Applications of Model Reuse When Using Estimation of Distribution Algorithms to Test Concurrent Software

Jan Staunton and John Clark

University of York, SEBASE EPSRC project

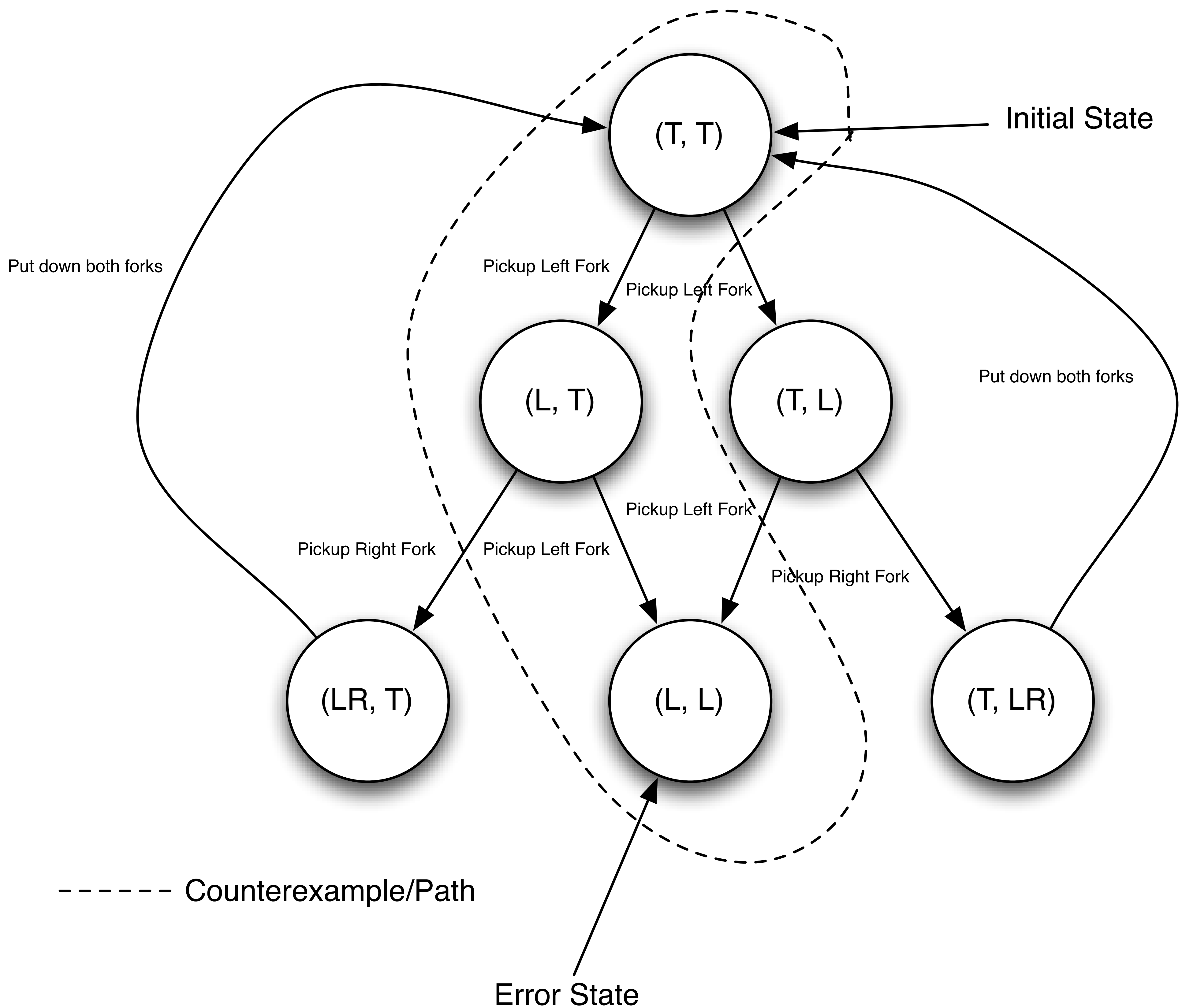
