Comparing Metaheuristic Algorithms for Error Detection in Java Programs

LENGUAJES Y CIENCIAS DE LA COMPUTACIÓN
UNIVERSIDAD DE MÁLAGA

UNIVERSIDAD DE MÁLAGA

Francisco Chicano, Marco Ferreira and Enrique Alba

Motivation

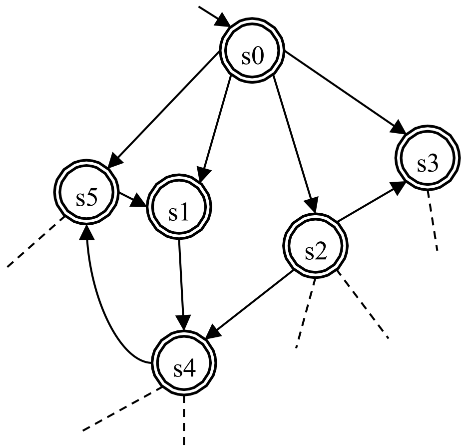
- **Concurrent software is difficult to test ...**
- **... and it is in the heart of a lot of critical systems**

- **Techniques for proving the correctness of concurrent software are required**
- **Model checking → fully automatic**
- **Traditional techniques for this purpose have problems with large models**
- **We compare several metaheuristics and classical algorithms for model checking**

Explicit State Model Checking

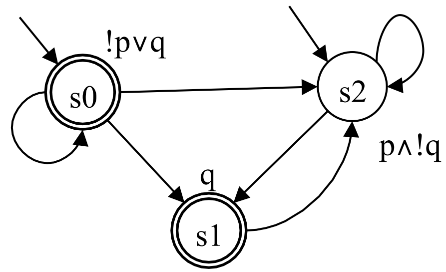
- **Objective: Prove that model M satisfies the property f :** $M \models f$
- **In the general case, *f* is a temporal logic formula (LTL, CTL, etc.)**

Model *M*

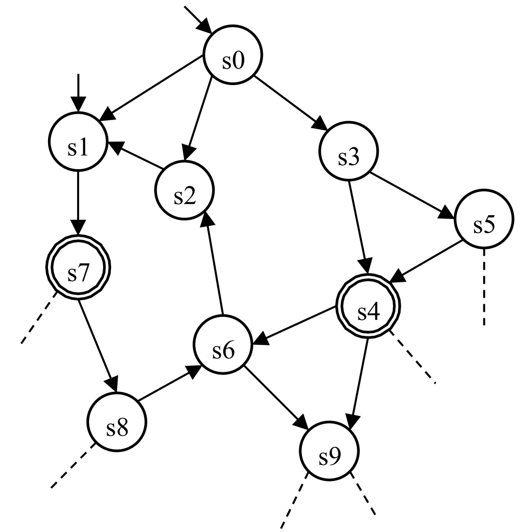


$\cap$

LTL formula ¬ *f* (never claim)

