Extended Finite-State Machine Inference with Parallel Ant Colony Based Algorithms

<u>Daniil Chivilikhin</u>
PhD student
ITMO University

Vladimir Ulyantsev
PhD student
ITMO University

Anatoly Shalyto
Dr.Sci., professor
ITMO University

BIOMA @ PPSN’14

September 13, 2014

Motivation: Reliable software

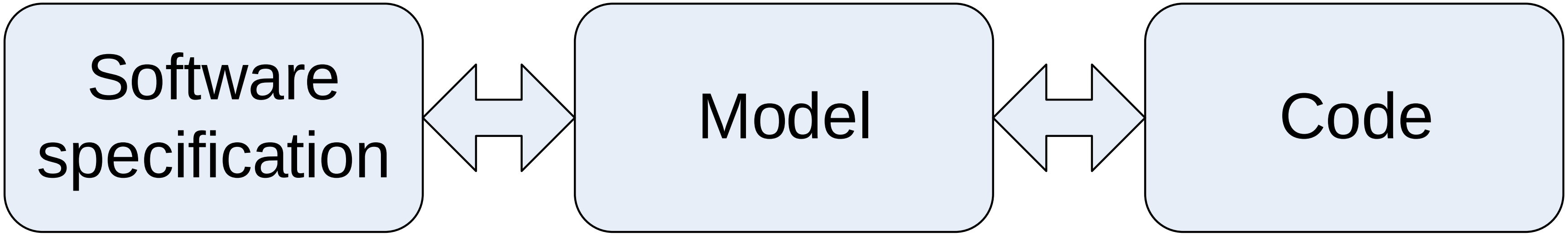
- Systems with high cost of failure
  - Energetics
  - Aerospace
  - Finances
  - …
- We want to have **reliable software**
  - Testing is not enough
  - **Verification** is needed

Challenge

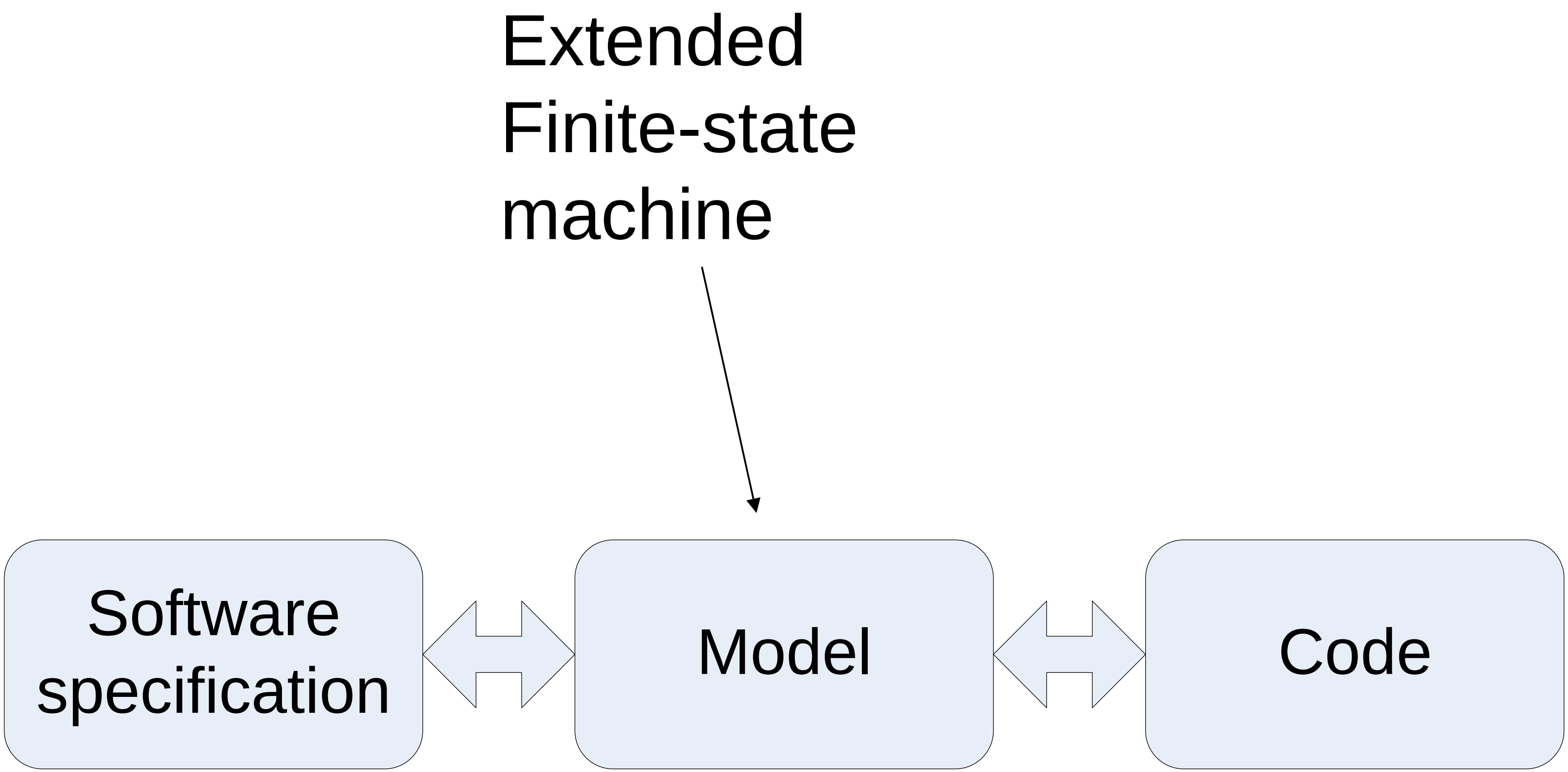
- Reliable systems are hard to develop
- Verification is time consuming

Model-driven development

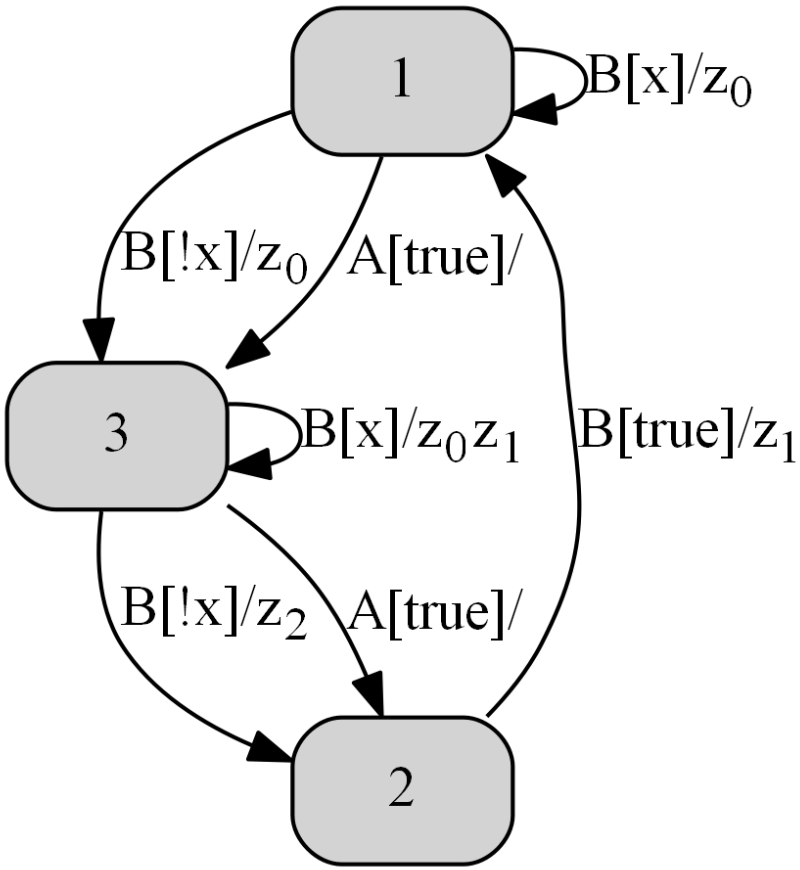
- Automated software engineering
- Model-driven development

