Randomized Backtracking: Next Steps

Pavel Parízek, Ondřej Lhoták

University of Waterloo

Outline

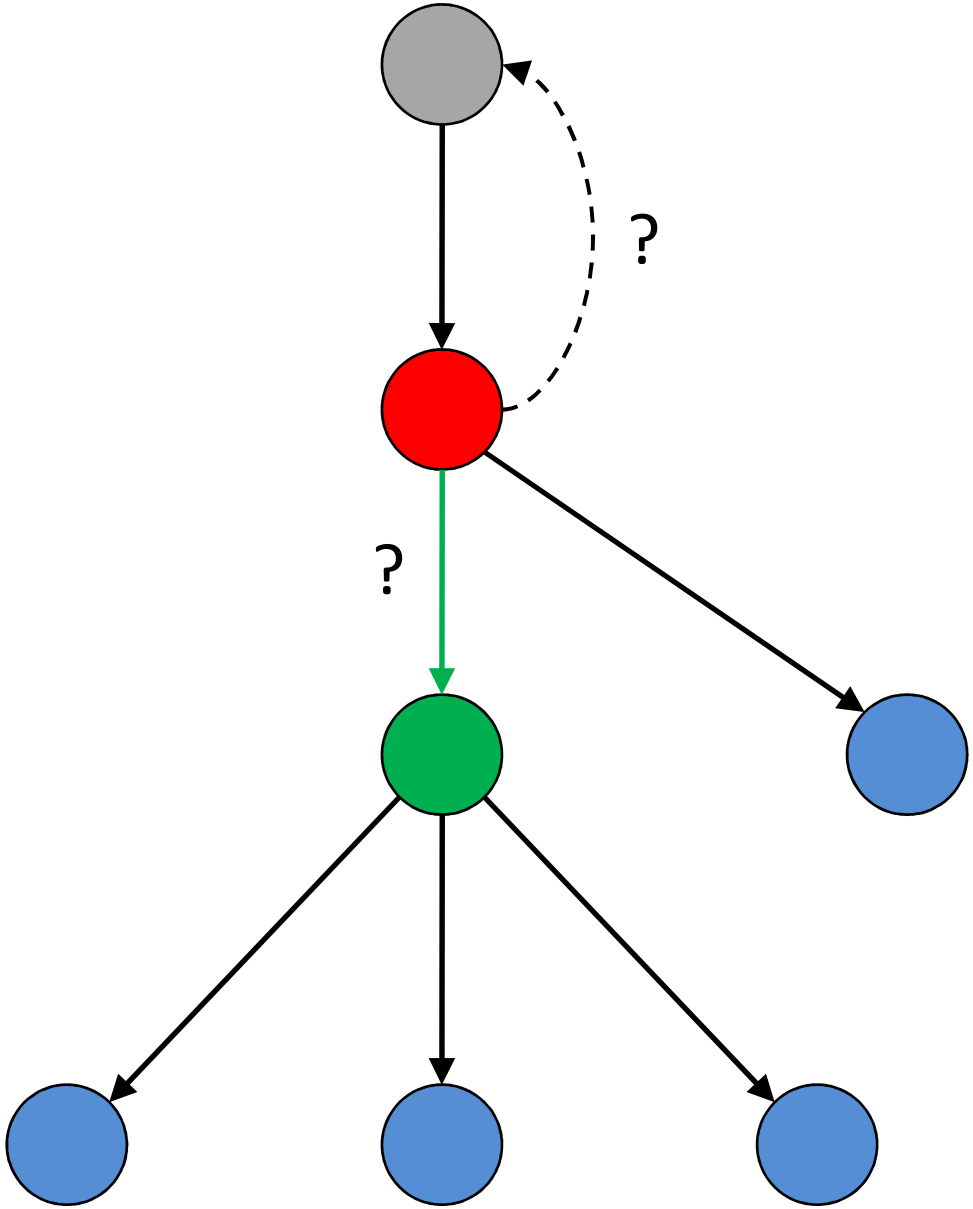
- Randomized backtracking [SPIN 2011]
  - Basic algorithm
  - Experiments

