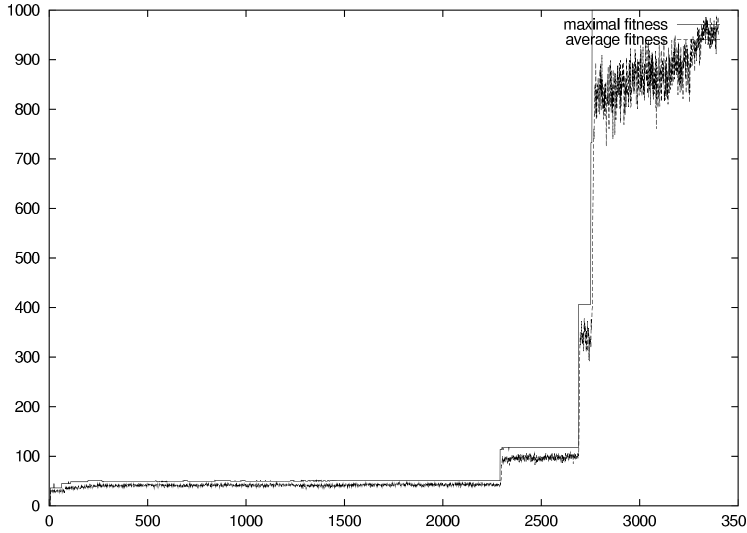
- **Evolving arithmetical functions**
  - $2x + 1$
  - 0
  - $x^3 - 2y - 3$
  - $\ldots$
- **Success depends on**
  - Function complexity
  - Initial population
  - Operator set and their parameters ($x^2 + y^2$ vs. $x^2 + y^2 + 1$)

Experiments with evolution II

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







Evolving the "real" MAS

- **automatically** solve a given problem (data)
- consisting of **agents** like NNs, GAs, FLCs, filters, data sources, visualizers, . . .
- requires a lot of computational power
- **ontologies:**
  - hierarchies of agent types
  - their roles
  - their interfaces
- combining EA with **logical reasoning**

Logical reasoning about MAS

Why use logical reasoning?

- **Sanity check:** Sort out non-functioning systems during EA without having to actual construct and test them.
- **Fault Analysis:** Isolate non-working parts of a system, or parts that do not satisfy the constraints.
- **System Construction:** From an incomplete description, generate a MAS that satisfies the constraints

Declaring Agents

An **agent** is defined by…

- the agent's **properties**
- **constraints** on these properties

Agent:	DecisionTreeAgent
Properties:	ComputationalAgent, Trainable, hasGate(Input), hasGate(Output)
Constraints:	connectedTo(Input, I), DataSource(I), connectedTo(output, O), DataSink(O)

Declaring MAS

- A MAS consists of agents and **global constraints** that define required properties of the MAS as a whole.
- **Type of MAS:** MAS must contain a computational agent and a GUI agent connected to it.
- **Validity of configuration:** For all connections between agents, the input and the output gate must match.
- **Trust:** All agents must trust the agents they connect to.

Evolution is cool

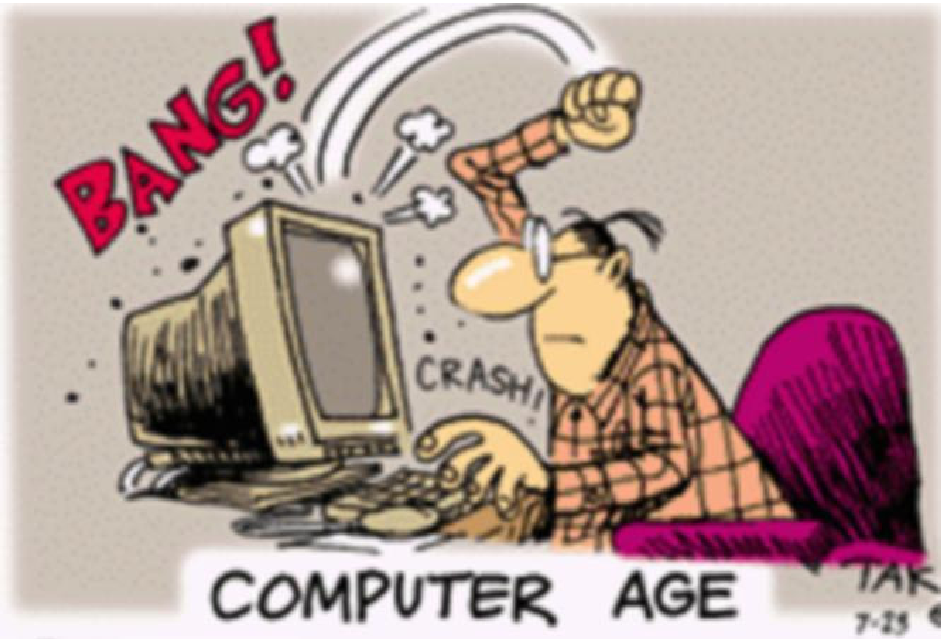
- Given just the task description – usually in the form of a data set,
- and using tools we already have available: ontology services, reasoner, EAs, MASman, bunch of computational agents,
- we can "automatically" search for solutions – hybrid models of the task,
- expressed as MAS schemes,
- and evaluate their performance, etc.

Conclusions

- No one universal solution to all problems.
- Theory provides worst/best case scenarios, but it’s the gray zone between we live in.
- Custom, possibly hybrid solutions:
    - talk to other agents,
    - gather experience, reason,
    - evolve solutions.
- Bang might help with this.

TODO:

- going WWW: html/http GUI, ...
- connection to Racer, KR-Hyper, ...
- FIPA-ACL interface, Agentcities, ...



Credits

bang.sf.net

- Prague: P. Krusina, P. Kudova, P. Rydvan, R. Vaculin, P. Soxac
- Koblenz: G. Beuster, A. Sinner
- Nimes: D. Pearson
- Chico: R. Renner, J. T. Stimatze