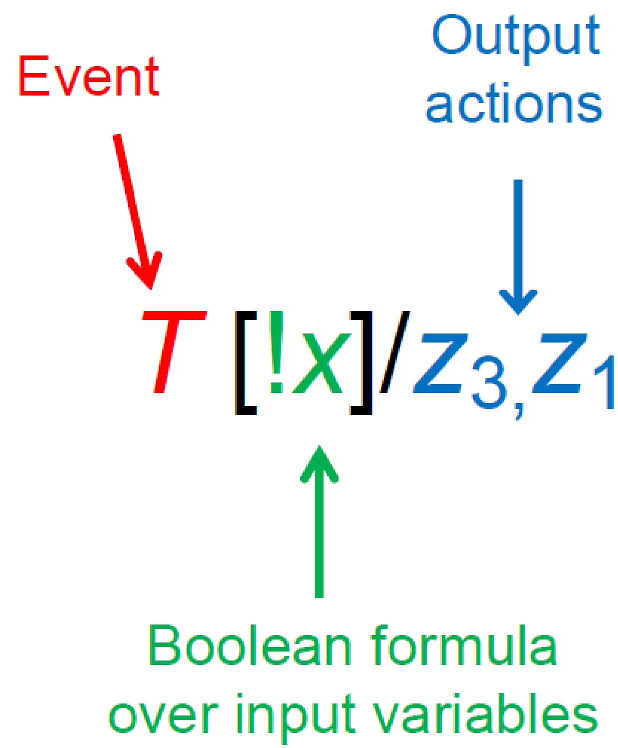


Automata-based programming

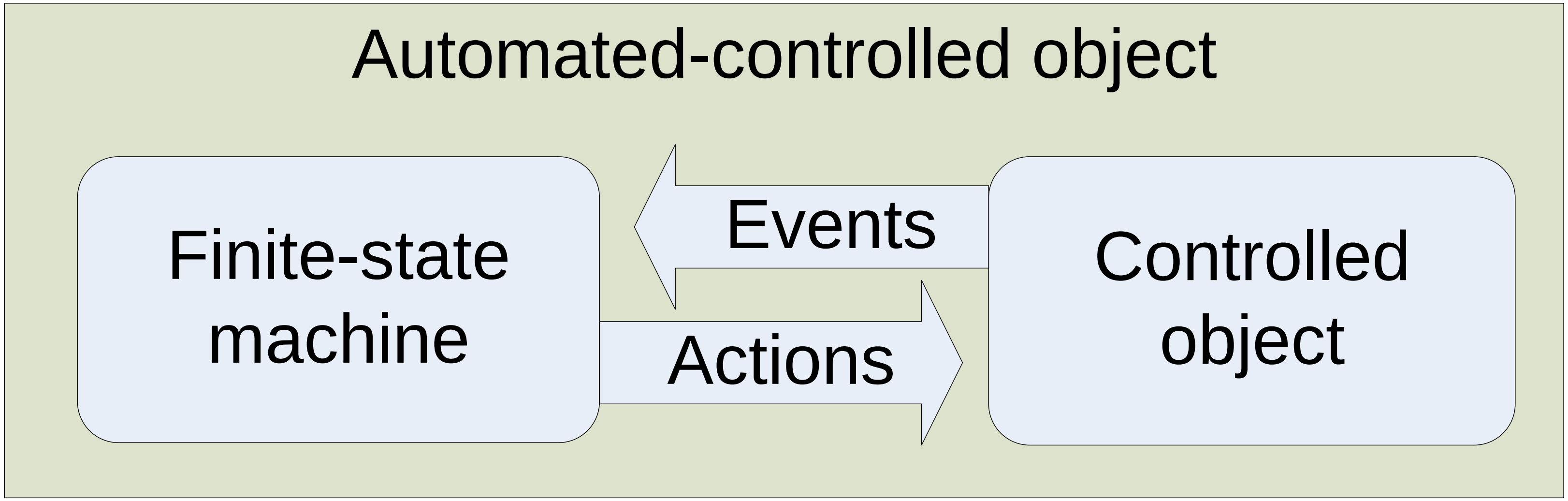


Extended Finite-State Machine





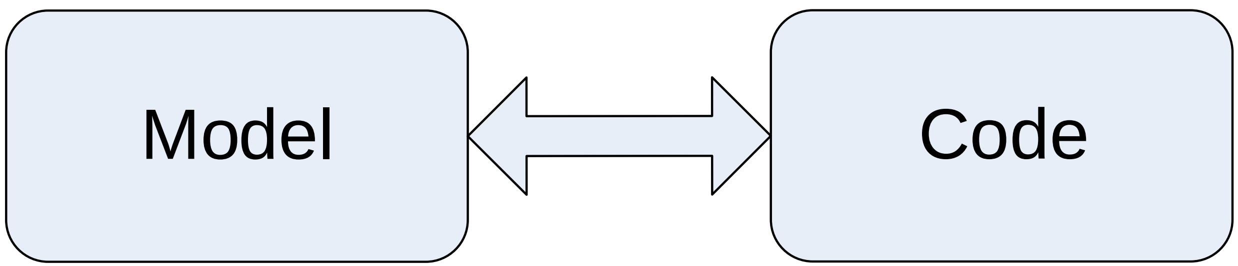
Automata-based programming



Automata-based programming: advantages

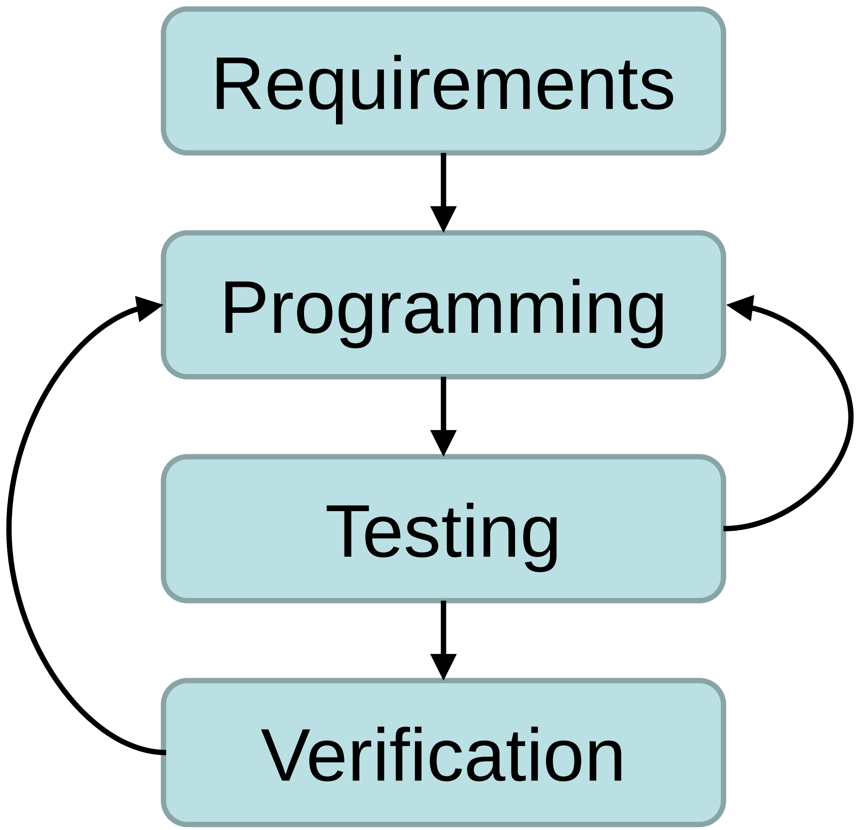
- Model before programming code, not vice versa

Finite-state machine

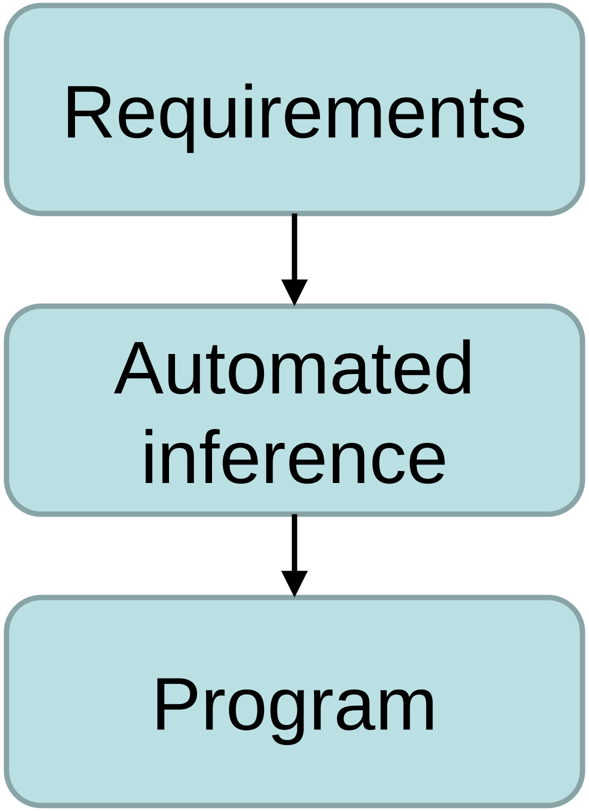


