Low-Connectivity State Space Exploration using Swarm Model Checking on the GPU

Leonard Techel

October 7, 2021

Table of Contents

Motivation

Motivation 1/2: Waypoints Benchmark Model

- Current state: Represented by a 32 bit integer
- Eight concurrent processes, each in control of four bits
- Each process toggles one random bit on successor generation
- **Goal:** Find 100 random uniform distributed 32 bit integers (Waypoints)

P1	P2	P3	P4	P5	P6	P7	P8
0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0

Figure 1: Waypoints model state

Motivation 1/2: Waypoints Benchmark Model

- Current state: Represented by a 32 bit integer
- Eight concurrent processes, each in control of four bits
- Each process toggles one random bit on successor generation
- **Goal:** Find 100 random uniform distributed 32 bit integers (Waypoints)

P1	P2	P3	P4	P5	P6	P7	P8
0 0 0 1	0 0 1 0	0 1 0 0	0 1 0 0	0 0 0 1	1 0 0 0	0 0 0 1	0 0 1 0

Figure 1: Waypoints model state

Motivation 1/2: Waypoints Benchmark Model

- Current state: Represented by a 32 bit integer
- Eight concurrent processes, each in control of four bits
- Each process toggles one random bit on successor generation
- **Goal:** Find 100 random uniform distributed 32 bit integers (Waypoints)

P1	P2	P3	P4	P5	P6	P7	P8
1 0 0 1	0 0 1 1	0 1 1 0	0 0 0 0	0 1 0 0	1 0 1 0	0 1 0 1	0 0 0 0

Figure 1: Waypoints model state

Motivation 2/2: Observations

- **Model Checking:** Verify whether a transition system satisfies a specification
- We have $2^{32} = \left(2^4\right)^8 = 4\,294\,967\,296$ states to check
- **Problem:** State Space Explosion
- Exhaustive verification cannot be completed in reasonable time
- Storing all states in memory takes up $2^{32} \cdot 32\,\text{bit} = 17\,\text{GB}$
- What can we do?

Concepts: Swarm Verification, Grapple Framework, Low-Connectivity Models

Swarm Verification 1/2

From the paper *Swarm Verification* by G. J. Holzmann et al. [2]

- Approach on explicit-state model checking
- **Idea:** Split state space exploration into small, independent verification tests (VTs)
- Each VT only covers a subset of the state space
- **Result:** VTs can be massively parallelized, for example on the GPU
- How is the state space exploration split up?

Swarm Verification 2/2: Diversification Techniques

Diversification techniques:

- **State pruning using hash collisions**
- Reversing search order
- Randomizing order of nondeterministic choice
- ...

How to implement Swarm Verification on the GPU?

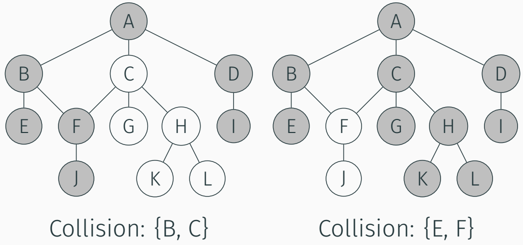


Figure 2: State pruning using hash collisions

Grapple Model Checking Framework

From the paper *Swarm model checking on the GPU* by R. DeFrancisco et al. [1]

- A framework for parallel Swarm Verification model checking on the GPU using CUDA
- Why GPUs? — GPUs are massively parallel, widely available and cheap
- Each VT executes a parallel breadth-first search

What are the limitations?

Low-Connectivity Models 1/2: Definition

A model has *low connectivity* if at least one of the following properties is satisfied:

- **Generally Linear**: The average # of edges per state is close to 2: One inbound, one outbound
- **Bottleneck Structures**: A single state or group of states other than the initial state that needs to be passed to reach most of the state space

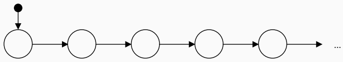


Figure 3: Example of a generally linear graph

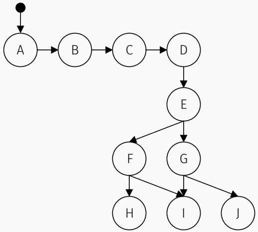


Figure 4: Example of a bottleneck structure

Low-Connectivity Models 2/2: What to do?

- According to the paper, coverage slows down: Less unique states visited
- Questions proposed in the intermediate presentation:
    1. How can we (automatically) classify a model as low-connectivity?
    2. How can we improve the state space exploration of such models?
- How can we see the effect? $\rightarrow$ Count unique states visited

Contributions: Estimating Unique States Visited, Start Overs

Estimating Unique States Visited 1/2

- Verification Tests may overlap in covered state space
- **Unique States Visited:** Number of unique states visited across all VTs
- **Total States Visited:** