Test-Based Extended Finite-State Machines Induction with Evolutionary Algorithms and Ant Colony Optimization

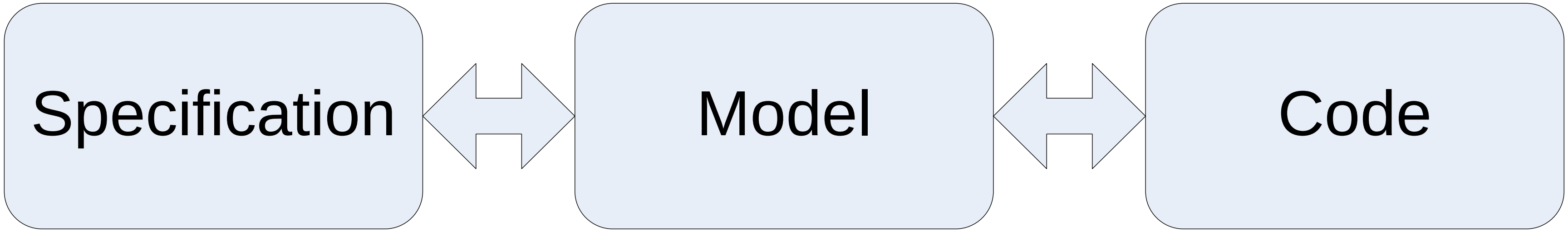
Daniil Chivilikhin, Vladimir Ulyantsev, Fedor Tsarev

St. Petersburg National Research University of Information Technologies, Mechanics and Optics

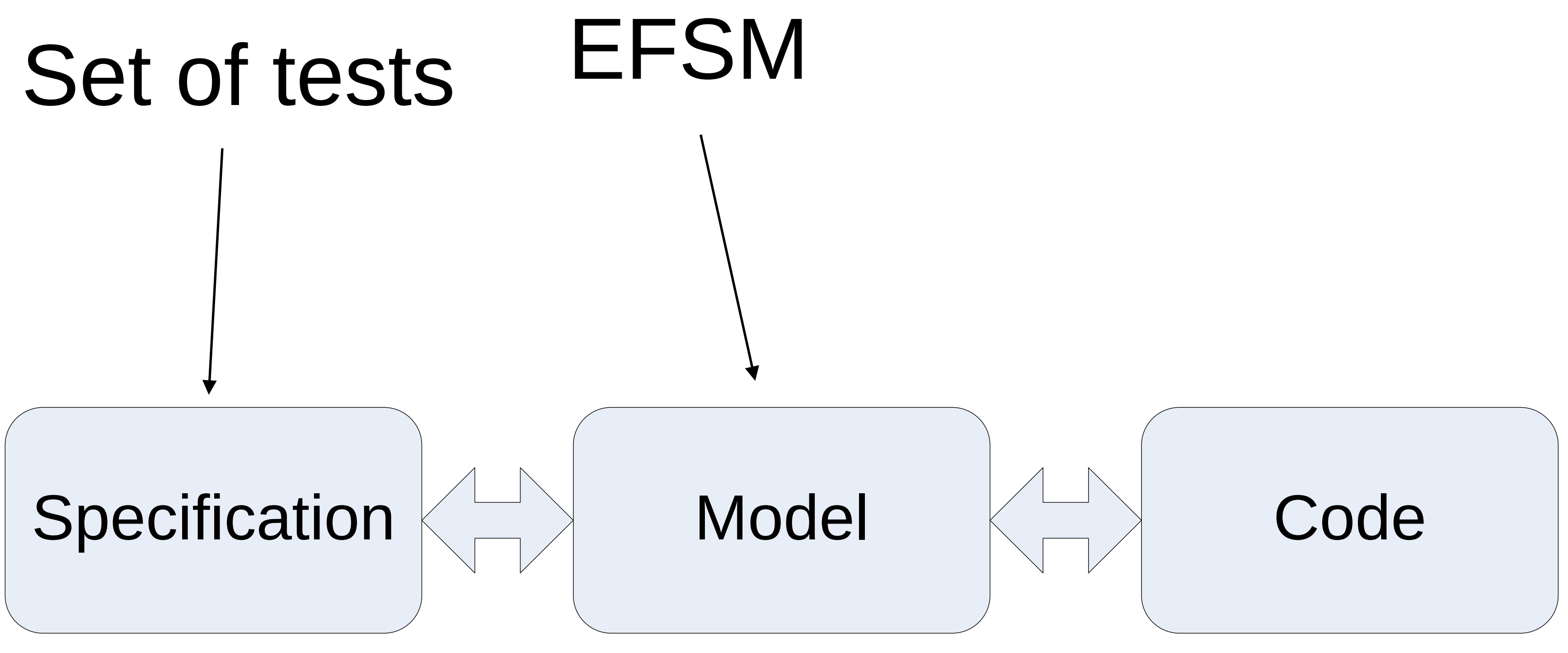
GECCO-2012
Graduate Students Workshop
July 7, 2012

Overview (1)

- Part of a bigger project on automated software engineering and automata-based programming
- We focus on model driven-development