- Possibility of program verification using *Model Checking*

Conventional workflow



Automata-based programming workflow



- ✓ Easy for the user
- ✓ Time-consuming for computer

Issues

- Hard to build an EFSM with desired behavior
- Sometimes, several hours on a single machine
- Use parallel algorithms

EFSM inference algorithms

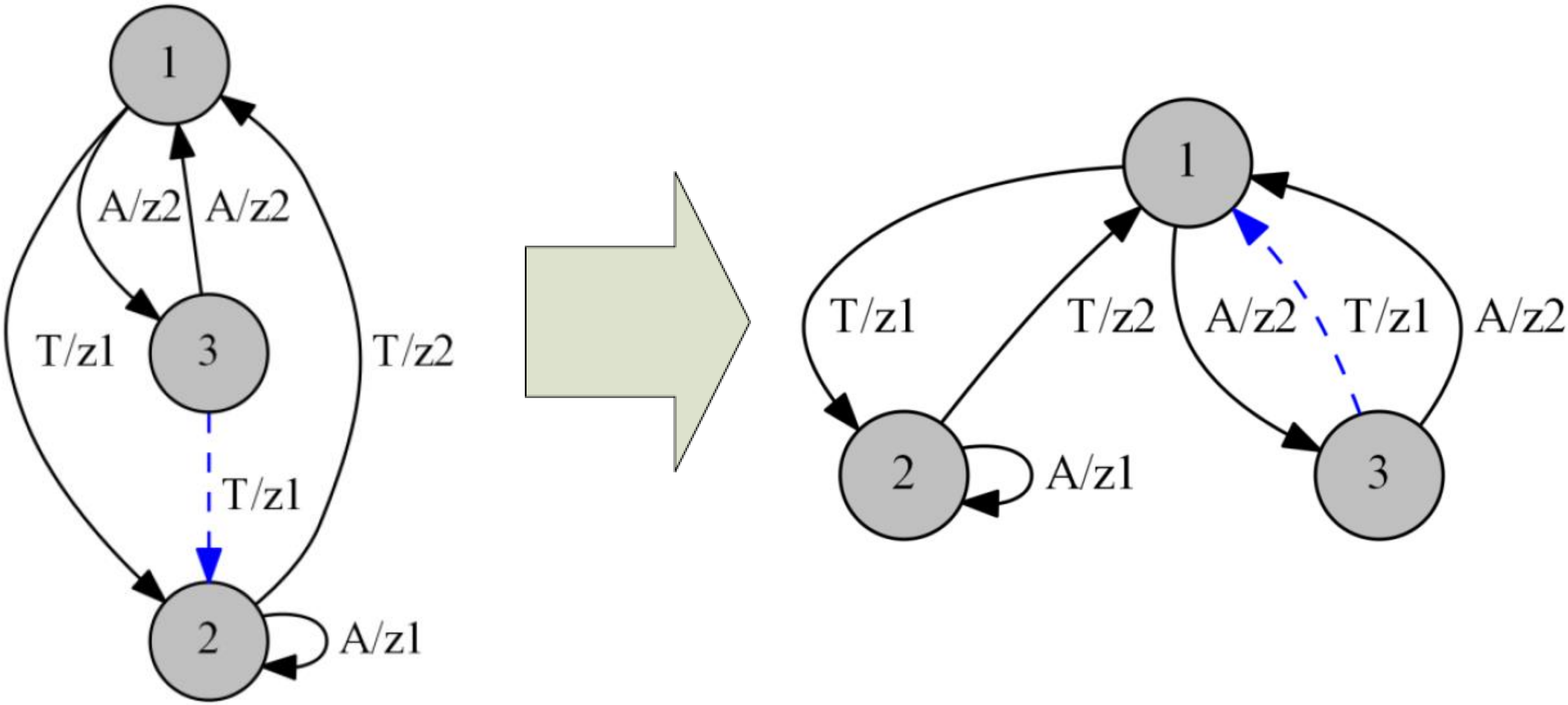
- Genetic algorithm (GA)
- Previous work: Mutation-based Ant Colony Optimization (MuACO)
- …