SSBSE 2011, 11th of September 2011

Presentation Outline

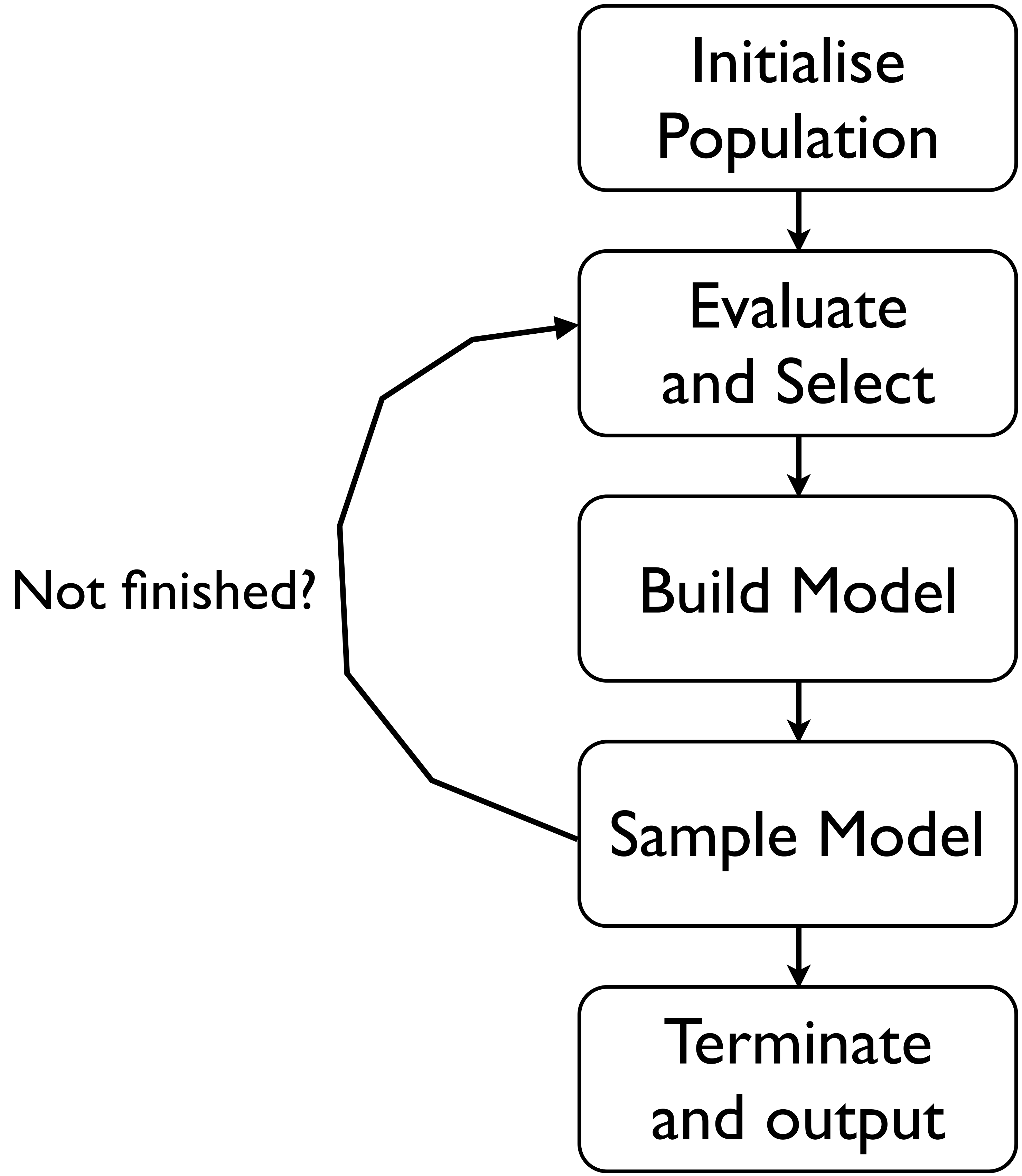
- Learning Strategies
- Potential Reuse Scenarios
- Experimentation
- Summary