- Future plans
  - Optimizations
  - Extensions

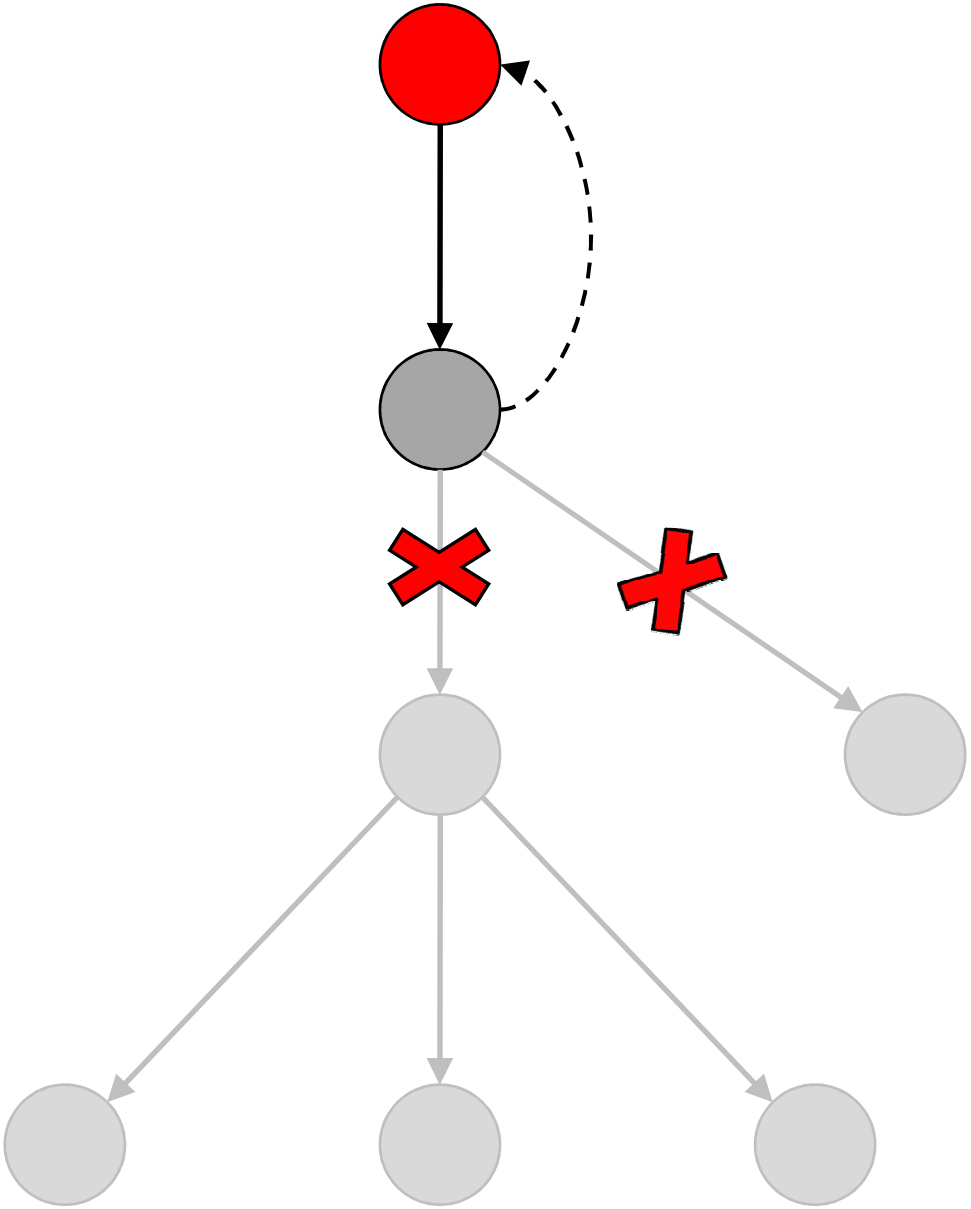
Randomized backtracking

- Key ideas
  - Backtracking from states with unexplored transitions
  - Decision whether to backtrack influenced by random number choice