- No parallel implementations so far

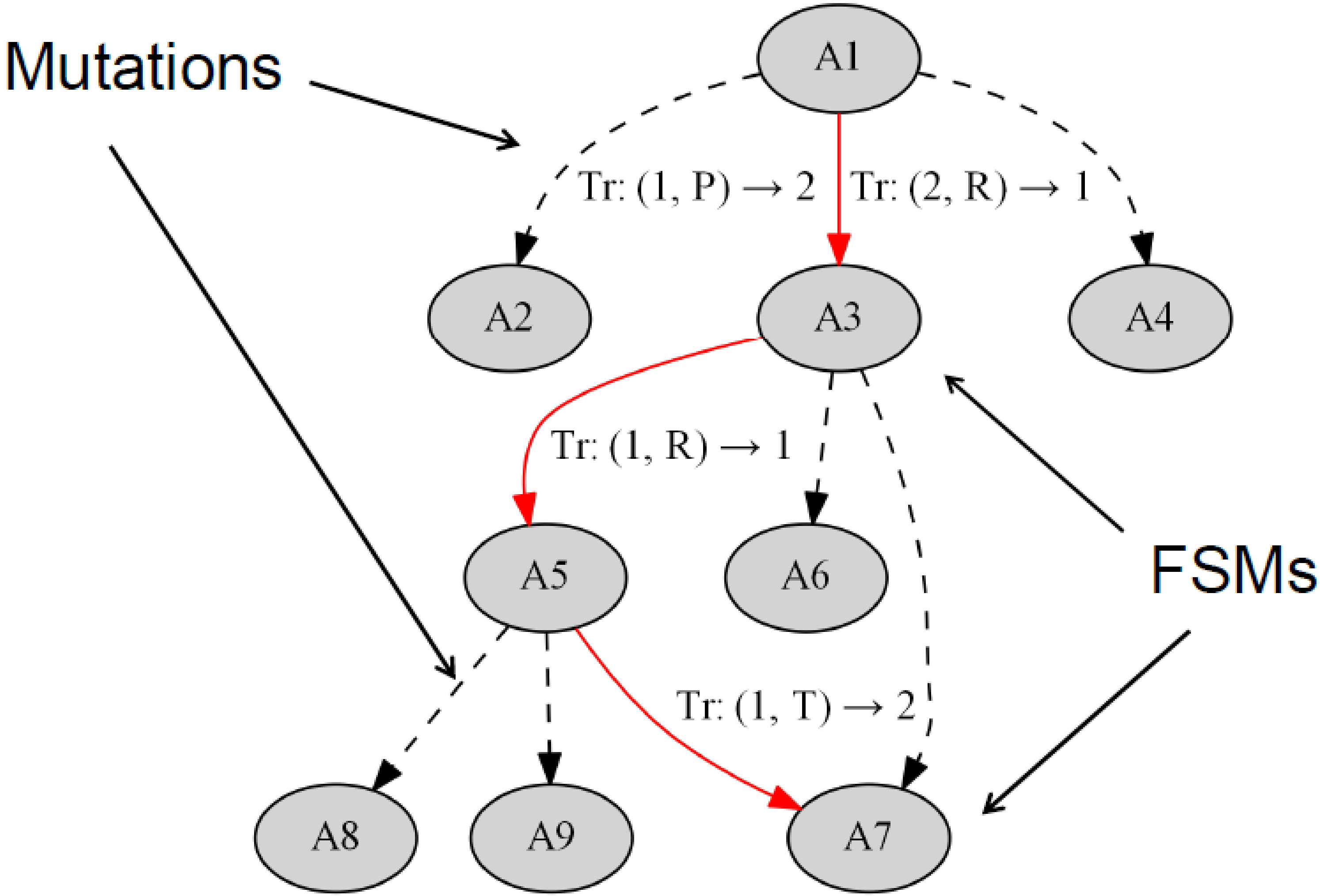
In this work

- Develop several parallel versions of MuACO
- Compare
  - With each other
  - With parallel GA
  - Statistical significance