Learning Strategies

Our Problem


(T, T)
Initial State
Put down both forks
Pickup Left Fork
Pickup Left Fork
(L, T)
(T, L)
Put down both forks
Pickup Left Fork
Pickup Right Fork
Pickup Left Fork
Pickup Right Fork
(LR, T)
(L, L)
(T, LR)
Counterexample/Path
Error State

Estimation of Distribution Algorithm



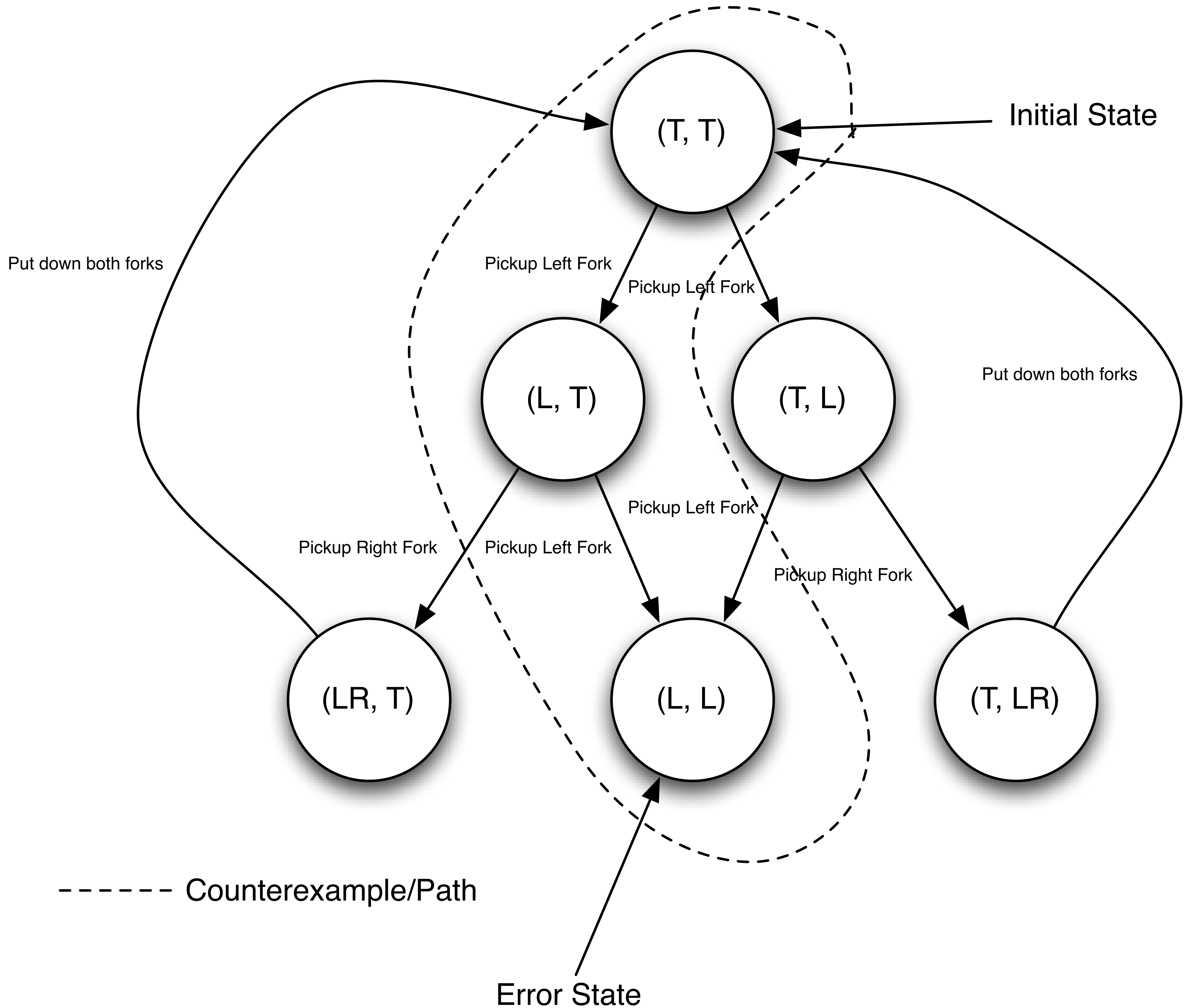
Our method

- Based upon an EDA called N-gram GP*
- Solution space is strings of actions that constitute paths in state space
- Paths start in the initial state, end in goal state, terminal state or previously encountered state
- Paths are sampled, and the best are used to build strategy. Strategy is then used to sample a new set of solutions from which the next strategy is constructed
- Strategy answers the following question...

* R. Poli and N.F. McPhee. A linear estimation-of-distribution GP system. Lecture Notes in Computer Science, 4971:206–217, 2008.

We construct a path *p* that starts in the initial state. Given the *n* most recent actions that have occurred on *p* currently under construction, by what distribution should the next action be selected?

Example


(T, T)
Initial State
Put down both forks
Pickup Left Fork
Pickup Left Fork
(L, T)
(T, L)
Put down both forks
Pickup Left Fork
Pickup Right Fork
Pickup Left Fork
Pickup Right Fork
(LR, T)
(L, L)
(T, LR)
Counterexample/Path
Error State