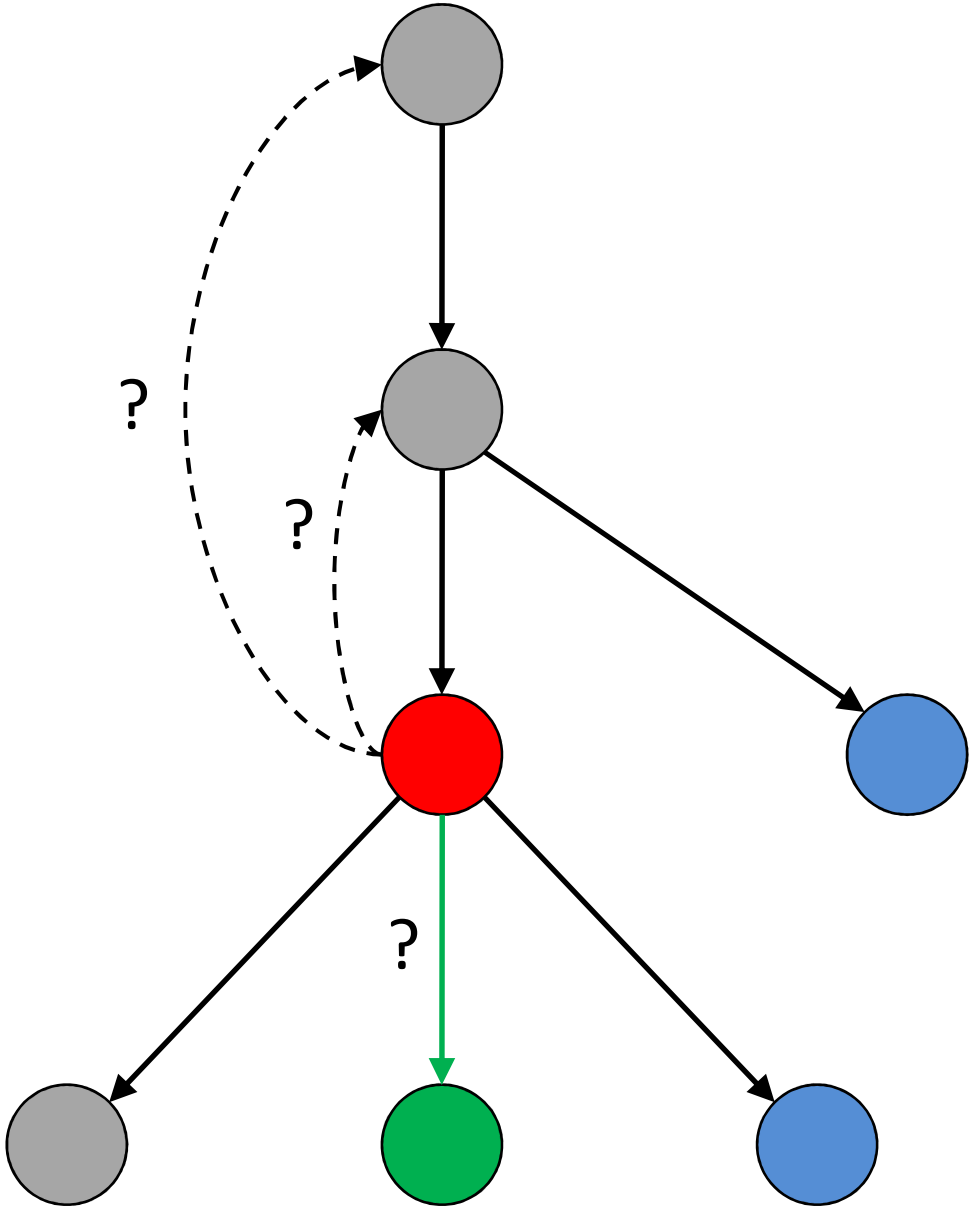
DFS with randomized backtracking



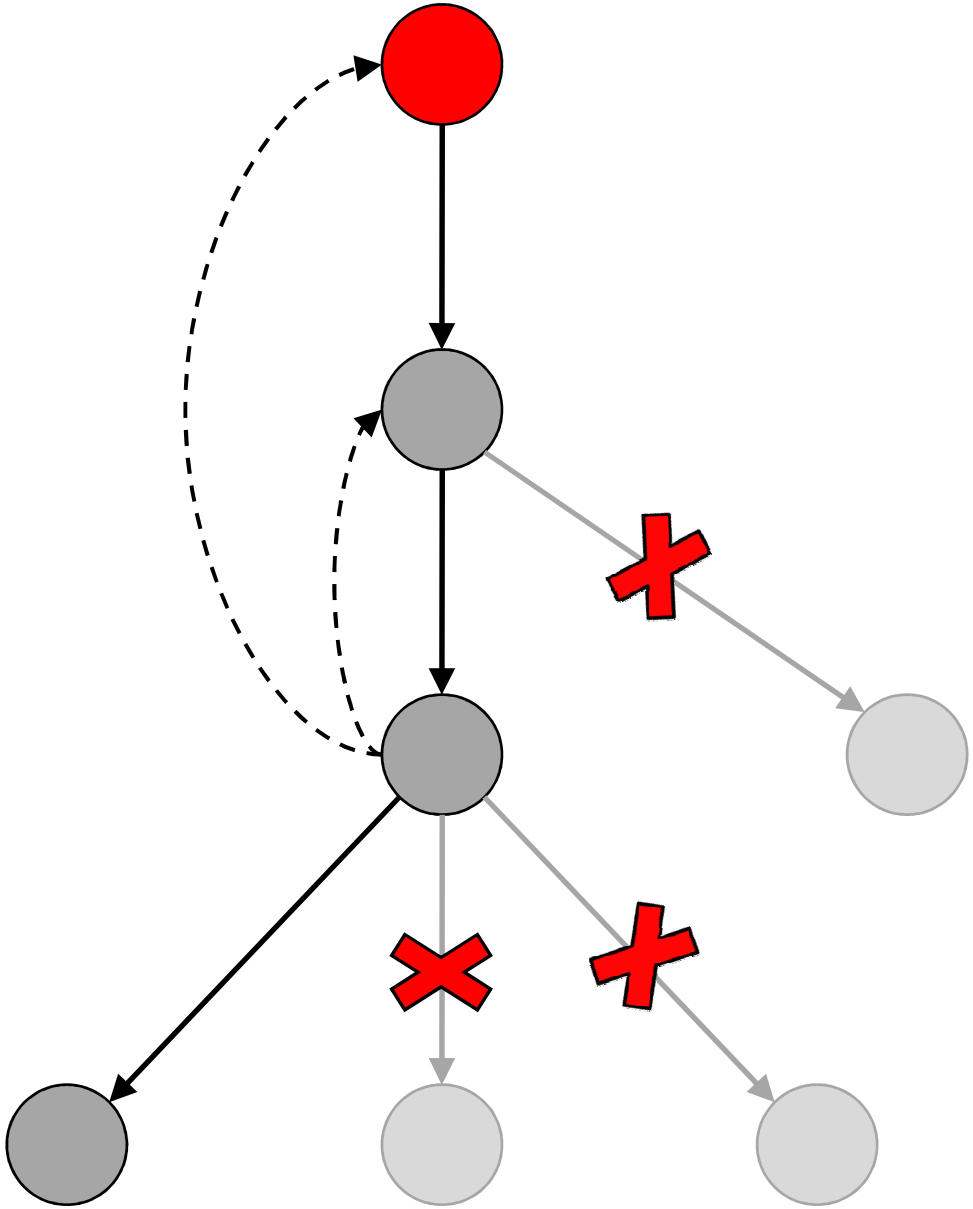
DFS with randomized backtracking



DFS with randomized backtracking



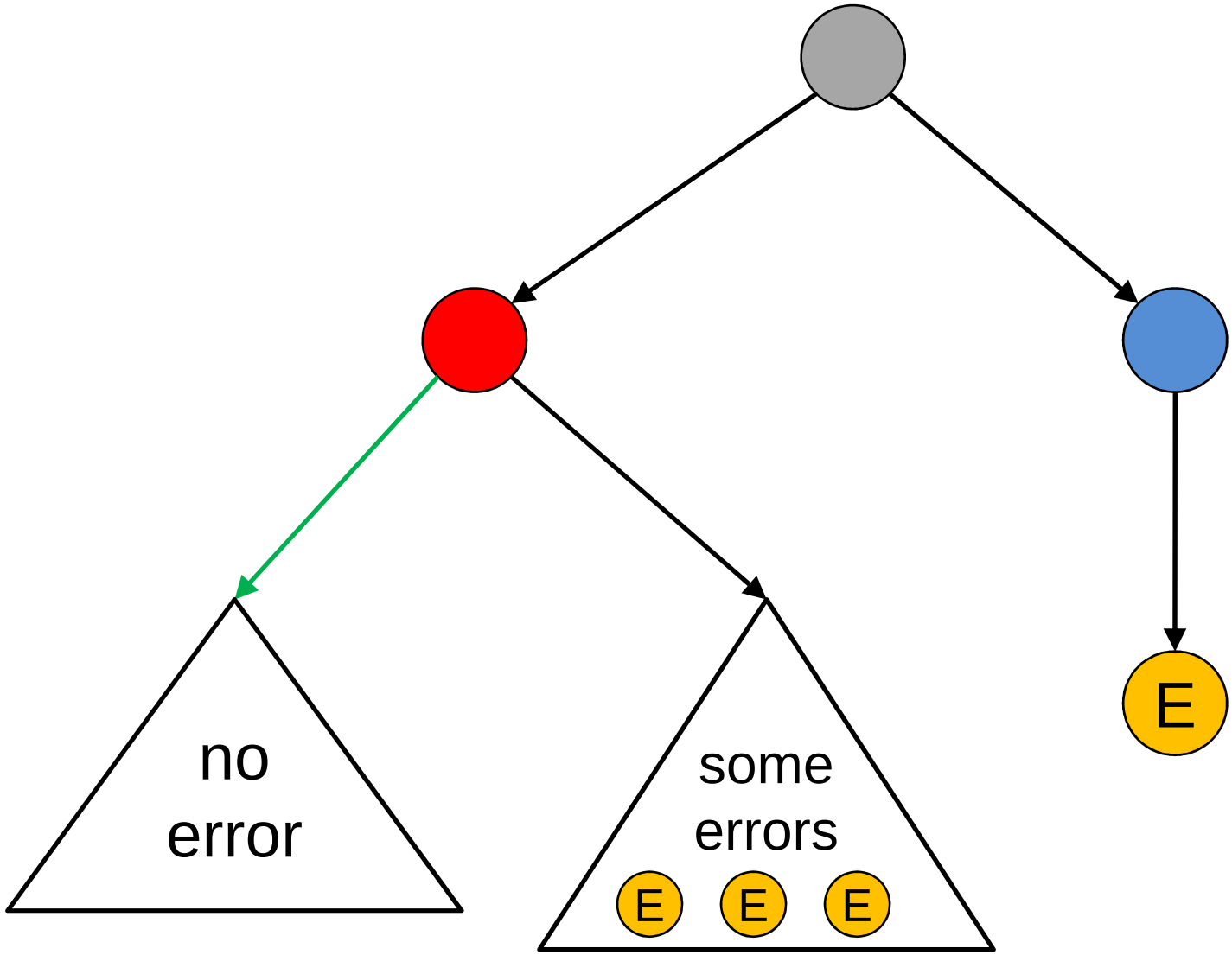
DFS with randomized backtracking



DFS with randomized backtracking

```
procedure explore (s)
  if error(s) then counterexample = stack; terminate; end if
  transitions = order( enabled(s) )
  for tr ∈ transitions do
    depth = size(stack)
    if depth >= threshold then
      if rnd(0,1) > ratio(depth) return
    end if
    s' = execute(tr)
    if s' ∉ visited then
      visited = visited ∪ s'
      push(stack, s'); explore(s'); pop(stack)
      if backtrackAgain(strategy) return
    end if
  end for
end proc
```