EFSM mutations



MuACO algorithm



MuACO algorithm

A_0 = random FSM

Graph = $\{A_0\}$

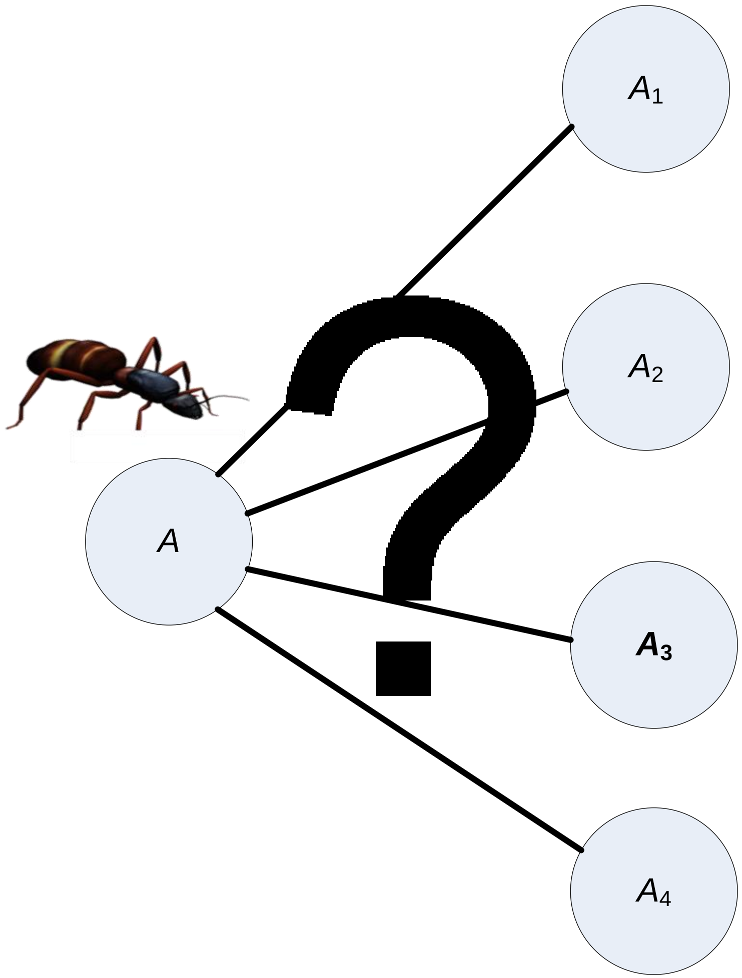
while not stop() do

    ConstructAntSolutions

    UpdatePheromoneValues

Constructing ant solutions

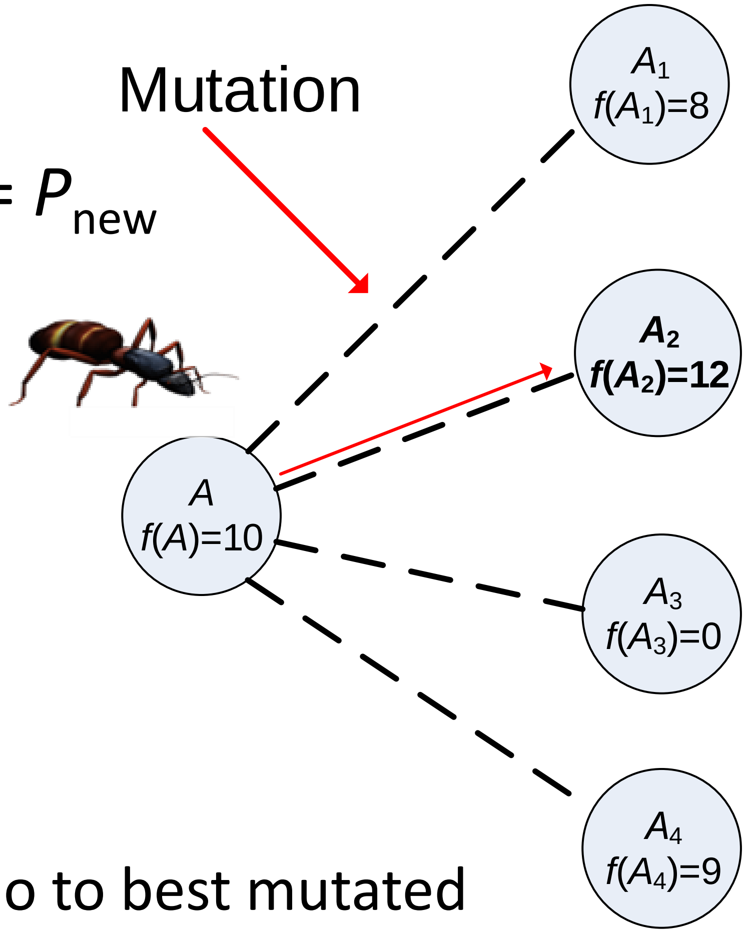
- Use a colony of ants
- An ant is placed on a graph node
- Each ant has a limited number of steps
- On each step the ant moves to the next node



Ant step: selecting the next node

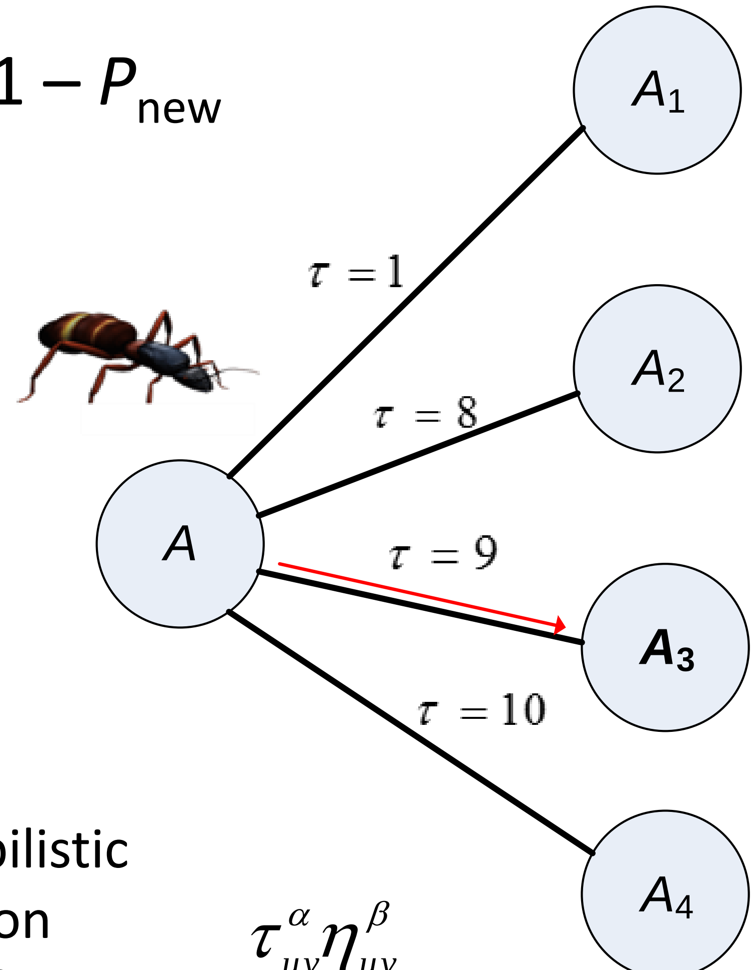
Mutation

$P = P_{new}$



Go to best mutated FSM

$P = 1 - P_{new}$



Probabilistic selection

$$p_{Av} = \frac{\tau_{uv}^{\alpha} \eta_{uv}^{\beta}}{\sum_{w \in \{A1, A2, A3, A4\}} \tau_{uw}^{\alpha} \eta_{uw}^{\beta}}$$

Why parallel MuACO?

- Single-node MuACO is more efficient than GA for EFSM inference
  - Chivilikhin D., Ulyantsev V. MuACOsm - A New Mutation-Based Ant Colony Optimization Algorithm for Learning Finite-State Machines / In GECCO'13
  - Chivilikhin D., Ulyantsev V. Inferring Automata-Based Programs from Specification With Mutation-Based Ant Colony Optimization / In GECCO'14

Parallel combinatorial optimization

- Randomized algorithms
- More exploration – higher chance of finding optimal solution
- Increase exploration using parallelism

Parallel metaheuristics

- Evolutionary algorithms
  - Island scheme
  - Migration
  - MuACO doesn’t have a population
- Ant Colony algorithms
  - Multiple colonies
  - This can work