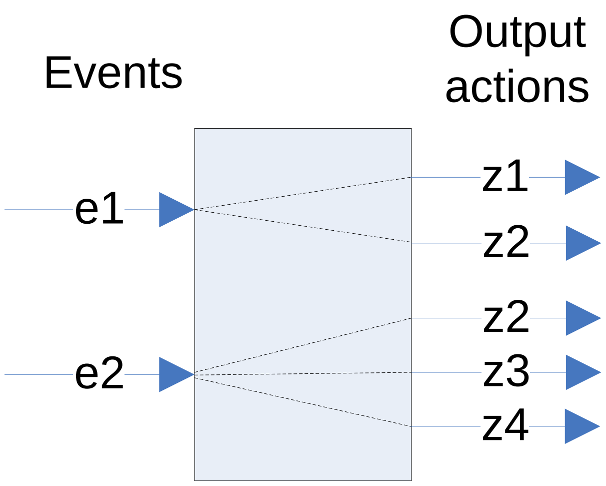


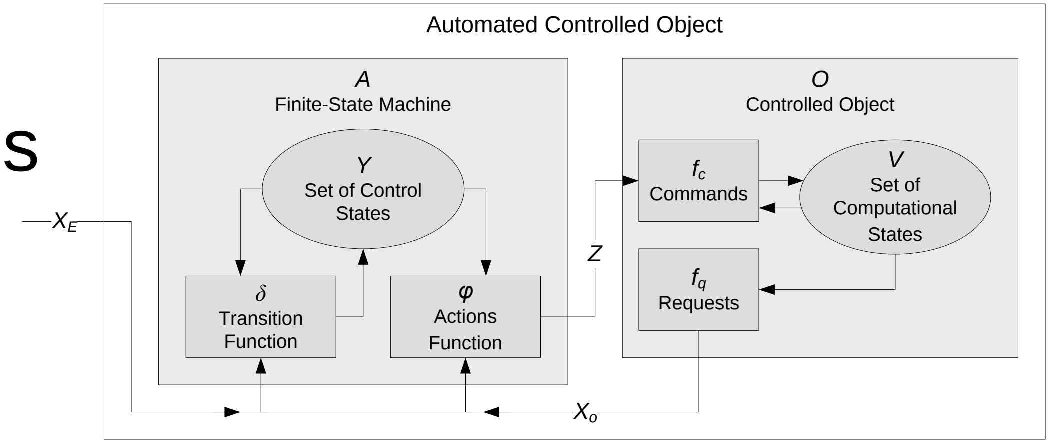
Overview (2)



Automata-based Programming

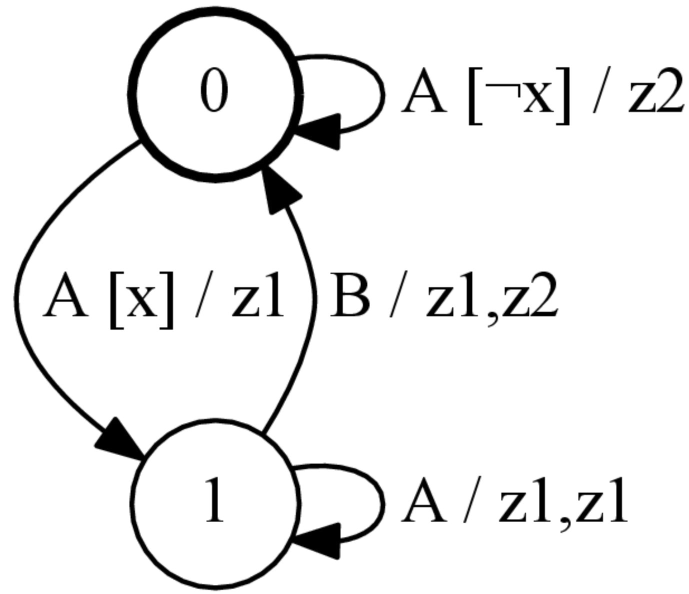
- Entities with complex behavior should be designed as automated controlled objects
- Control states and computational states
- Events
- Output actions





Definitions

- EFSM:
  - input events
  - input Boolean variables
  - output actions
- Test is a pair of two sequences
  - Input sequence of pairs *I* = <*e*, *f*>
    - *e* – input event
    - *f* – guard condition – Boolean formula on input variables
  - *A* – reference sequence of output actions
- EFSM on the picture complies with
  - <*A*, !*x*>, <*A*, *x*>
  - *z*2, *z*1
- EFSM on the picture does not comply with
  - <*A*, *x*>
  - *z*2



Example – Alarm Clock (1)

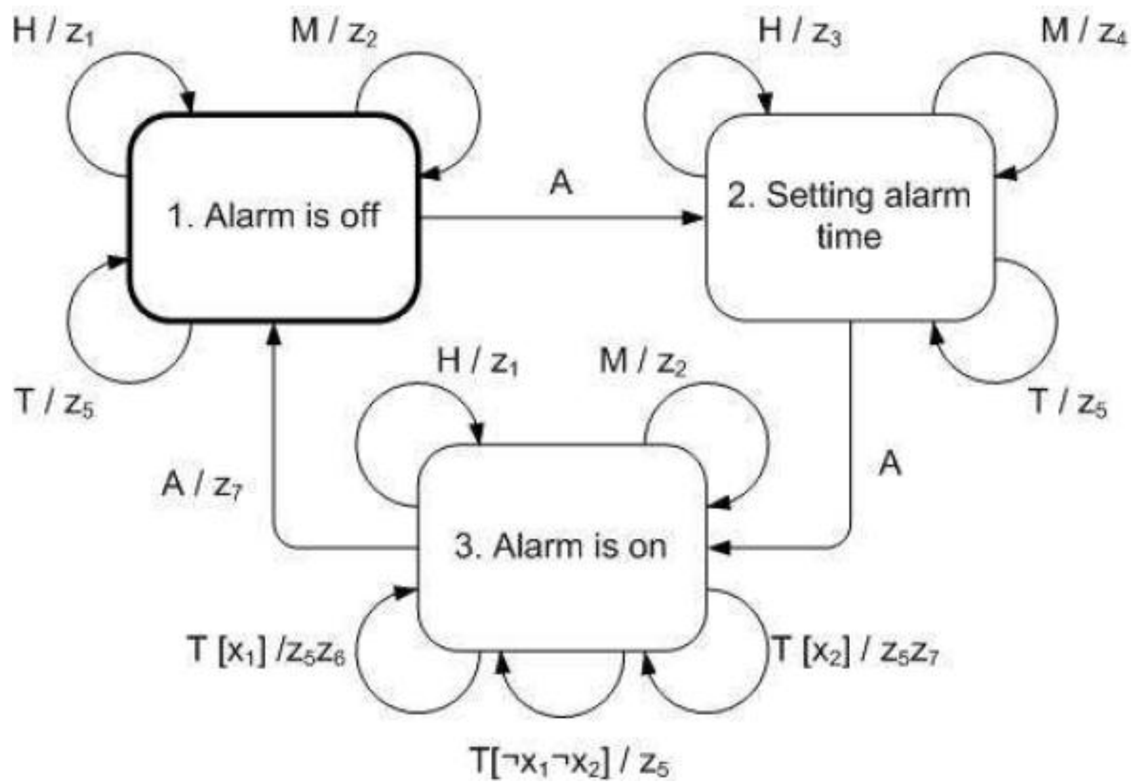
- Four events
  - H – button “H” pressed
  - M – button “M” pressed
  - A – button “A” pressed
  - T – occurs on each time tick
- Two input variables
- Seven output actions

Example – Alarm Clock (2)