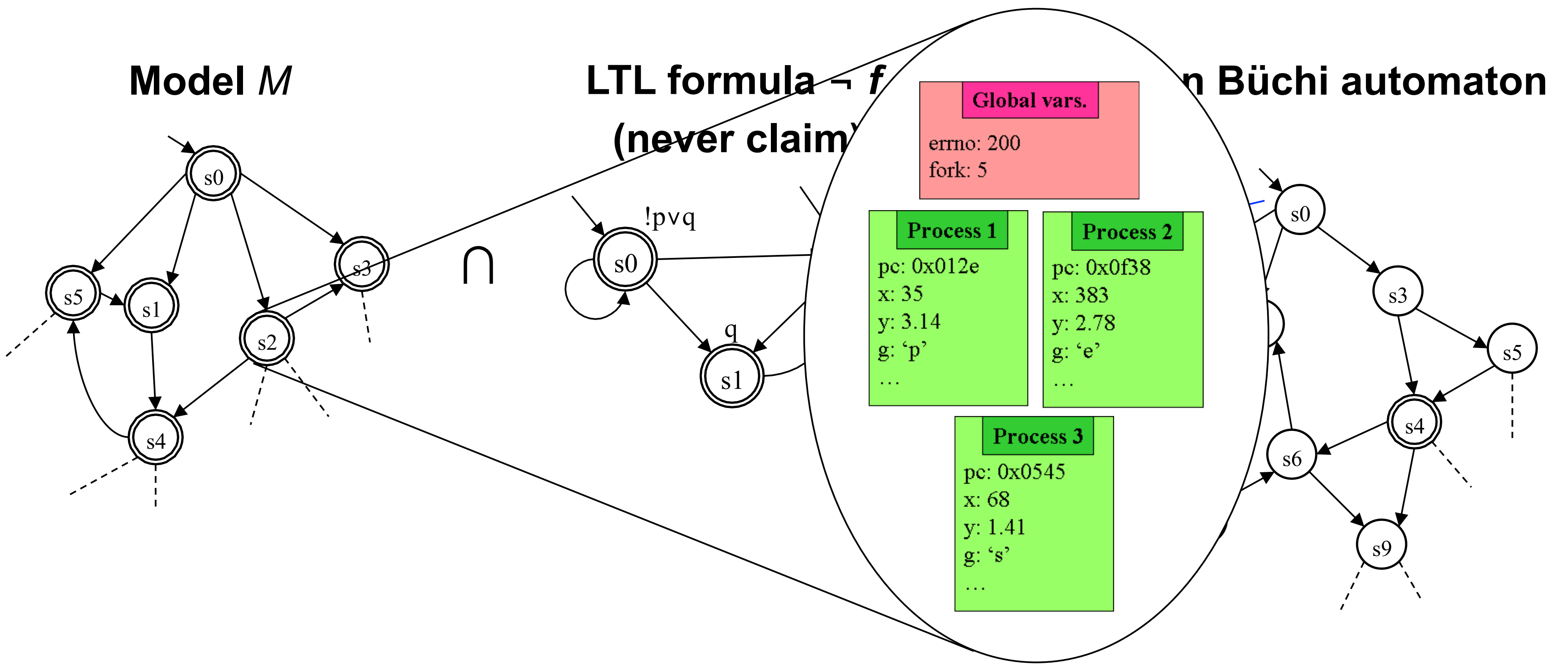


Intersection Büchi automaton



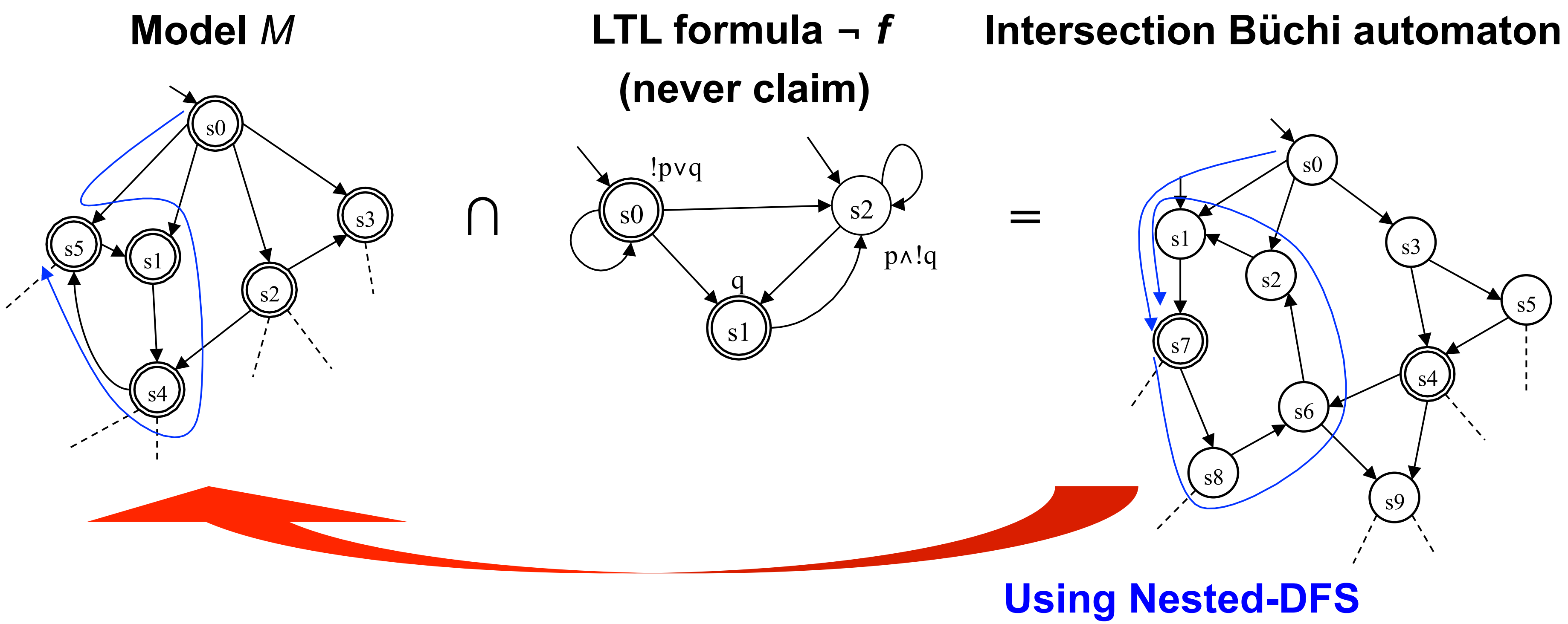
Explicit State Model Checking

- **Objective: Prove that model M satisfies the property f :** $M \models f$
- **In the general case, *f* is a temporal logic formula (LTL, CTL, etc.)**



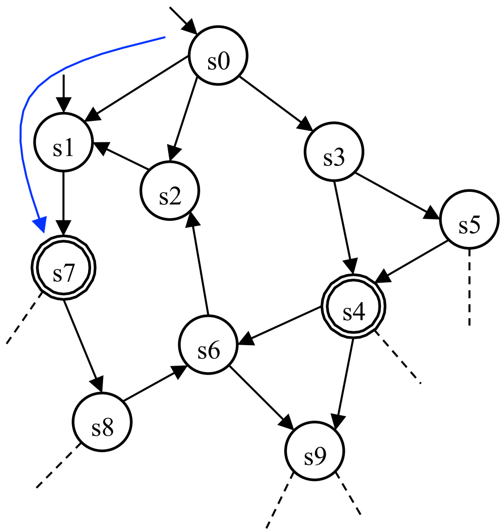
Explicit State Model Checking

- **Objective: Prove that model M satisfies the property f :** $M \models f$
- **In the general case, *f* is a temporal logic formula (LTL, CTL, etc.)**



Safety properties

$$\forall \sigma \in S^{\omega} : \sigma \nvdash \mathcal{P} \Rightarrow (\exists i \geq 0 : \forall \beta \in S^{\omega} : \sigma_i \beta \nvdash \mathcal{P})$$



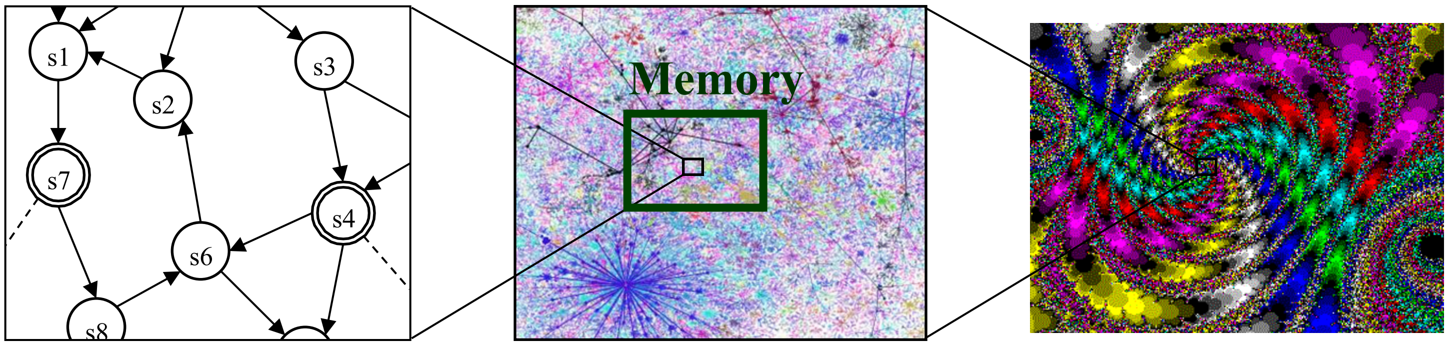
Properties in JPF

- Exceptions
- Deadlocks

- **An error trail is an execution path ending in an error state**
- **The search for errors is transformed in a graph exploration problem (DFS, BFS)**