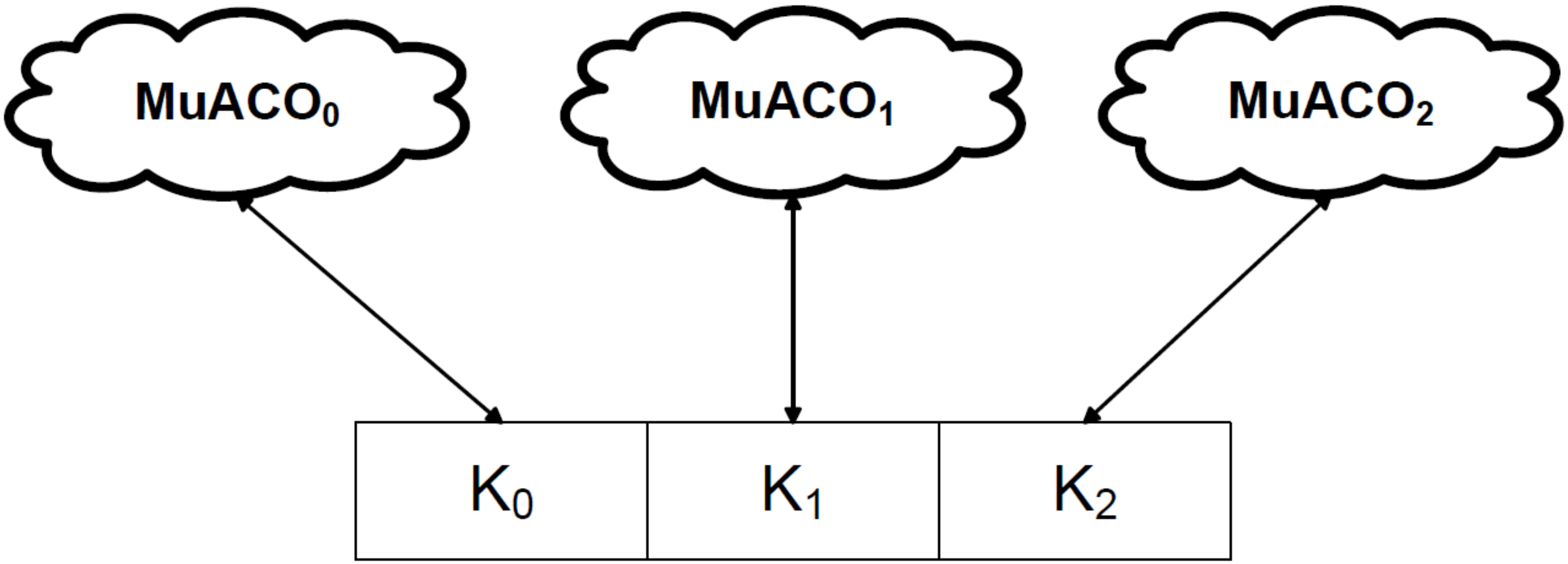
Three parallel MuACO algorithms

1. Independent parallel MuACO
2. Shared best solutions
3. MuACO with crossover

Independent parallel MuACO

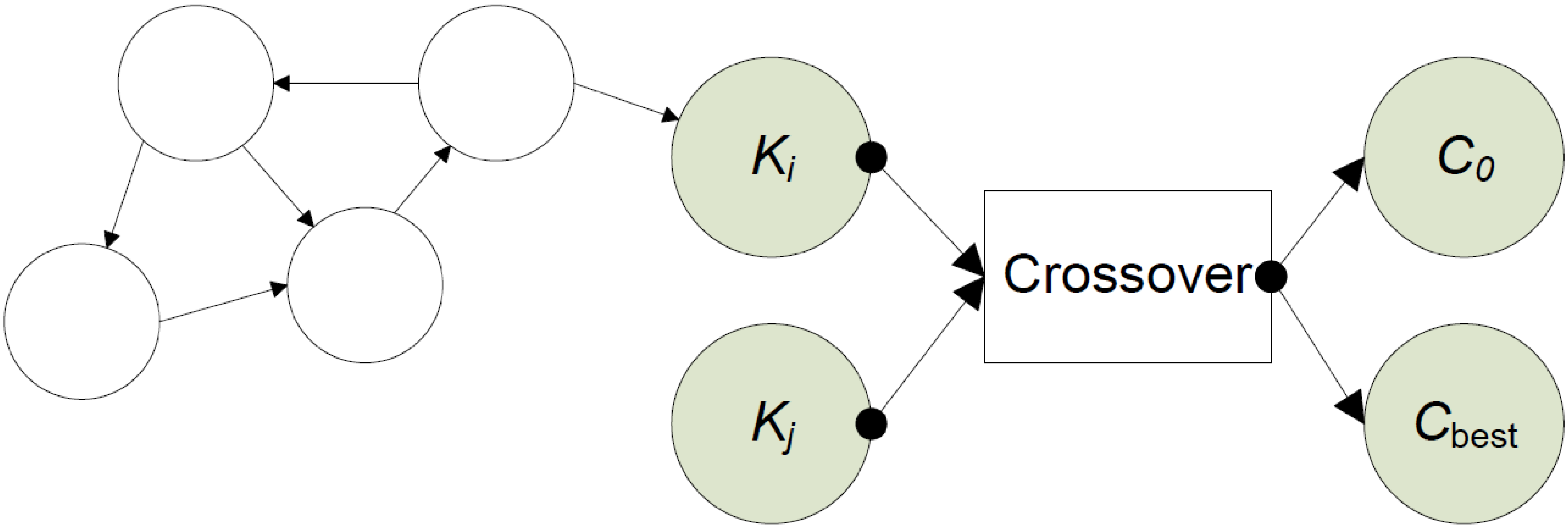
- *m* processors
- Generate *m* random initial solutions
- Start *m* MuACO algorithms
- Terminate when at least one finds optimal solution
- NO interaction between algorithms

Shared best solutions



- *i*-th algorithm restarts with *j*-th algorithm's best solution

MuACO with crossovers



Crossovers from: F. Tsarev and K. Egorov. Finite state machine induction using genetic algorithm based on testing and model checking. In GECCO'11 Companion Proc., pp.759–762, Dublin, Ireland, 2011.

Other tested approaches

- Parallel fitness evaluation
- Different algorithm settings
- …
- No good

Learning EFSMs from scenarios and temporal properties

Input data:

- Number of states *C*
- *S*et of test scenarios
- Set of temporal properties

Goal: build an EFSM with *C* states compliant with scenarios and temporal properties

Scenarios and temporal properties

- Scenario
  - T [x_1 & x_2]/z_1, A [true], A [x_2 & !x_1]/z_2, T [x_1]/z_3
- Temporal properties – Linear temporal logic
  - $\mathrm{G}(\mathrm{wasEvent}(T) \Rightarrow \mathrm{wasAction}(z_1))$

Learning EFSMs: Fitness function

- Pass inputs to EFSM, record outputs
- Compare generated outputs with references
- Use verifier to check temporal properties
- Fitness = string similarity measure (edit distance) + verification part

Experimental setup

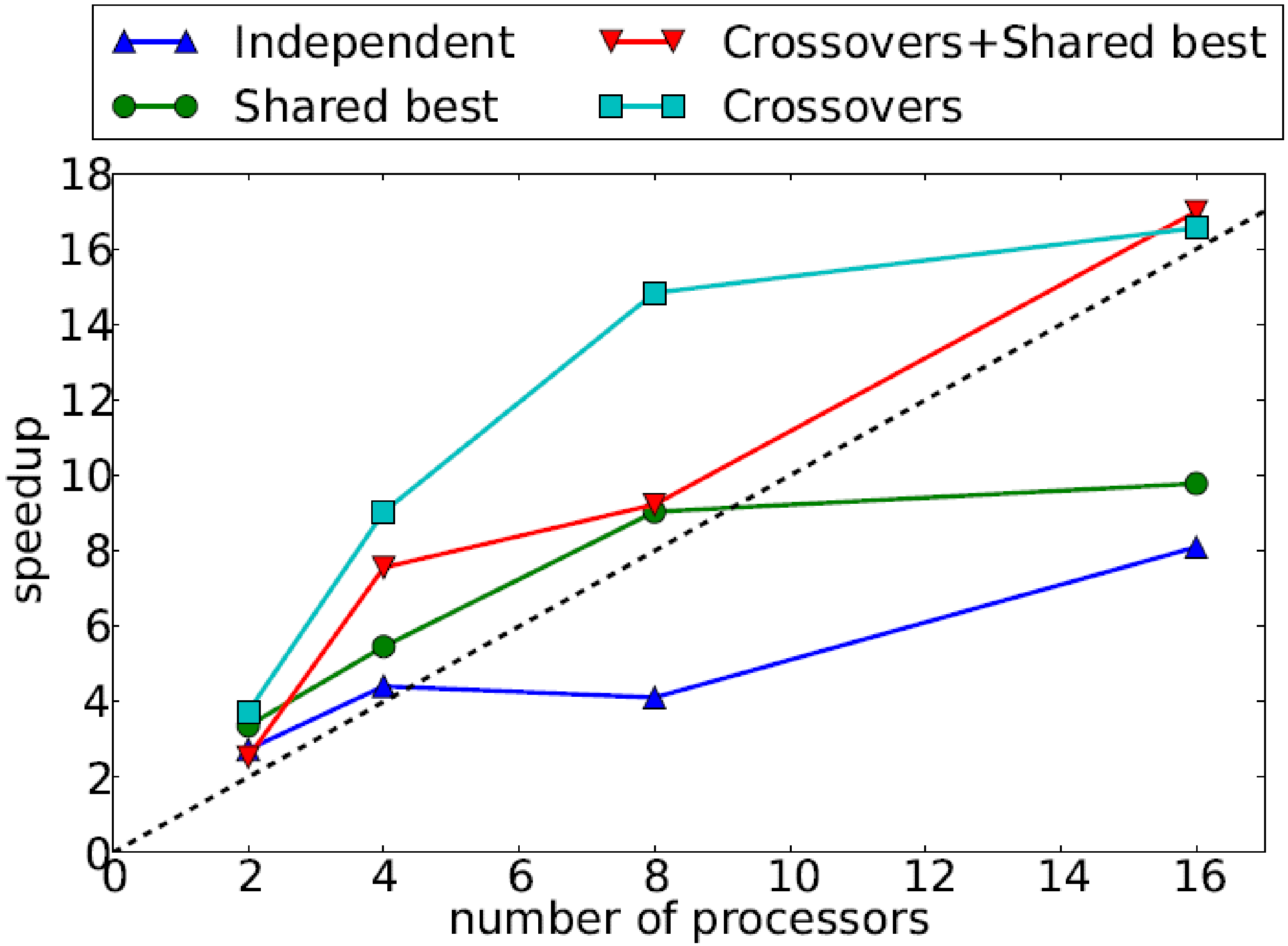
- 50 random EFSMs with 10 states
- One input variable
- Two input events
- Two output actions
- Sequence length up to 2