Strategies vs solutions

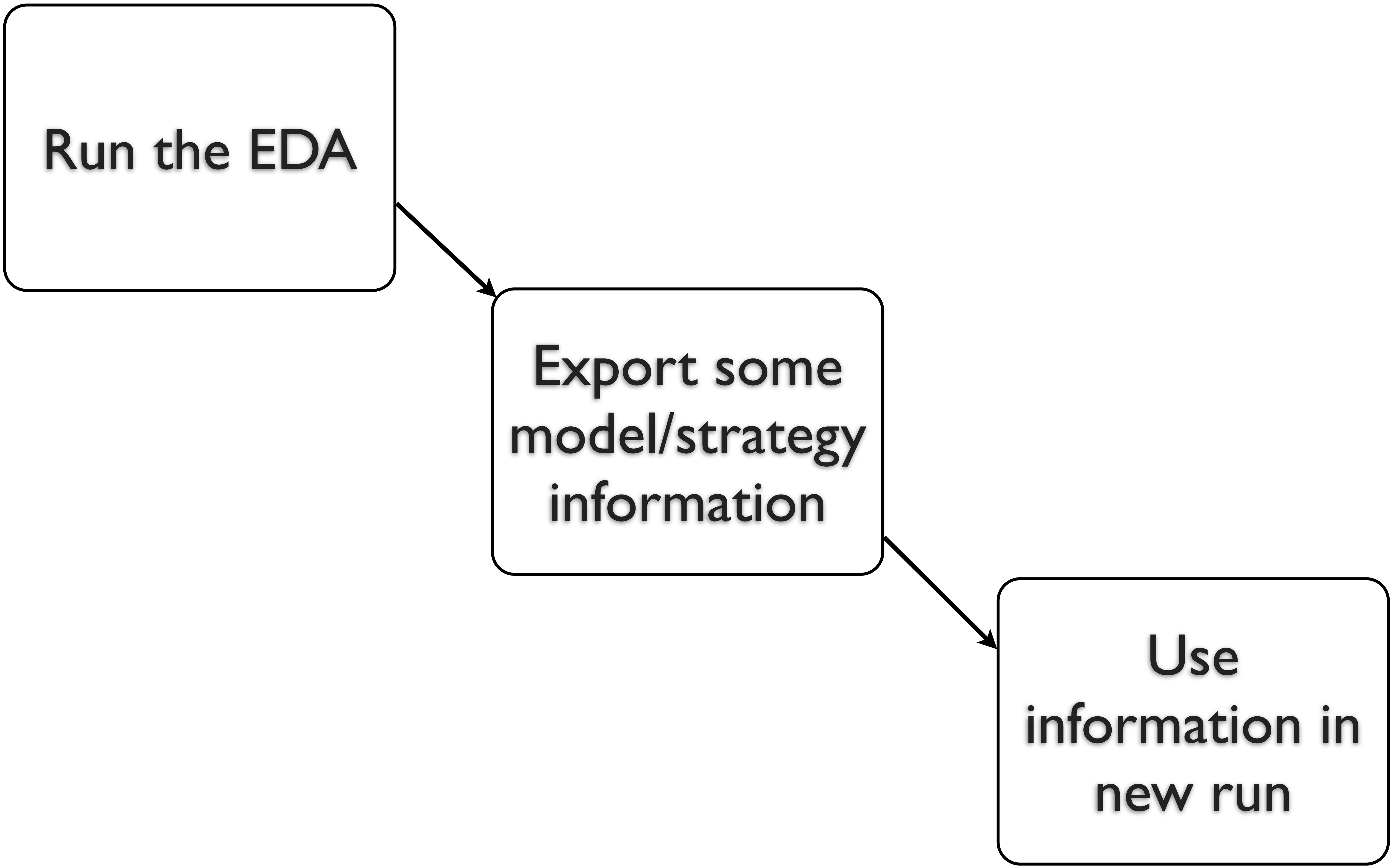
- Other mechanisms find sets of solutions only (search trajectory can reveal some insight)
- Our method learns a strategy for exploring the state space
- Solutions only have little scope for reuse, whereas our strategy can be used again in future runs (potentially on changed systems)

Reuse Scenarios

Model Reuse

- Often touted advantage of EDAs is making use of models either through analysis or reuse
- Because our EDA models action sequences, strategy can be generic over varying systems
- We are interested in reducing effort at certain stages in the development life cycle

Reuse workflow



Scenario: Debugging

- Use our EDA to find a concurrent fault *b*, found by strategy *s*
- “fix” the bug *b*, creating a new revision of the system
- Use *s* that found *b* to search revised system
- *s* may focus the search on multiple areas of the state space, at the very least will very likely lead to the now “fixed” area of the search space for a subsequent check. Can also tabu the strategy...

Scenario: Refinement

- Running the EDA on a system without errors can yield a strategy that highlights areas of the state space that "peak the interest" of the heuristic being used
- If the system is later refined (potentially with the changes between the previous and refined version of the system are linked/related) then strategies learned on the previous system could be used on the refined system

Scenario: Problem Families

- Some systems can be scaled to yield bigger state spaces (e.g. systems with more clients and servers)
- Assuming you can find an error with the EDA, one can use the strategy learned to find the same error in varying sizes of the same system
- Extra information about the same error can help a programmer to more effectively fix the bug
- Assumption is that a strategy that detects errors in a small model can be used to find errors in larger systems with the same description

Experimentation

Experimentation

- Tested the problem families scenario
- Implemented technique using HSF-SPIN and the ECJ toolkit
- Systems under test are PROMELA specifications
- Tested three systems, with deadlock, an assertion error and a liveness property violation

Method

- Run the EDA on a "small" instance of the system and save the strategy from the last generation (terminate after certain of states)
- Use this strategy to seed the first generation of a run on a large system
- Strategy is destroyed and rebuilt at each generation, in both runs
- Looking for effort reduction from the combined small and large run
- Compared against running the EDA without the seed