State Explosion Problem

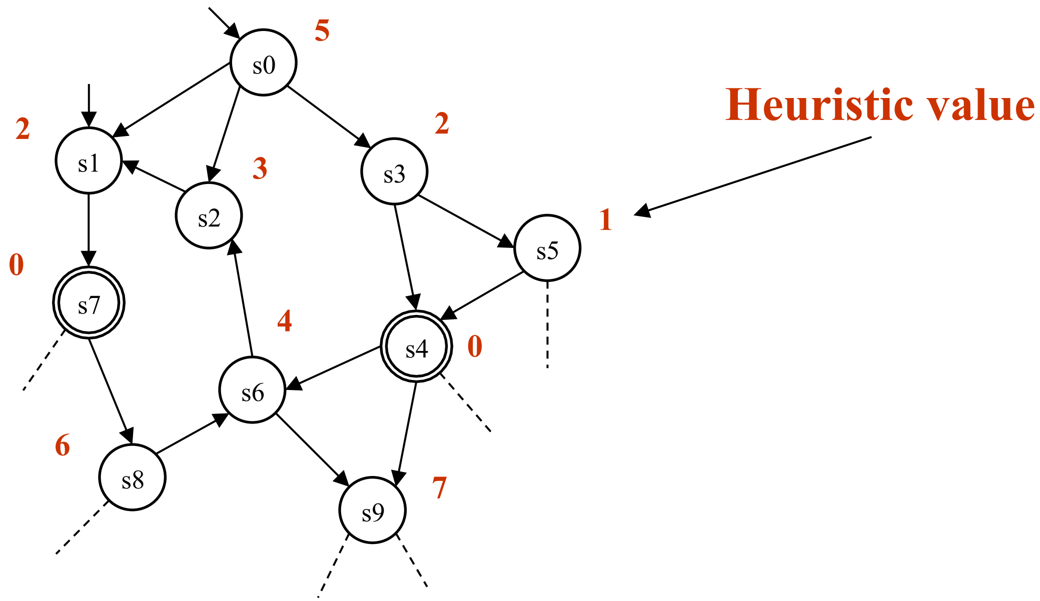
- Number of states very large even for small models



- Example: Dining philosophers with *n* philosophers → 3^n states
- For each state we need to store the heap and the stacks of the different threads
- Solutions: collapse compression, minimized automaton representation, bitstate hashing, partial order reduction, symmetry reduction
- Large models cannot be verified but errors can be found

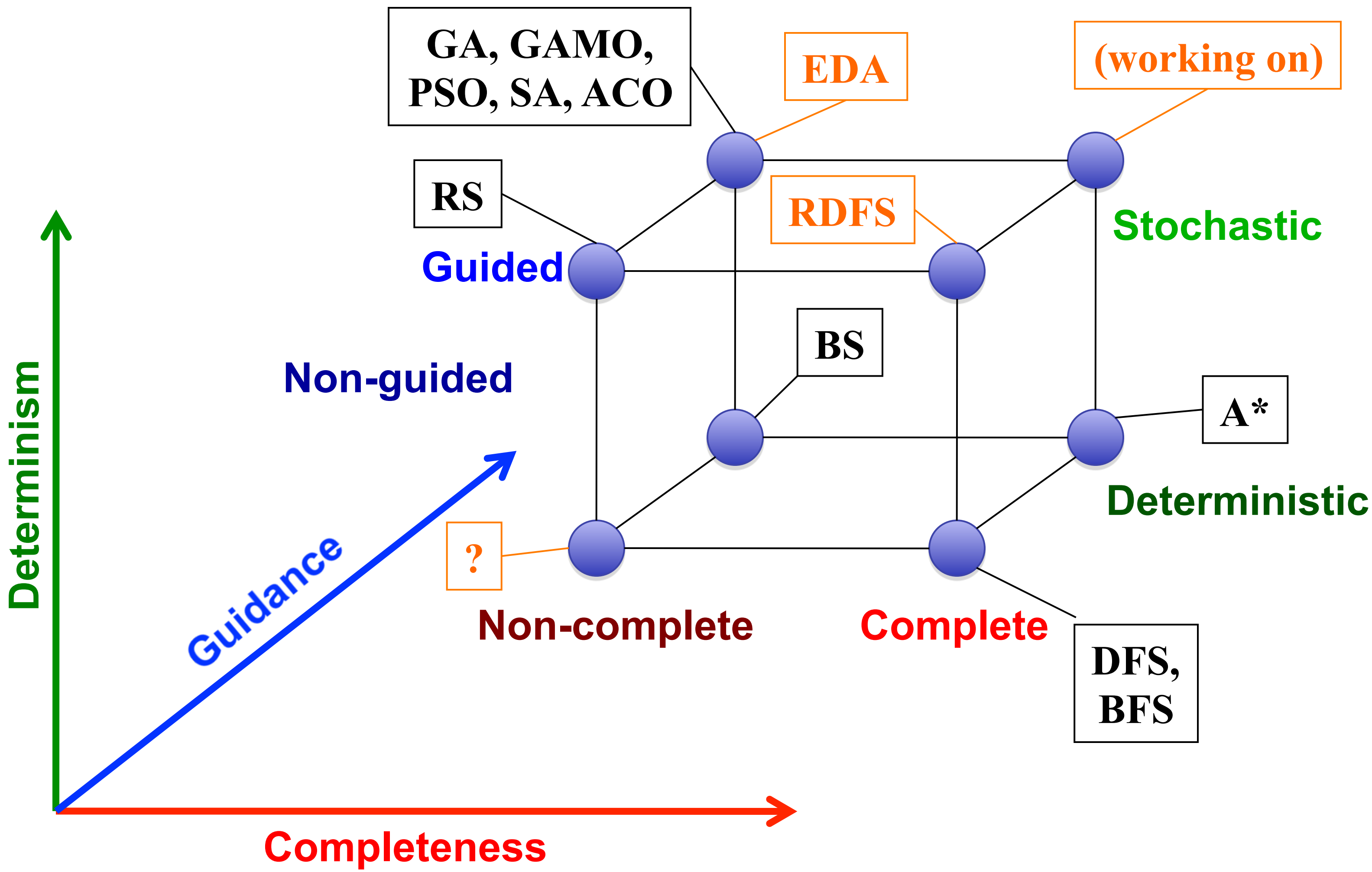
Heuristic Model Checking

- The search for errors can be directed by using heuristic information



- Different kinds of heuristic functions have been proposed in the past:
  - Formula-based heuristics
  - Structural heuristics
  - Deadlock-detection heuristics
  - State-dependent heuristics

Classification of Algorithms



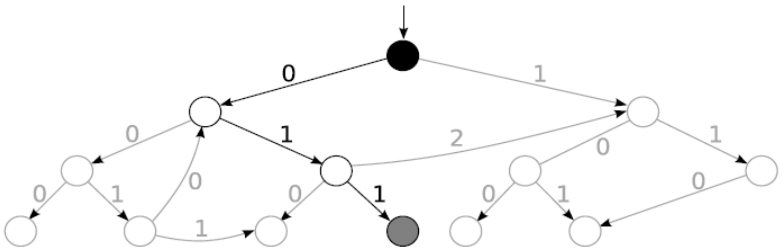
Genetic Algorithm

P = generateInitialPopulation();
evaluate(P);
while not stoppingCondition() **do**
  P' = selectParents(P);
  P' = applyVariationOperators(P');
  evaluate(P');
  P = selectNewPopulation(P, P');
end while
return the best found solution

Solution encoding

(floating point values)

0.5 0.1 0.9 0.3 0.5 0.9



Crossover

0.5 0.1 | **0.9 0.3 0.5 0.9** → 0.5 0.1 | **0.2 0.0 0.6**

0.2 0.6 0.1 0.7 0.8 0.4 | **0.2 0.0 0.6** → 0.2 0.6 0.1 0.7 0.8 0.4 | **0.9 0.3 0.5 0.9**