- 24-core AMD Opteron 6234 2.4 GHz processor

Compared algorithms

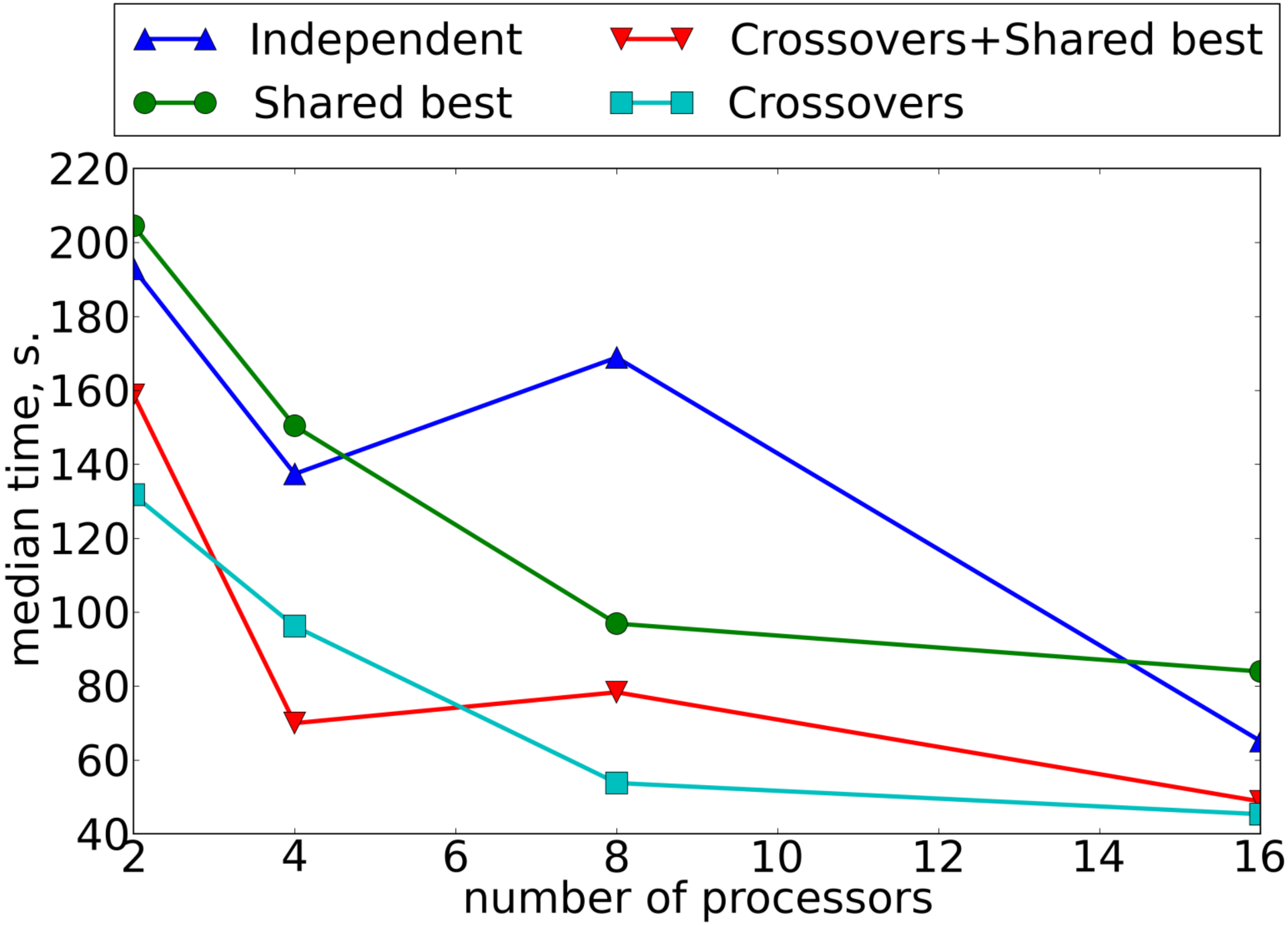
- Sequential MuACO
- Independent parallel MuACO
- Parallel MuACO + Shared best
- Parallel MuACO + Crossovers
- Parallel MuACO + Shared best + Crossovers
- Independent parallel GA