State space pruning



Implementation

- Extension for Java Pathfinder (JPF)
  - Custom search driver
- Input
  - Configuration: tuple (threshold, strategy, ratio)

Experimental evaluation: benchmarks

Program	LoC	Threads
Daisy file system	1150	2
Elevator	320	4
Alarm Clock	180	3
Linked List	185	2
Producer Consumer	135	7
RAX Extended	150	5
Replicated Workers	430	6

- Source
  - Daisy: CAV/ISSTA 2004 challenge
  - Elevator: Parallel Java Benchmarks
  - Other programs: CTC repository

Results: performance

Benchmark	Configuration	# states explored			Runs found error
		AVG	MIN	MAX	
Elevator	random search order	2399	1062	3833	100 %
	(50, random, 1–d/100)	**270**	**255**	**293**	**100 %**
RAX Extended	thread interleavings	104			100 %
	(10, luby, 0.99)	97	86	113	100 %
	(5, luby, 0.9)	**8**	**8**	**9**	**59 %**
Producer Consumer	random search order	48	25	73	100 %
	(10, fixed, 1-d/100)	57	23	157	100 %

9 x

12 x

- best existing technique
- best configuration of randomized backtracking with 100 %
- best configuration of randomized backtracking where less than 100 % runs found error

Results: variability

Benchmark	Configuration	# states explored			Runs found error
		AVG	MIN	MAX	
Elevator	(50, fixed, 0.99)	358	253	424	100 %
	(100, fixed, 1–d/50)	2263	2227	2338	100 %
	(5, fixed, 0.5)	-	-	-	**0 %**
Linked List	(5, fixed, 1-d/50)	74	38	133	56 %
	(20, fixed, 0.9)	235	170	408	100 %
	(5, luby, 0.5)	-	-	-	**0 %**
Replicated Workers	(50, fixed, 0.5)	339	**71**	1282	100 %
	(50, luby, 1-d/20)	6258	**139**	**19190**	100 %

lowest minimum highest maximum fewest errors found

Results: “good” configuration

Fixed strategy and ratio: (?, random, 0.9)

Benchmark	Configuration	# states explored			Runs found error
		AVG	MIN	MAX	
Elevator	random search order	2399	1062	3833	100 %
	(50, random, 1–d/100)	270	255	293	100 %
	(20, random, 0.9)	-	-	-	0 %
	(50, random, 0.9)	290	261	312	100 %
	(100, random, 0.9)	2185	2170	2213	100 %
Daisy file sys	thread interleavings	139			100 %
	(10, random, 0.9)	110	103	119	100 %

best existing technique

best configuration of randomized backtracking

“good” configuration of randomized backtracking with different threshold

Future plans

Search algorithms

- Breadth-first search (BFS)
  - Random selection from queue
  - Different meaning of parameters
- Mixed search algorithms

- Optimized DFS
  - Thread choices
  - Different strategies

Parameter values

- Goals
  - Good **performance**
  - Small **variability**

- Options
  - Good threshold values for (?, random, 0.9)
  - Dynamically changing parameter values

Complete verification

- No error ➔ bounded verification + random walks
- Measuring coverage: very hard
- Idea
  - Exploring pruned transitions later

Experiments

- Goals
  - Combination with other techniques
  - Large multi-threaded programs

- Problem: **finding benchmarks**
  - Reasonable complexity
  - Concurrency errors
  - Supported libraries (MJI)

- **Public repository: ??**

Questions ?