Mutation

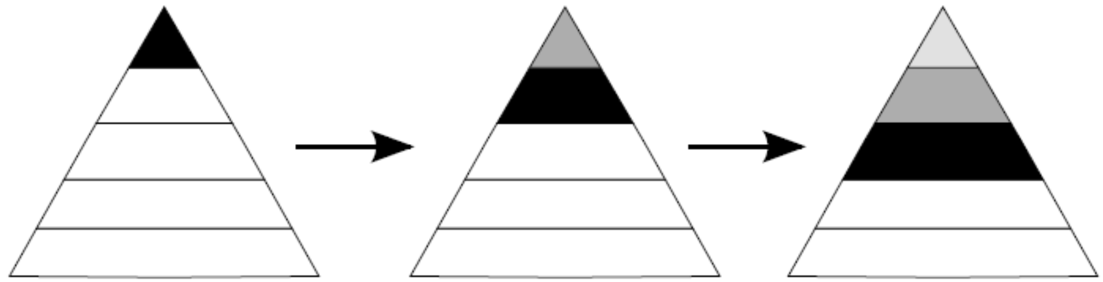
0.5 0.1 0.9 0.3 0.5 0.9 → 0.5 0.1 0.6 0.3 0.5 0.9

Genetic Algorithm with Memory Operator

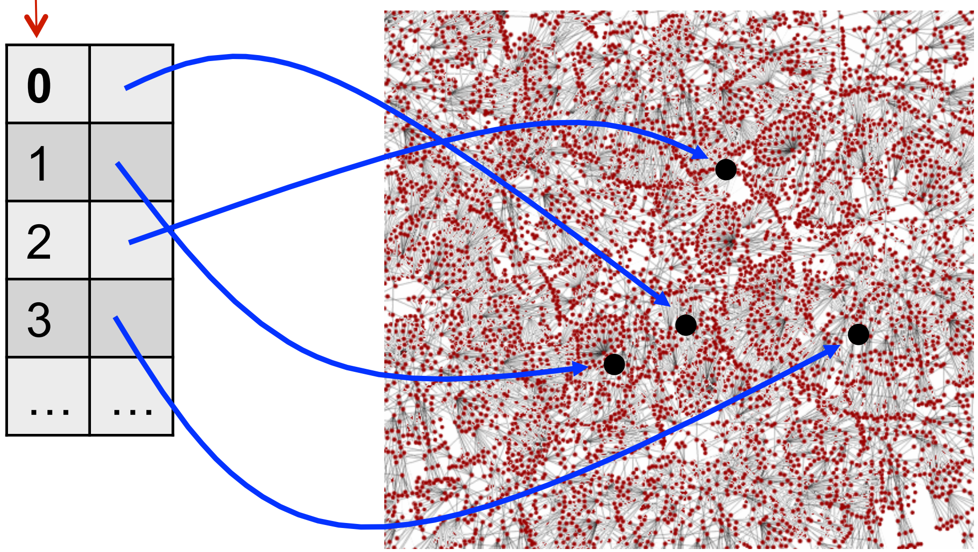
Solution encoding

(floating point values)

0.5 0.1 0.9 0.3 0.5 0.9



Index in a table of states



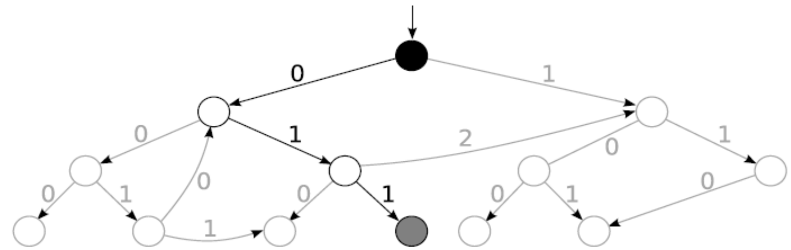
Particle Swarm Optimization

```
P = generateInitialPopulation();
while not stoppingCondition() do
    evaluate(P);
    calculateNewVelocityVectors(P);
    move(P);
end while
return the best found solution
```

Particles

0.2 -1.4 -3.5 → **Position (solution)**

1.0 10.3 7.2 → **Velocity**



Personal best

Inertia

$$v^i_j(t+1) = w \cdot v^i_j(t) + c_1 \cdot r_1 \cdot (p^i_j - x^i_j(t)) + c_2 \cdot r_2 \cdot (n^i_j - x^i_j(t))$$

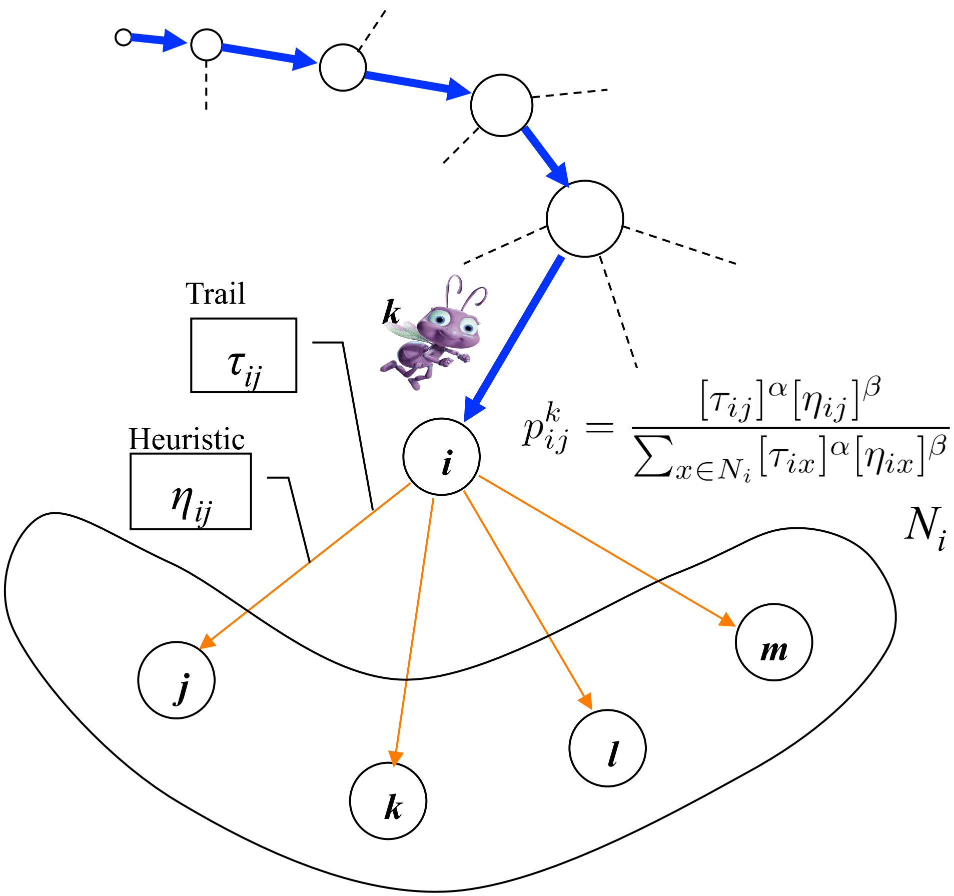
$$x^i_j(t+1) = x^i_j(t) + v^i_j(t+1)$$

Global best

Ant Colony Optimization

```
procedure ACOMetaheuristic
    ScheduleActivities
        ConstructAntsSolutions
        UpdatePheromones
        DaemonActions // optional
    end ScheduleActivities
end procedure
```