Tests

- Test 1:
  - T
  - z5
- Test 2:
  - H
  - z1
- Test 3:
  - A, H
  - z3
- …

Model

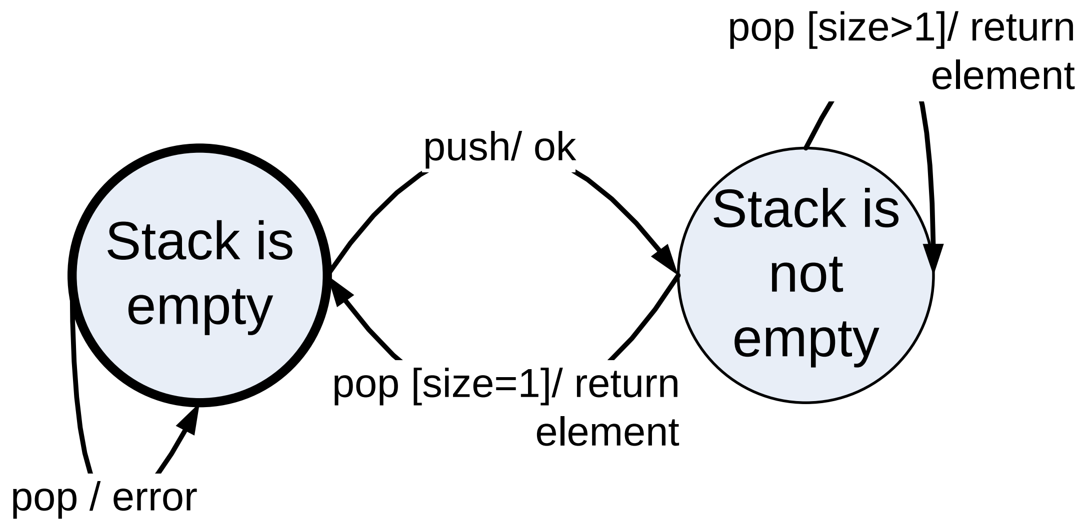


Example – Stack (1)

Tests

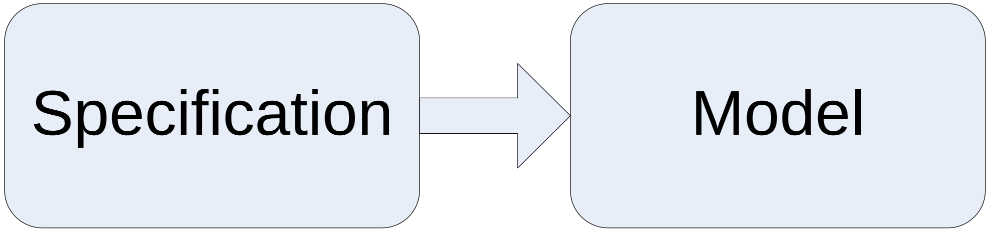
- Test 1:
  – push, pop
  – ok, return element
- Test 2:
  – push, pop, pop
  – ok, return element, error
- Test 3:
  – push, push, pop, pop
  – ok, ok, return element, return element
- …

Model

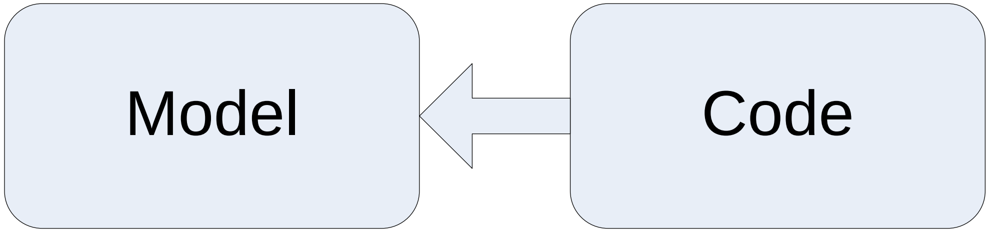


Problems Considered

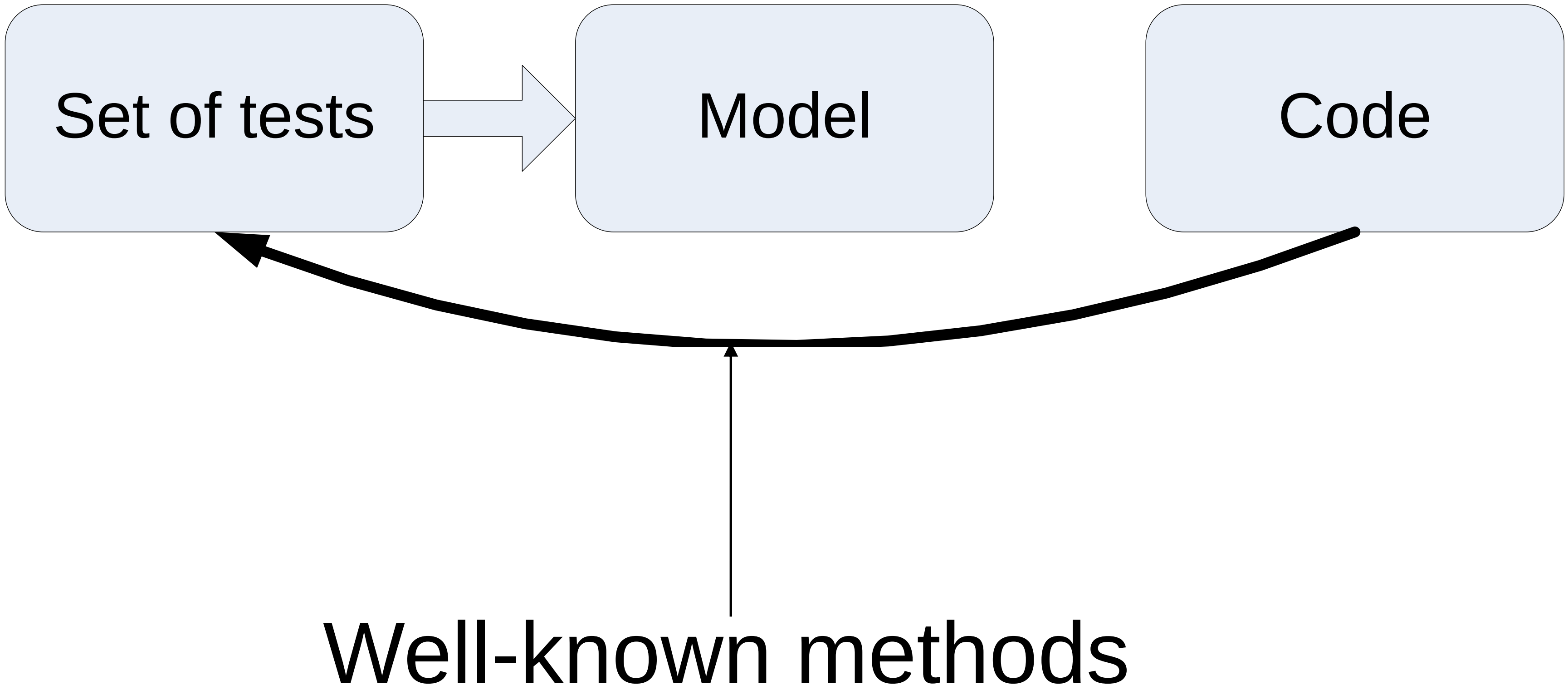
- Automated model design



- Model mining



Reduction to Automated Model Design



Problem Definition

- Input data:
  - Set of tests
  - Number of states in EFSM (C)
- Need to find an EFSM with C states complying with all tests