Test cases

- Dining Philosophers (no loop, eat and die, deadlock)
  - Small 32, Large 128
- Leader election system (assertion error)
  - Small 2, Large 10
- CORBA Global Inter-op Protocol (GIOP) (liveness issue)
  - Small 2, Large 20

Small instances

Measurement	Dining Philosophers	Leader	GIOP
First error:			
Generations	3	0	0
Path Length	34	35	59
States	73,058	35	729
Time	27.45s	0.3s	0.3s
Best error:			
Generations	3	0	17
Path Length	34	32	21
States	73,058	2,080	80,478
Time	27.45s	0.63s	3m8s
Total for run:			
Generations	50	200	200
States	1,150,400	1,040,495	931,691
Time	13m30s	19m47s	37m33s

Dining Philosophers (128)

Measurement	Without Model Reuse	With Model Reuse	Without Initial Run
First error:			
Generations	19/19.4(+)	**3/3**	**0/0**
Path Length	**130/130(-)**	**130/130**	**130/130**
States	1,831,394/1,898,568.21(+)	**73,831/74,281.1**	**773/1,223.1**
Time	47m24s/1h14m32s(+)	**29.572s/30.057s**	**2.122s/2.606s**
Best error:			
Generations	19/19.4(+)	**3/3**	**0/0**
Path Length	**130/130(-)**	**130/130**	**130/130**
States	1,831,394/1,898,568.21(+)	**73,831/74,281.1**	**773/1,223.1**
Time	47m24s/1h14m32s(+)	**29.572s/30.057s**	**2.122s/2.606s**

99% reduction in effort

Leader (10)

Measurement	Without Model Reuse	With Model Reuse	Without Initial Run
First error:			
Generations	**0/0(-)**	**0/0**	**0/0**
Path Length	84/82.75(+)	**71/71.21**	**71/71.21**
States	**84/82.75(+)**	2,151/2,151.21	**71/71.21**
Time	**0.239s/0.622s(+)**	1.127s/1.606s	0.497s/0.976s
Best error:			
Generations	17/20.26(-)	**15/19.23**	**15/19.23**
Path Length	**36/35.45(-)**	**36/35.47**	**36/35.47**
States	193,616/225,050.01(-)	**163,429/209,150.82**	**161,349/207,070.82**
Time	22m51s/25m57s(+)	**4m7s/5m19s**	**4m6s/5m18s**

75+% reduction in effort

CORBA GIOP (20)

Measurement	Without Model Reuse	With Model Reuse	Without Initial Run
First error:			
Generations	**0/0.01(+)**	**17/17**	**0/0**
Path Length	132/150.09(+)	**61/73.37**	**61/73.37**
States	**40,421/60,681.01(+)**	90,773/98,194.14	**10,295/17,716.14**
Time	**1m26s/2m1s(+)**	3m28s/3m46s	**19.56s/38.017s**
Best error:			
Generations	30/28.71(+)	**20/28.21**	**3/11.21**
Path Length	31/31.21+)	**26/25.6**	**26/25.6**
States	13,068,139/12,337,306(+)	**1,495,644/4,942,260.07**	**1,415,166/4,861,782.07**
Time	6h47m16s/8h13m24s(+)	**57m34s/3h12m14s**	**54m26s/3h9m6s**

68% reduction in mean effort, higher quality results

Discussion

- Two errors of different sizes, and the EDA has optimised both errors (shorter paths to error are better)
- One can instantly learn about the error from the path lengths
- This process can be fully automated
- Effort saved allows for overnight runs as opposed to week-long runs on large systems

Summary

Summary

- Outlined methods of reusing strategy information between runs of an EDA
- Novel approach, something perhaps unique to EDAs
- Proven that it has the potential to reduce the effort required to gain extra information about errors in problem families
- Model/strategy reuse meme has the potential to be useful elsewhere

Thanks!

Any questions?