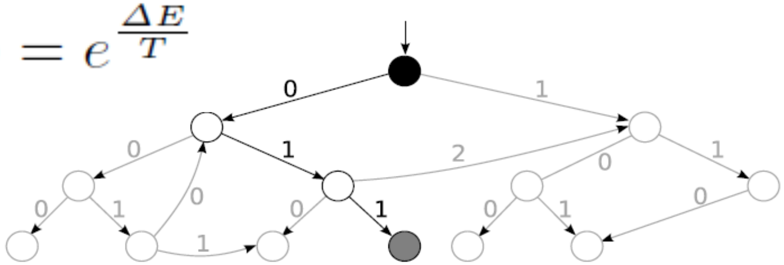
- **The ant selects stochastically its next node**
- **The probability of selecting one node depends on the pheromone trail and the heuristic value (optional) of the edge/node**
- **The ant stops when a complete solution is built**



Simulated Annealing

```
S = generateInitialSolution();
T = initialTemperature;
while not stoppingCondition() do
   N = getRandomNeighbor(S);
   ΔE = energy(N) - energy(S);
   if ΔE > 0 OR random(0,1) < probabilityAcceptance(ΔE, T) then
      S = N
   end if
   T = updateTemperature(T);
end while
return S
```

$$probabilityAcceptance(\Delta E, T) = e^{\frac{\Delta E}{T}}$$



Neighbor

0.5 0.1 0.9 0.3 0.5 0.9 → 0.5 0.1 0.6 0.3 0.5 0.9

Parameterization

- We used 3 scalable and 2 non-scalable models for the experiments

Program	LoC	Classes	Processes
`dinj`	63	1	j+1
`phij`	176	3	j+1
`marj`	186	4	j+1
`giop`	746	13	7
`garp`	458	7	7

- Maximum number of expanded states: 200 000
- Fitness function:

$$f(x) = deadlock + numblocked + \frac{1}{1 + pathlen}$$

- 100 independent executions of stochastic algorithms

Parameterization

- **We used 3 scalable and 2 non-scalable models for the experiments**

Program	Lo[illegible]	[illegible]s	Processes
`dinj`	[illegible]	1	j+1
`phij`	[illegible]	3	j+1
`marj`	186	4	j+1
`giop`	74[illegible]	[illegible]3	7
`garp`	458	7	7

j=4 to 20

j=4 to 36

j=2 to 10

- **Maximum number of expanded states: 200 000**
- **Fitness function:**

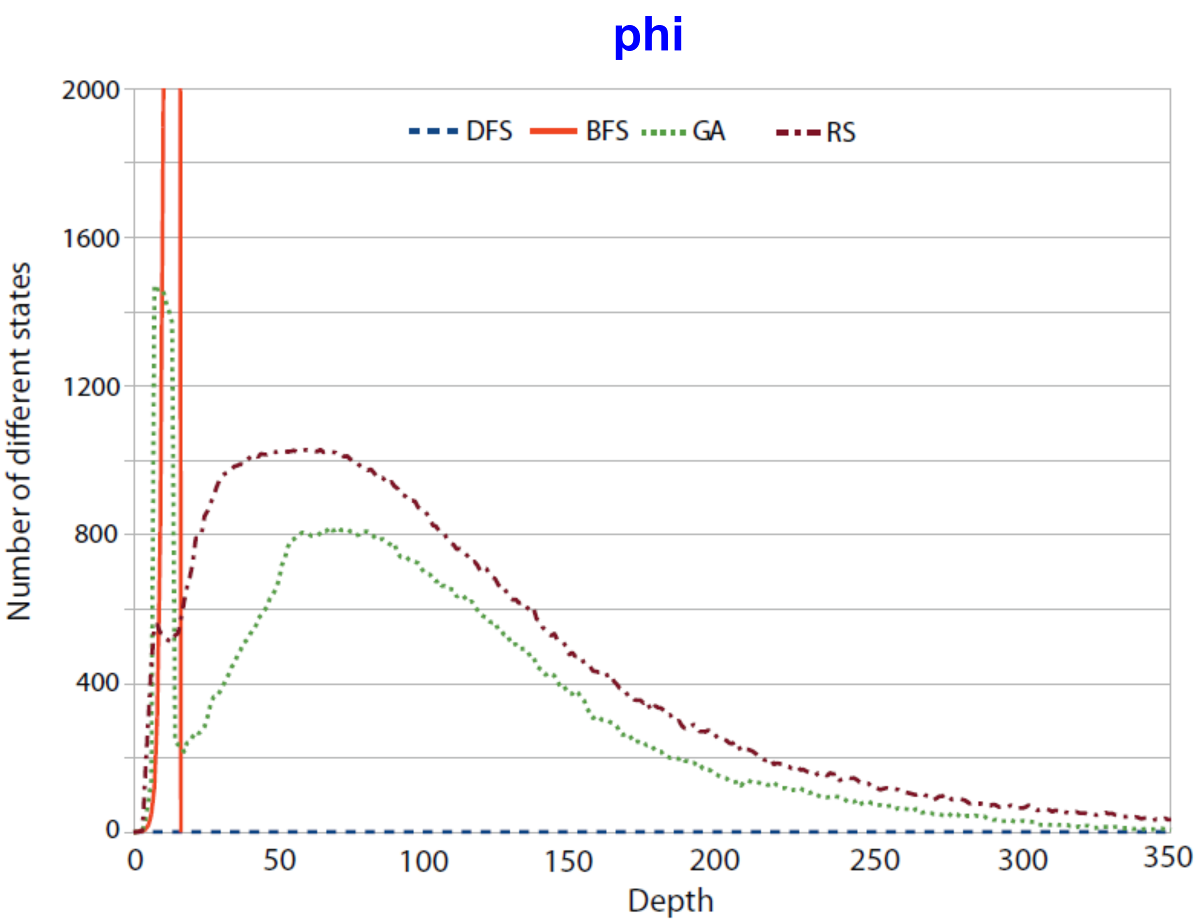
$$f(x) = deadlock + numblocked + \frac{1}{1 + pathlen}$$

- **100 independent executions of stochastic algorithms**

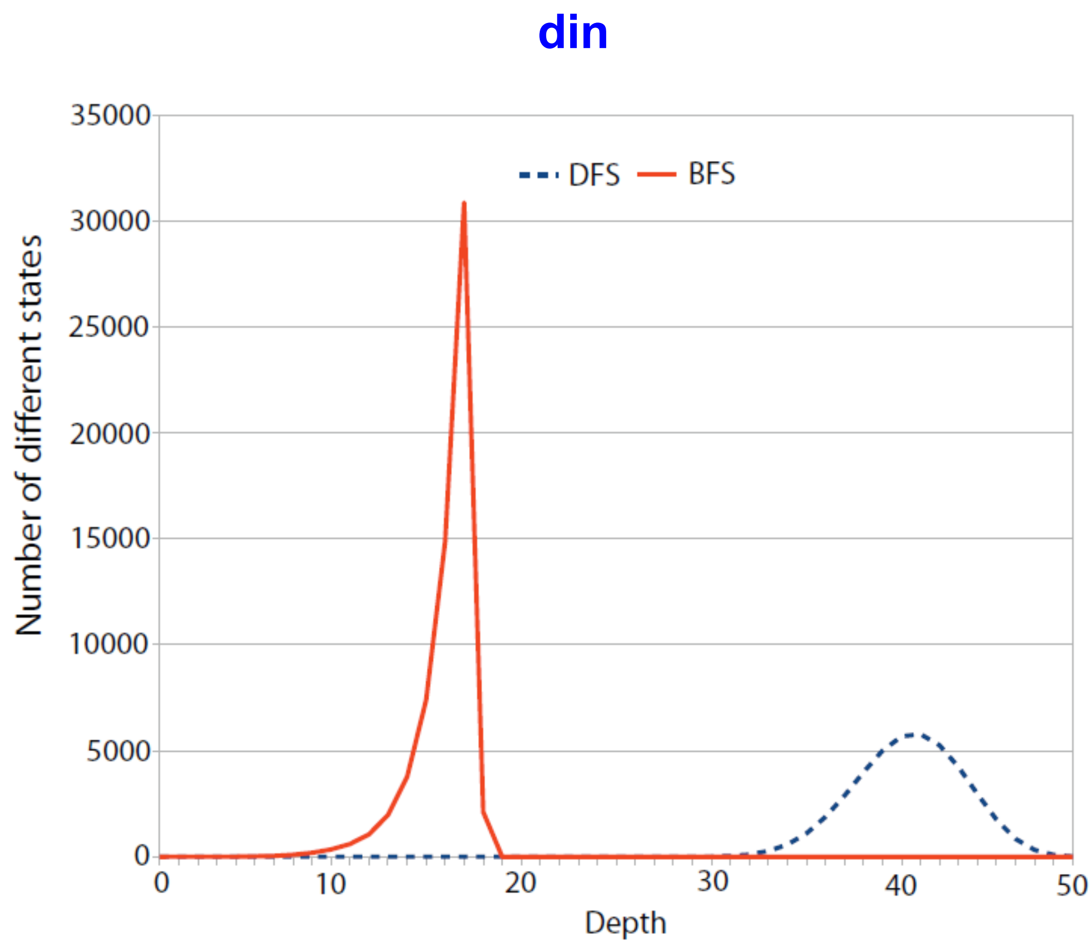
Hit rate

Problem	DFS	BFS	A*	GA	GAMO	PSO	SA	ACOhg	RS	BS
phi 4	100	100	100	100	100	100	100	100	100	100
phi 12	0	0	0	100	100	100	100	100	100	100
phi 20	0	0	0	100	100	100	100	100	100	100
phi 28	0	0	0	100	100	100	100	100	100	100
phi 36	0	0	0	82	100	53	79	100	100	100
din 4	100	100	100	100	100	100	100	100	100	100
din 8	100	0	0	100	100	100	76	100	96	100
din 12	100	0	0	100	96	85	13	68	0	100
din 16	0	0	0	91	58	20	0	2	0	100
din 20	0	0	0	52	24	0	0	0	0	100
mar 2	100	100	100	100	100	100	100	100	100	100
mar 4	100	100	100	100	100	100	96	100	100	100
mar 6	100	0	0	100	100	100	100	100	100	100
mar 8	100	0	0	100	95	100	100	100	100	100
mar 10	100	0	0	100	25	100	100	100	100	100
giop	100	0	0	100	68	100	100	100	100	100
garp	0	0	0	100	2	80	87	87	100	0

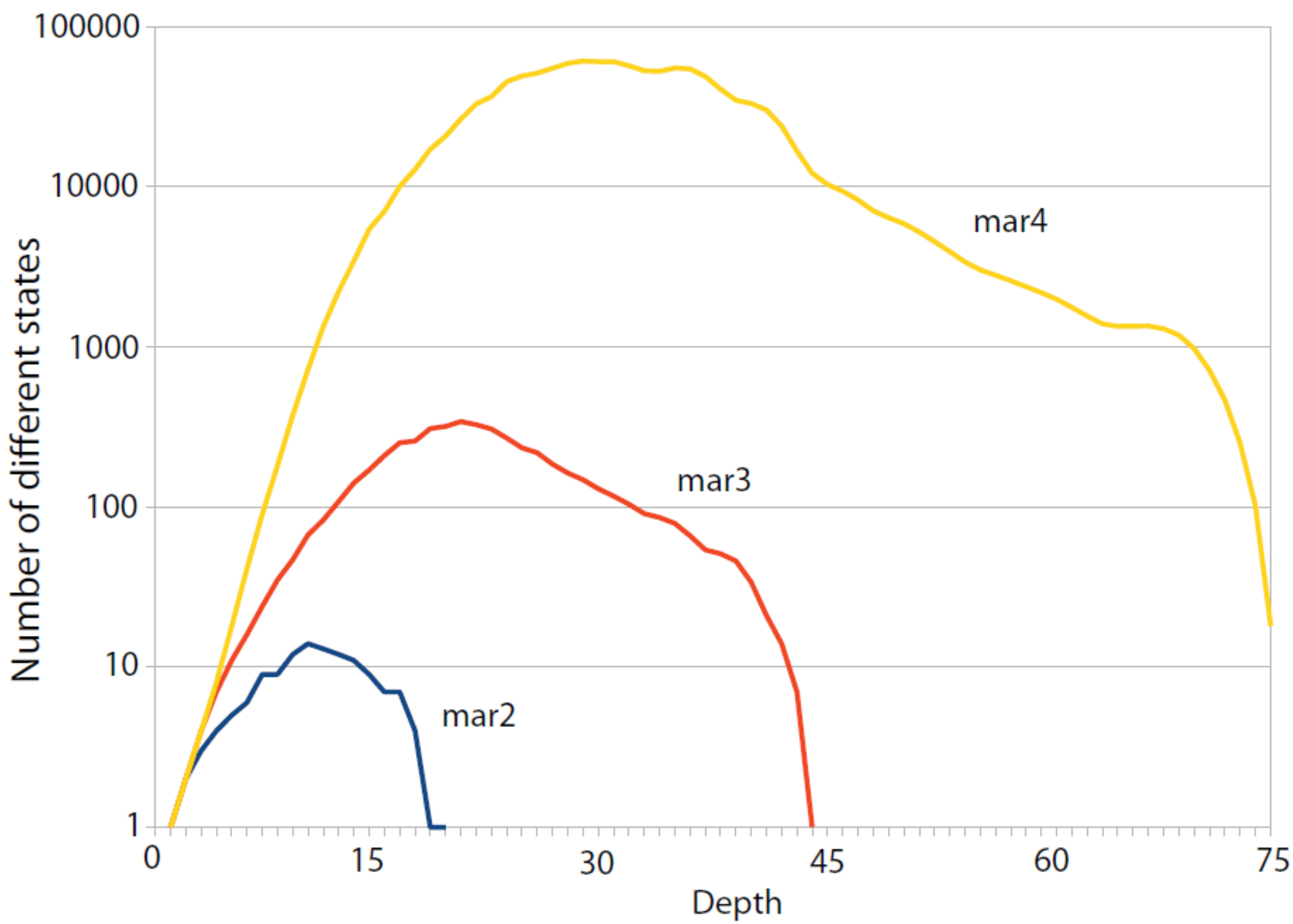
Hit rate


phi
DFS
BFS
GA
RS
2000
1600
1200
800
400
0
0
50
100
150
200
250
300
350
Number of different states
Depth

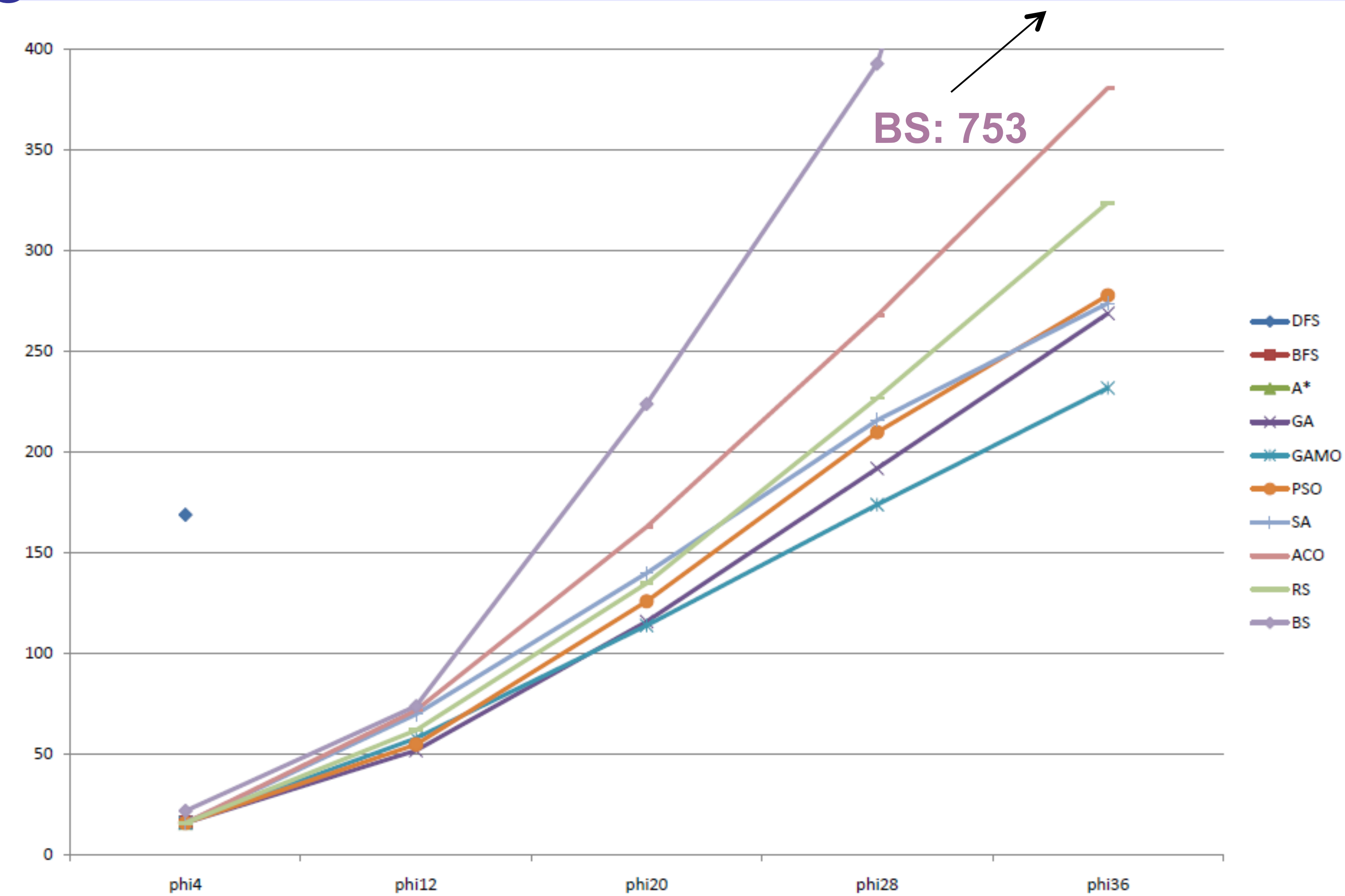
Hit rate


din
DFS
BFS
35000
30000
25000
20000
15000
10000
5000
0
0
10
20
30
40
50
Number of different states
Depth