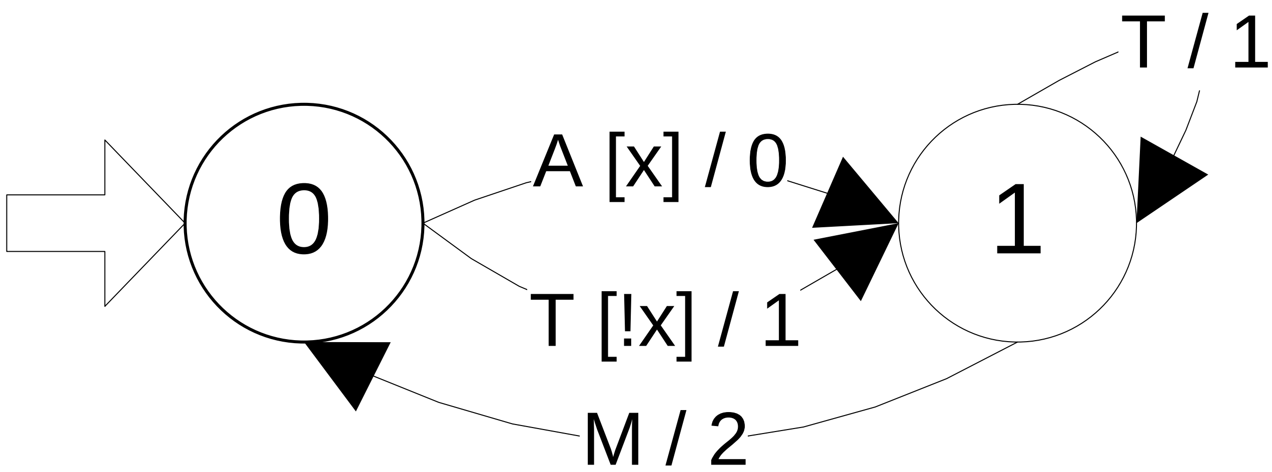
Precomputations

- For each pair of guard conditions from tests compute:
  - If they are same as Boolean functions
  - If they have common satisfying assignment
- Time complexity:
  - $O(n^2 2^{2m})$ where n is total size of tests' input sequences, m is maximal number of input variables occurring in guard condition (in practice m is not greater than 5)

Evolutionary Algorithms

- Random mutation hill climber and evolutionary strategy can be easily used
- Problem with genetic algorithms – no meaningful crossover (“it is hard to automatically identify functionally coherent modules in automata”)
  - Johnson, C. Genetic Programming with Fitness based on Model Checking. *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, 2007. Volume 4445/2007, pp. 114–124.
  - Lucas, S. and Reynolds, J. Learning Deterministic Finite Automata with a Smart State Labeling Algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. Vol. 27, №7, 2005, pp. 1063–1074.
- This problem can be solved with test-based crossover

Individual Representation

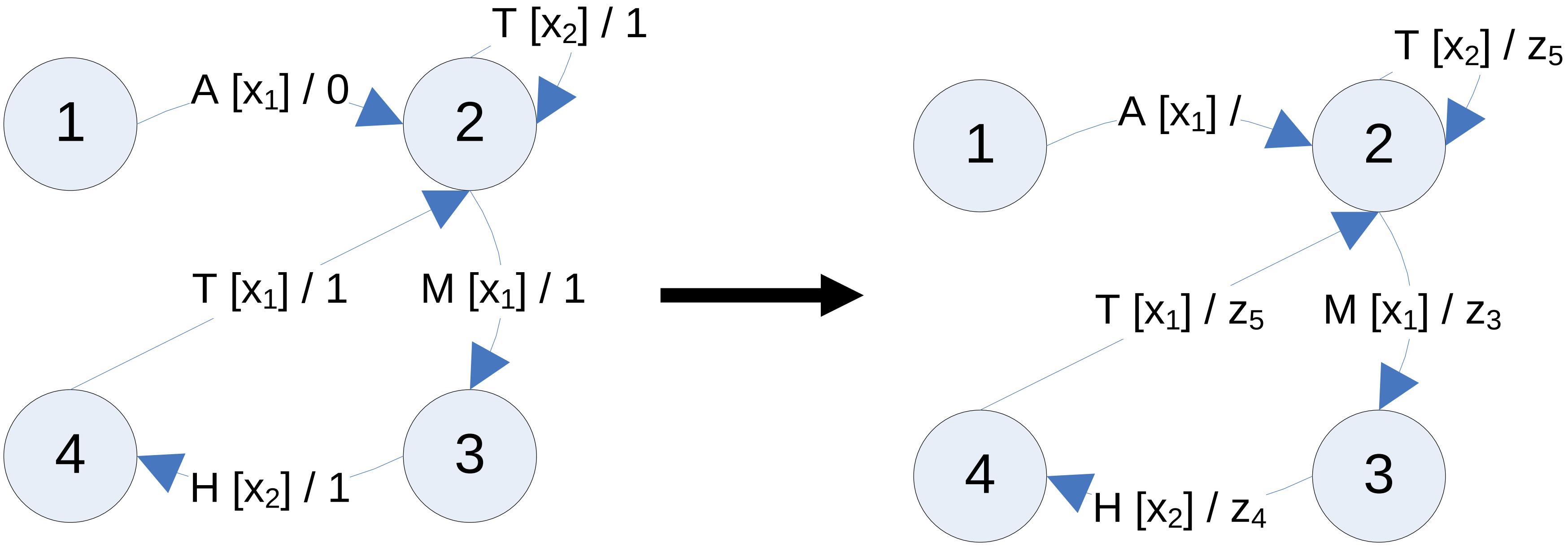


{2, 0, {{A, x, 1, 0}, {T, !x, 1, 1}}, {{T, true, 1, 1}, {M, true, 0, 2}}}

All EFSMs considered during one of evolutionary algorithm have the same number of states

Transition Labeling Algorithm

- Applied to each individual before calculation of fitness function



Mutation

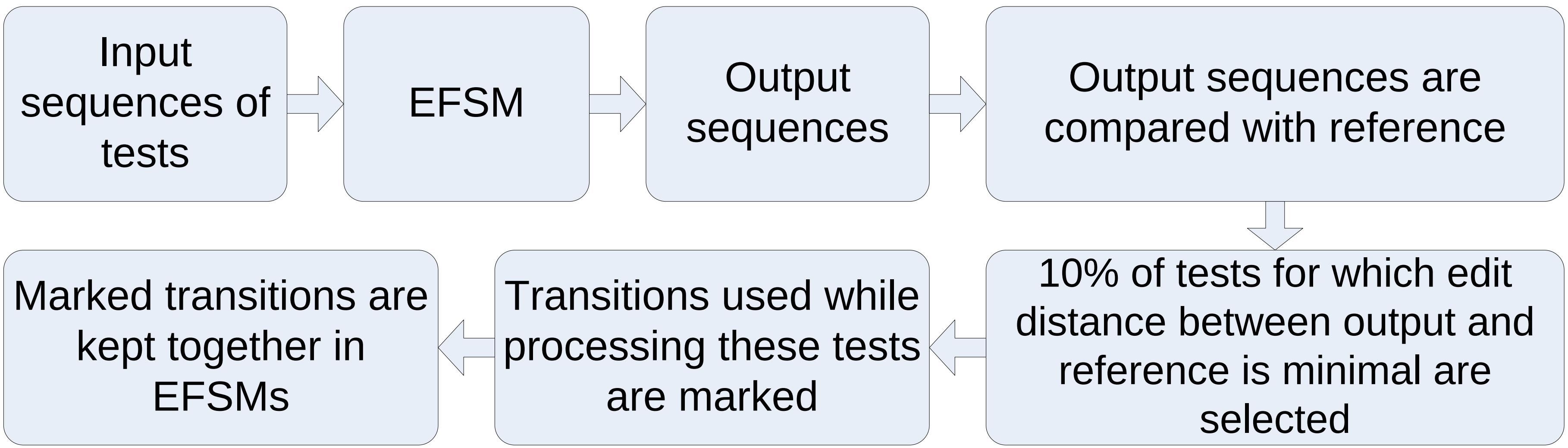
- Change of transition
  - Final state
  - Event
  - Guard condition
  - Number of output actions
- Addition of deletion of a transitions

Fitness Function

$$FF_1 = \frac{1}{|T|}\sum_{j=1}^{|T|}\left(1 - \frac{ED(O_j, A_j)}{\max\left(len(O_j), len(A_j)\right)}\right)$$

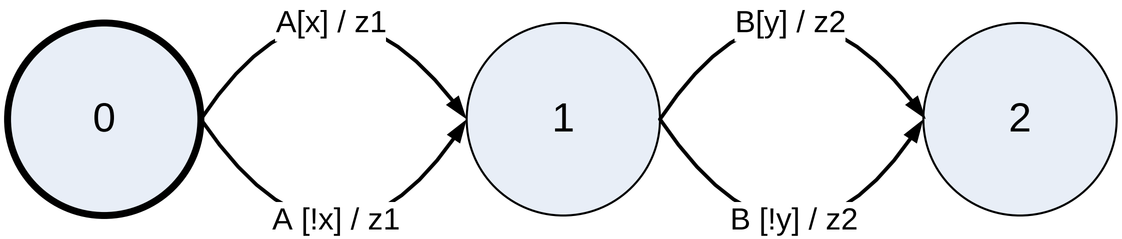
$$\mathrm{FF}_2 = \begin{cases} 10 \cdot \mathrm{FF}_1 + \dfrac{1}{M} \cdot (M - \mathrm{cnt}), \mathrm{FF}_1 < 1 \\ 20 + \dfrac{1}{M} \cdot (M - \mathrm{cnt}), \mathrm{FF}_1 = 1 \end{cases}$$

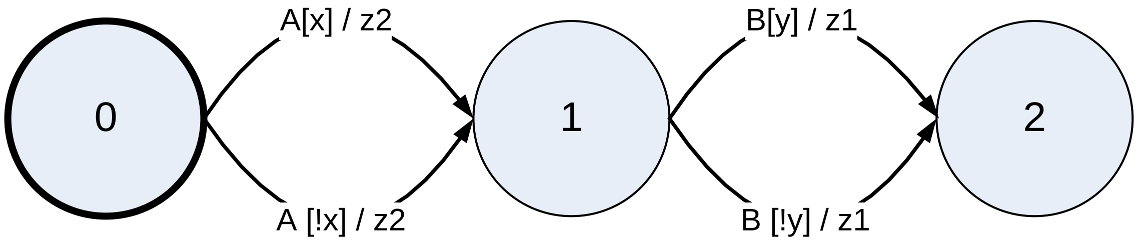
Test-based Crossover



Example (1)

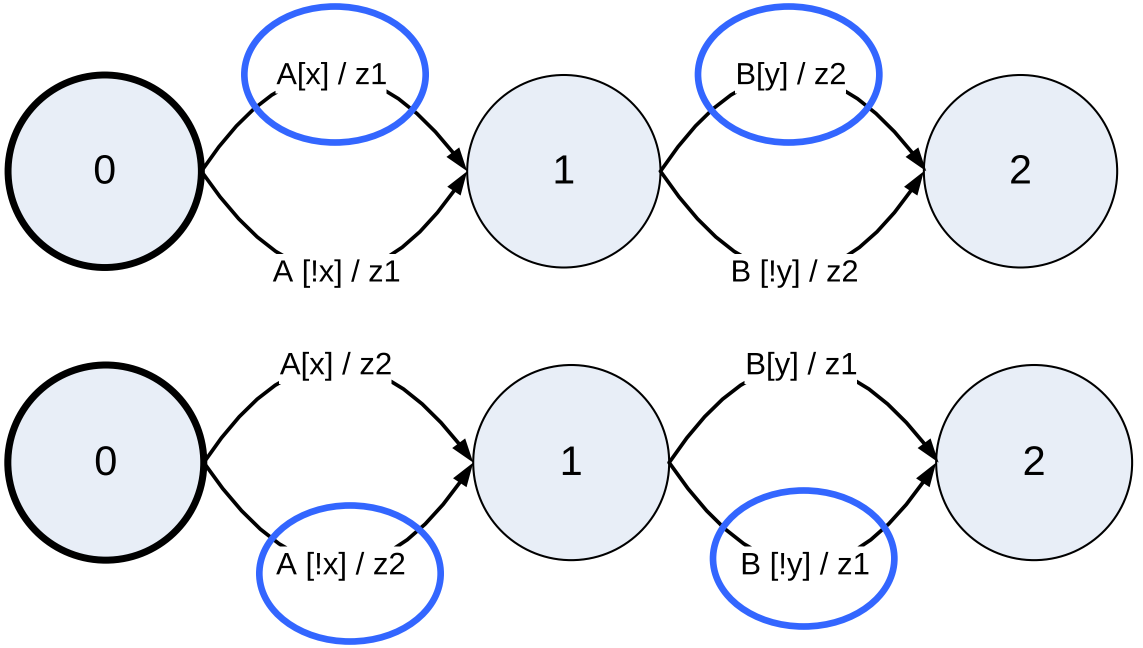
- Test set contains:
  - Test 1:
    - A [x], B [y]
    - z1, z2
  - Test 2:
    - A [!x], B [!y]
    - z2, z1
  - …





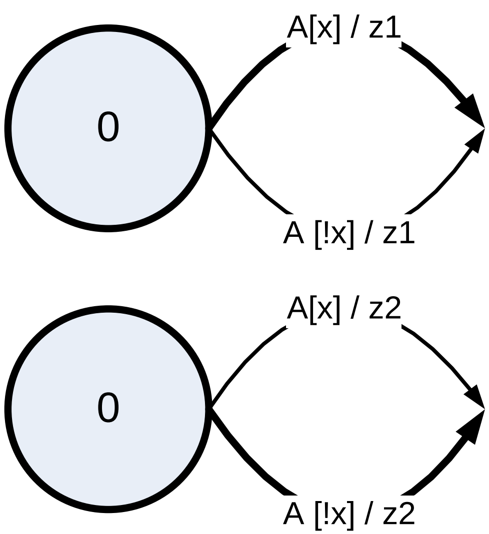
Example (2)

- Test set contains:
  - Test 1:
    - A [x], B [y]
    - z1, z2
  - Test 2:
    - A [!x], B [!y]
    - z2, z1
  - …

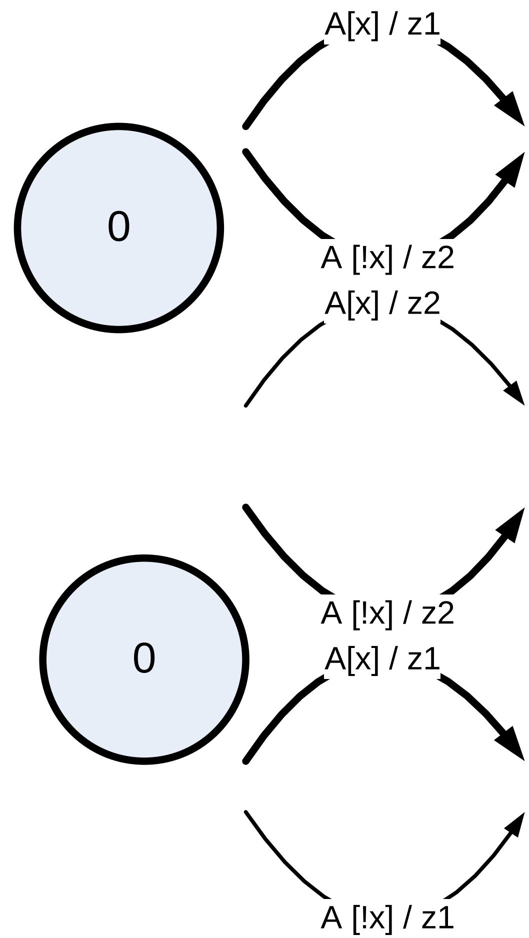


Example (3)