Results: MuACO speedup

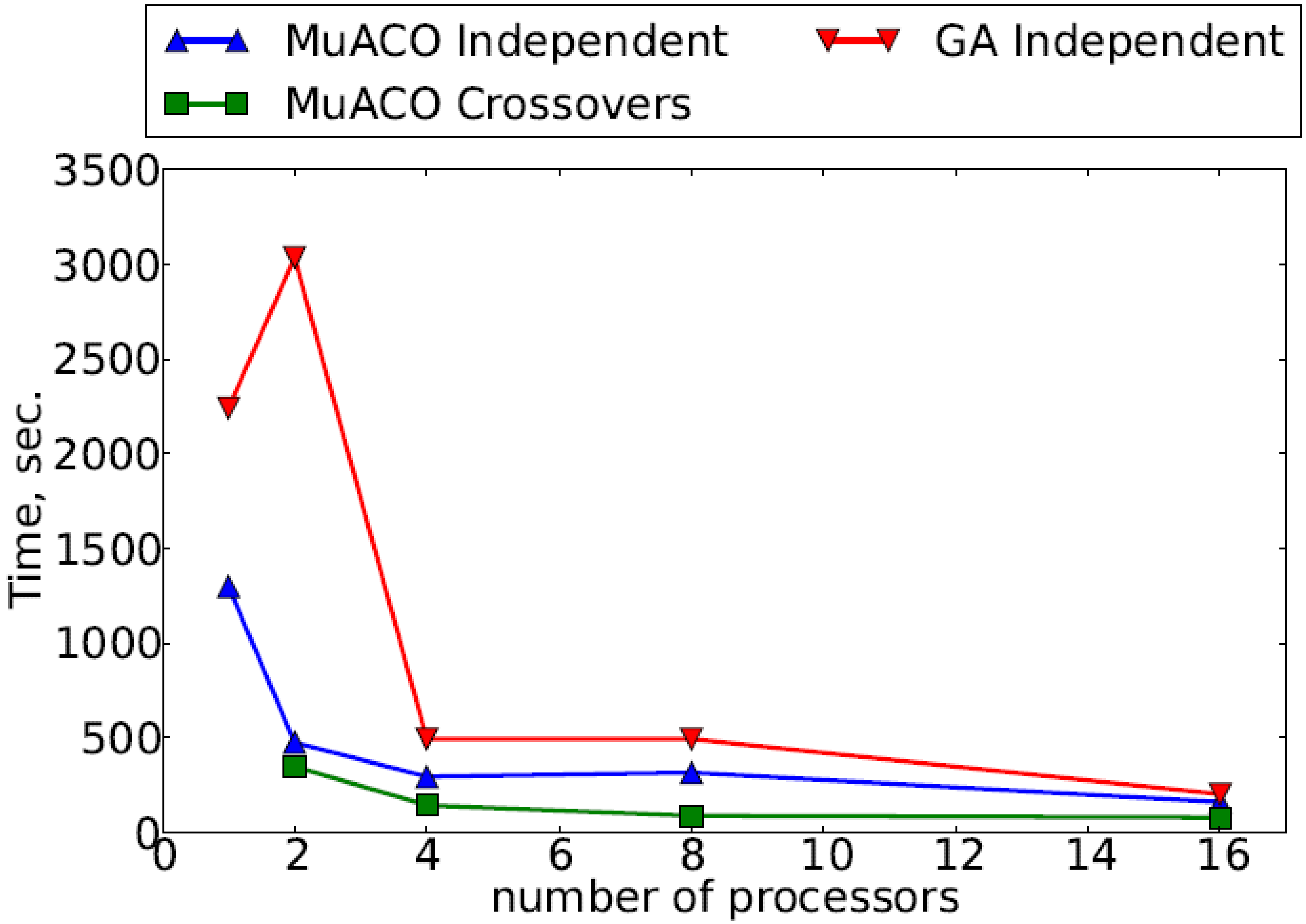


Sequential MuACO runtime = 1392 s.

Results: median time



Results: comparison with GA



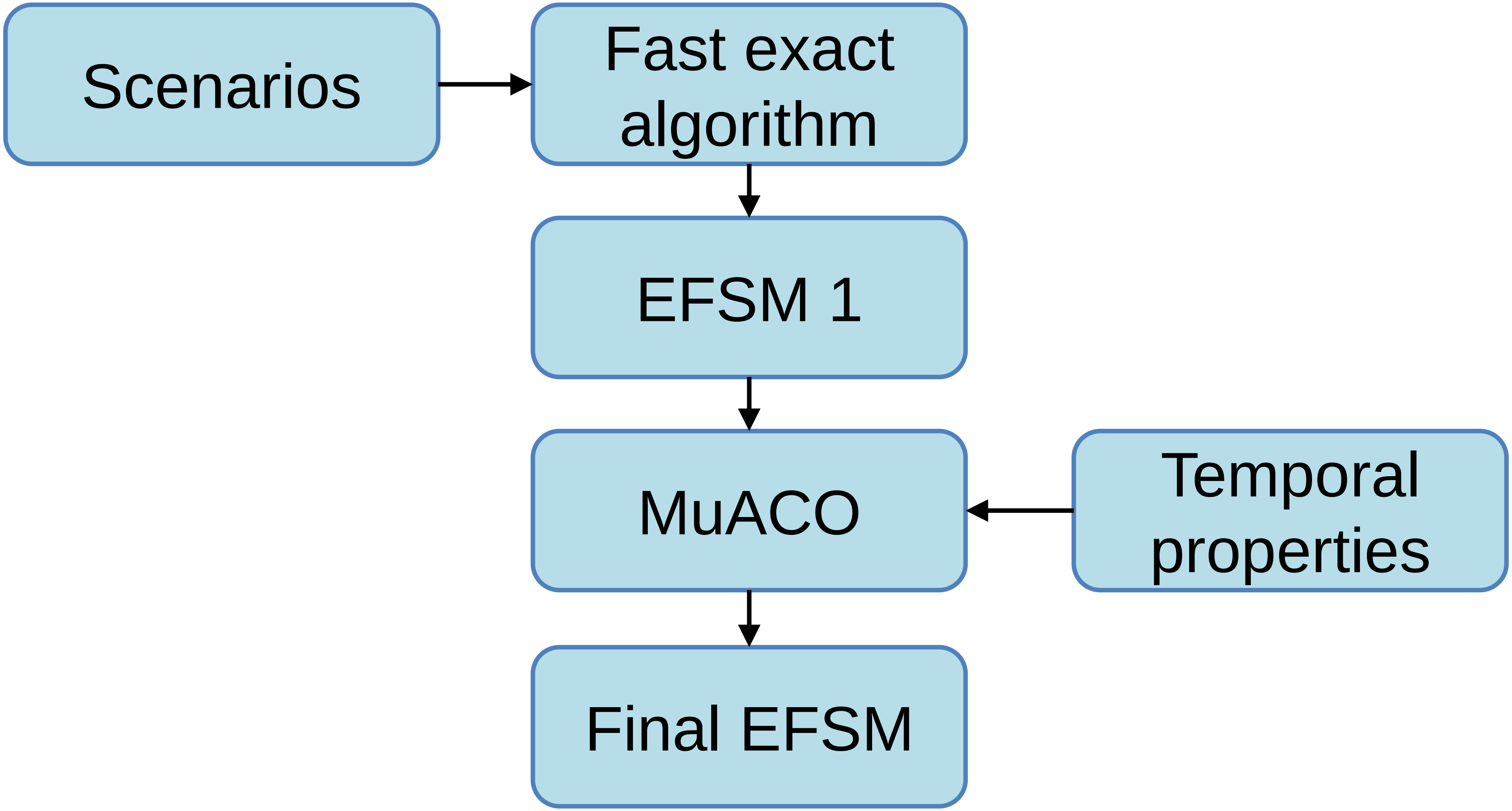
Statistical significance

- Both “Crossovers” are significantly better than other algorithms
- Not significantly different from each other

Combining exact and metaheuristic algorithms

- ICMLA’14: Combining Exact And Metaheuristic Techniques For Learning Extended Finite-State Machines From Test Scenarios and Temporal Properties (accepted)

Combining exact and metaheuristic algorithms



Combining exact and metaheuristic algorithms: results

	Crossovers	Exact + Crossovers
Mean time, s.	208	78
Median time, s.	73	28

Conclusion

- Parallel EFSM inference algorithms are very efficient
- Parallel MuACO algorithms with crossover demonstrated best performance
- With super-linear speedup

Future work

- Parallel MuACO-GA algorithm
- Experiments using more computational nodes
- More experiments with exact algorithms

Acknowledgements

- This work was financially supported by the Government of Russian Federation, Grant 074-U01, and also partially supported by RFBR, research project No. 14-01-00551 a.

Thank you for your attention!

Extended Finite-State Machine Inference with Parallel Ant Colony Based Algorithms

Daniil Chivilikhin
Vladimir Ulyantsev
Anatoly Shalyto
{chivdan,ulyantsev}@rain.ifmo.ru