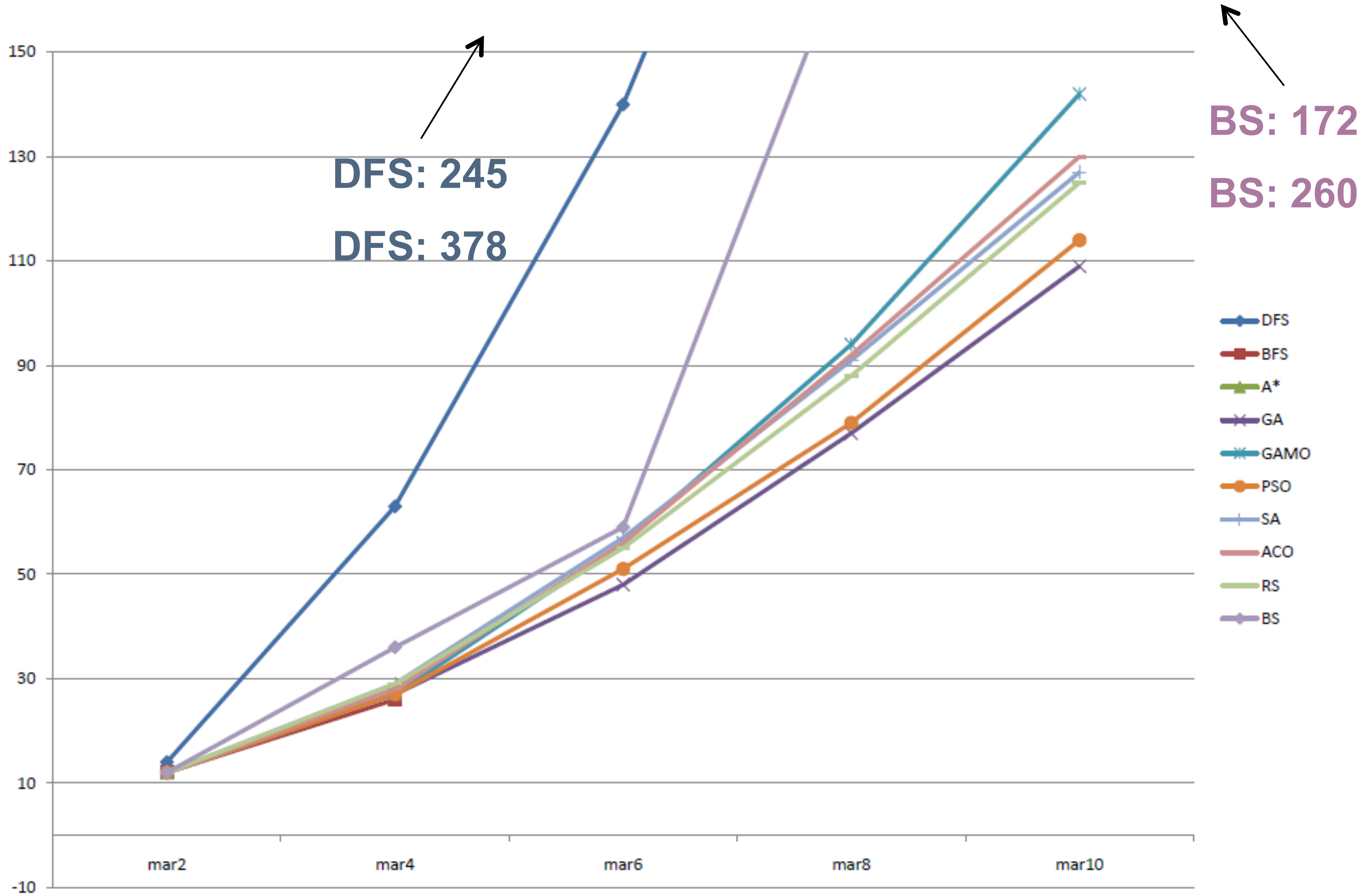
Hit rate


100000
10000
1000
100
10
1
Number of different states
0
15
30
45
60
75
Depth
mar4
mar3
mar2

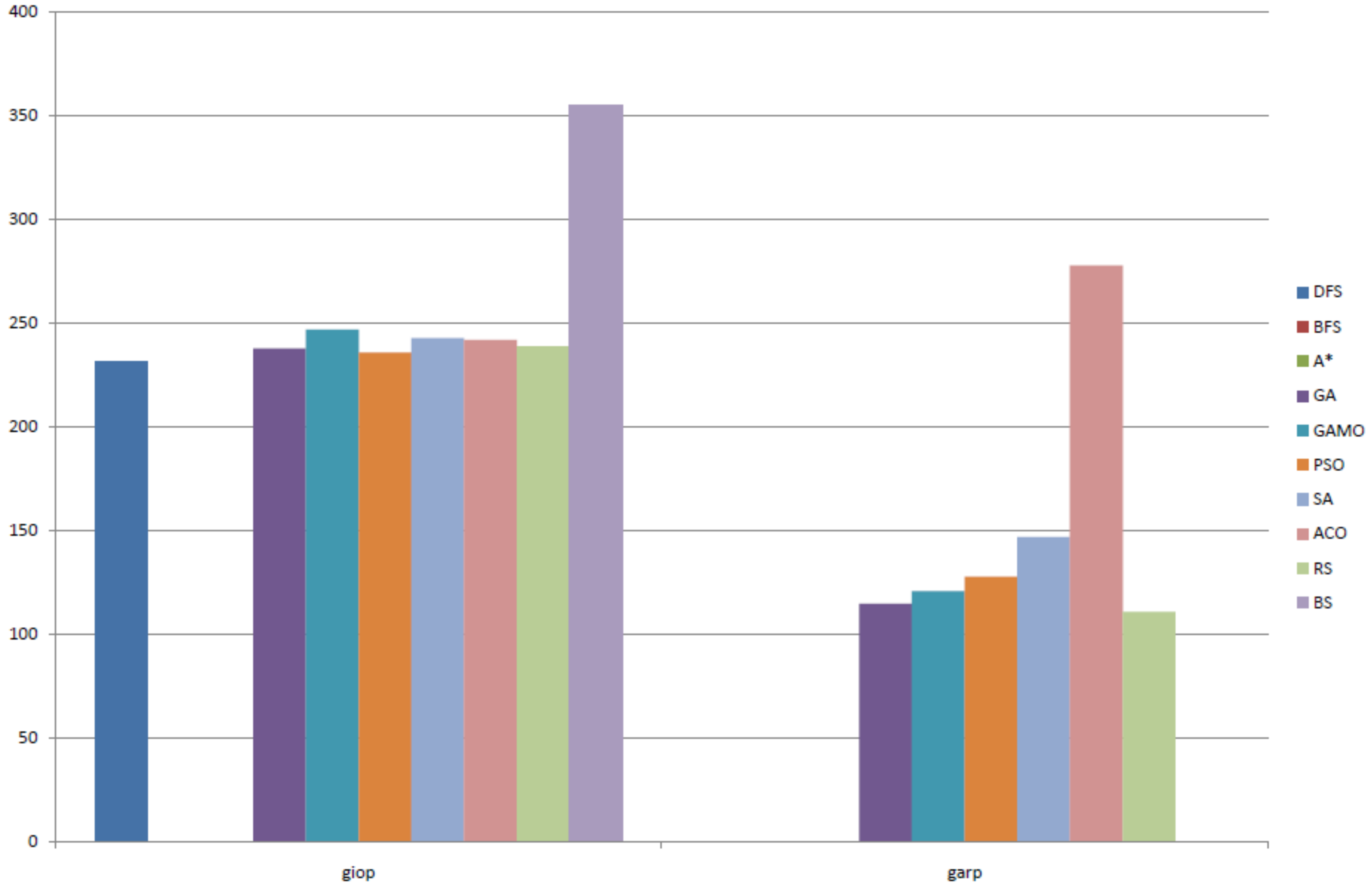
Length of Error Trails



Length of Error Trails



Length of Error Trails



Conclusions & Future Work

Conclusions

- Metaheuristics are more effective than classical algorithms in finding errors
- Beam Search has advantages over complete search algorithms
- An even distribution of the search in depth levels tends to raise hit rate
- Stochastic algorithms obtain short error trails

Future Work

- Design a stochastic complete guided algorithm to find errors and verify
- Design of hybrid algorithms to more efficiently explore the search space
- Explore the design of parallel metaheuristics for this problem

Thanks for your attention !!!