Parents

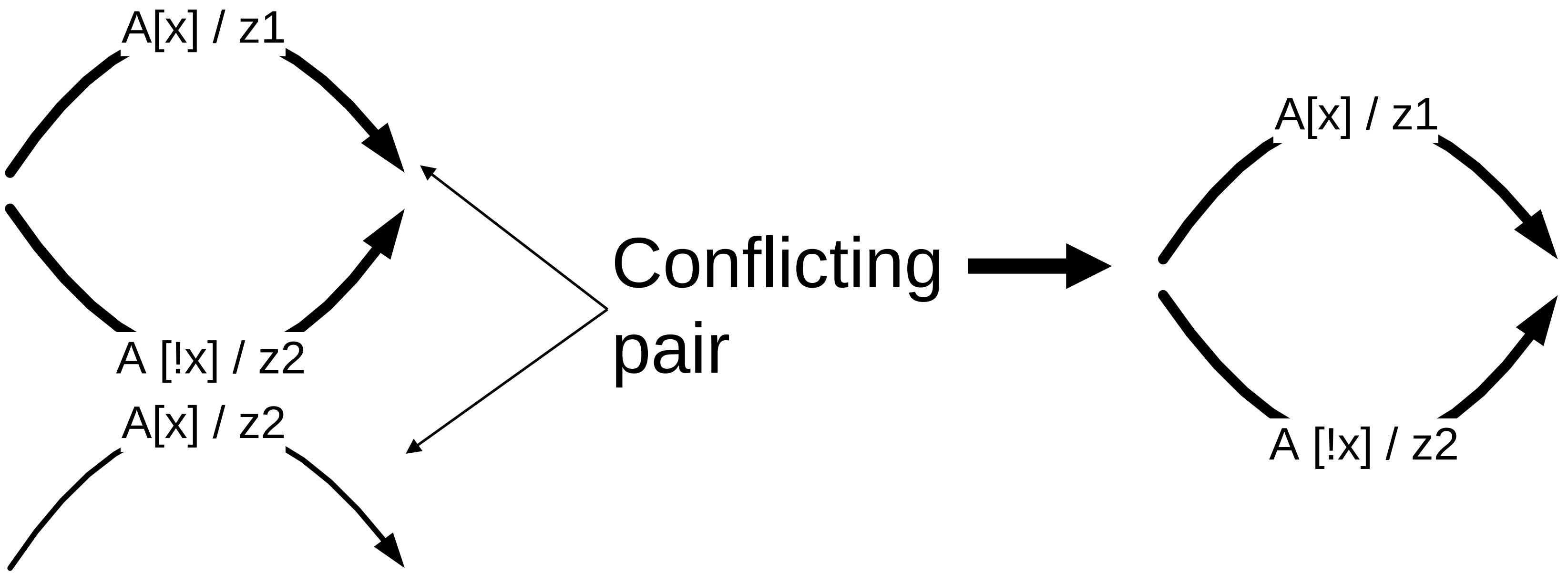


Offsprings



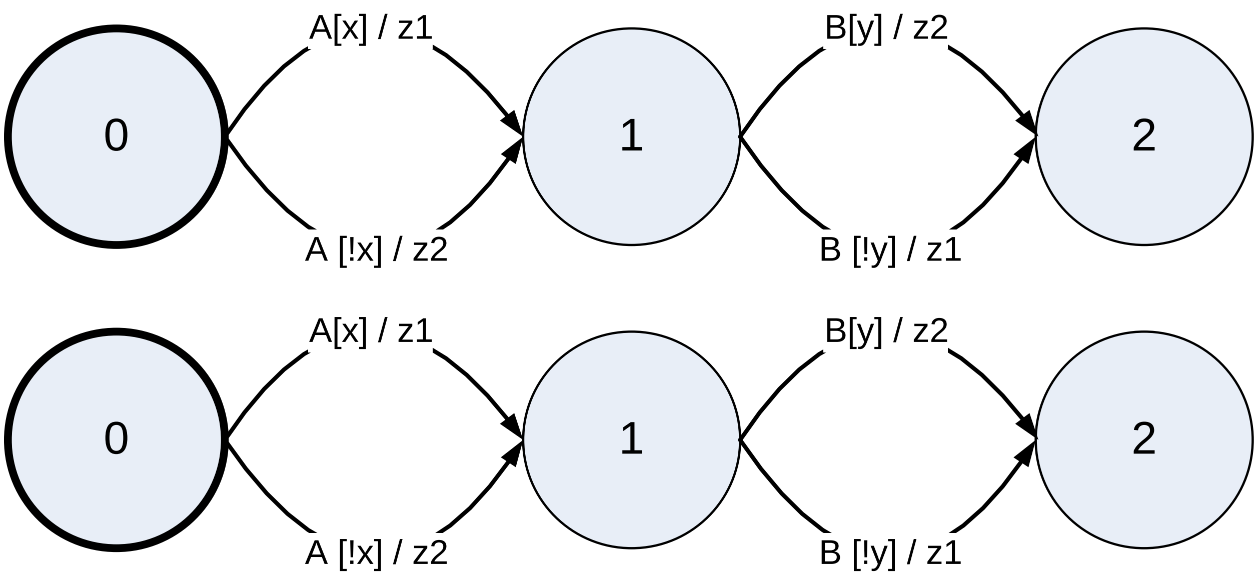
Example (4)

- Duplicate and contradictory transitions removal
- Showing for state 0 of first offspring



Example (5)

- Both offsprings pass both tests



Ant Colony Optimization

- Graph:
  - Nodes – finite-state machines
  - Edges – *mutations* of finite-state machines
  - Graph is too big to be constructed explicitly

Algorithm:

1. Graph G = {random FSM}
2. While (true)

   Launch colony on graph G

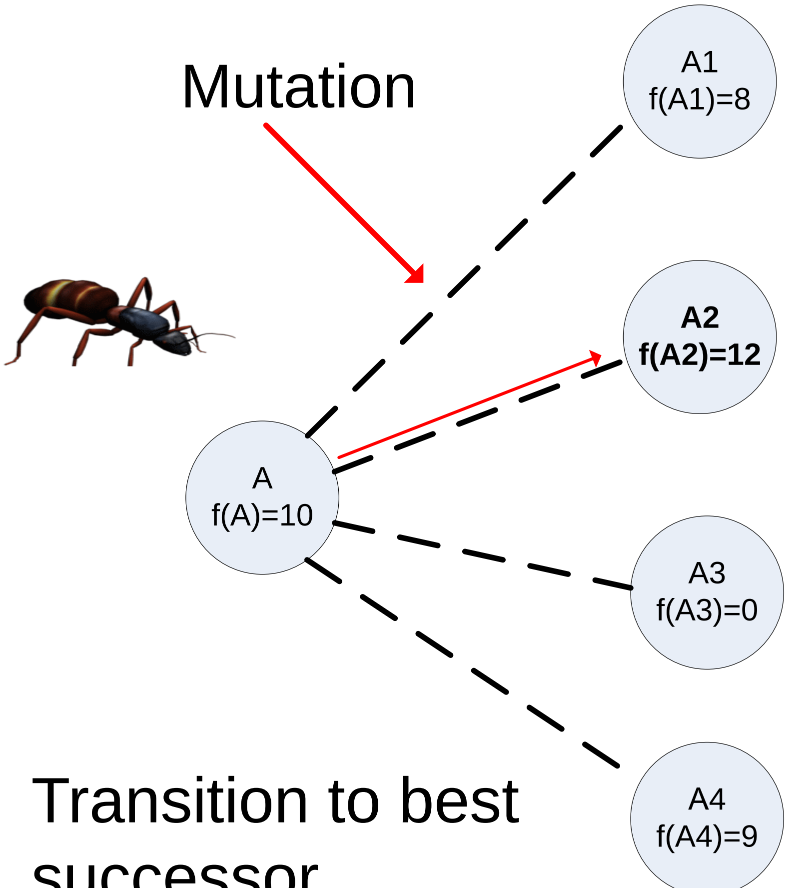
   Update pheromone values

   Check stop conditions:

   if stagnation, restart

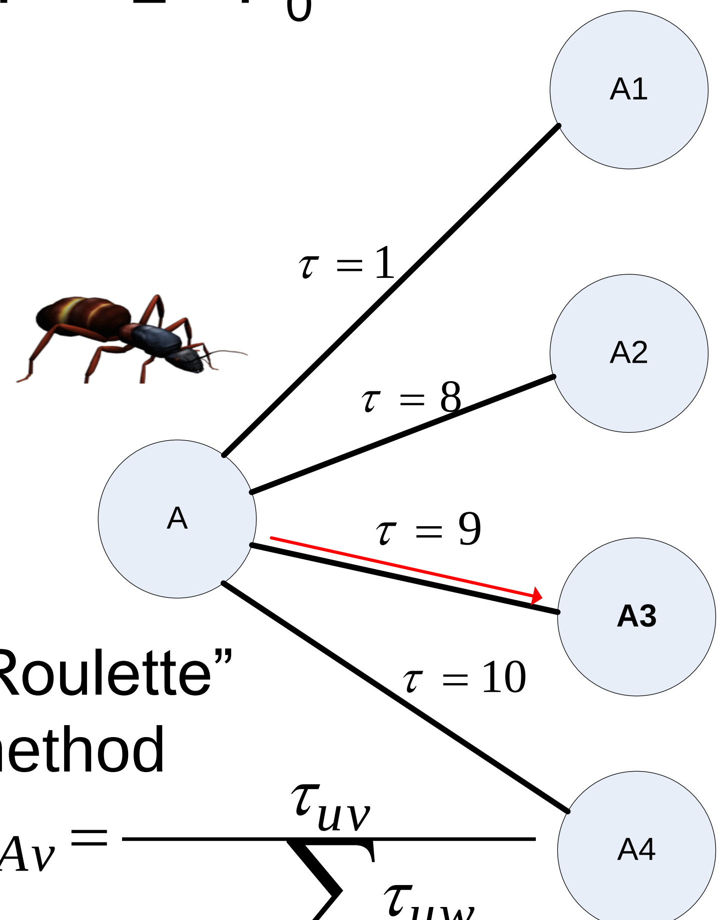
Choosing the Next Node

$P = P_0$



Transition to best successor

$P = 1 - P_0$



"Roulette" method

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

Update Pheromone Values

- Quality of solution (ant path) – max value of *f* among all nodes in path
- New pheromone value on edge:

$$\tau_{uv} = \rho\tau_{uv} + \Delta\tau_{uv}^{best}$$

- ρ < 1 – evaporation rate
- $\Delta\tau_{uv}^{best}$ – max pheromone value ever added to the edge (u, v)

Choosing Start Nodes on Restart

- **Best path** – path from some node to a node with max value of *f*
- Start nodes are selected with “roulette” method from nodes of best path

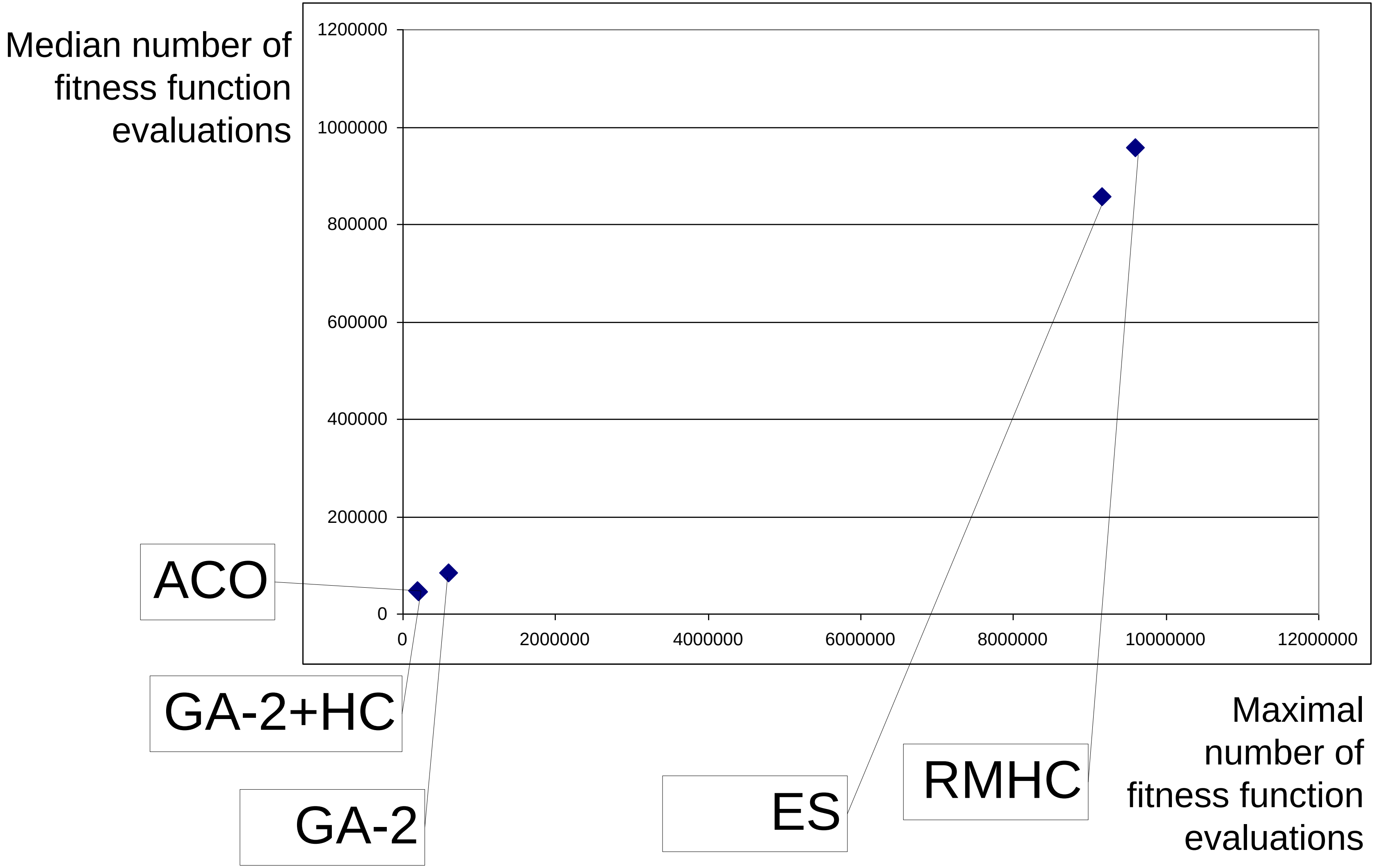
Experiments (1)

- Six algorithms:
  – a genetic algorithm with traditional crossover (GA-1)
  – a random mutation hill climber (RMHC)
  – (1+1) evolutionary strategy (ES)
  – a genetic algorithm with test-based crossover (GA-2)
  – GA-2 hybridized with RMHC (GA-2+HC)
  – ant colony optimization (ACO)
- Input data: 38 tests for alarm clock
  – total length of input sequences 242
  – total length of reference sequences 195
- 1000 runs of each algorithm

Experiments (2)

Algorithm	**Min**	**Max**	**Avg**	**Median**
GA-1	855390	38882588	5805943	4588736
RMHC	1150	9592213	1423983	957746
ES	1506	9161811	3447390	856730
GA-2	32830	599022	117977	83787
GA-2+HC	26740	188509	53706	48106
ACO	2440	210971	53944	46293

Experiments (3)



Summary

- Test-based crossover greatly improves the performance of GA
- GA on average significantly outperforms RMHC and ES
- ACO outperforms GA-2
- Difference between average performance of ACO and GA-2+HC is insignificant

Thank you!

Questions?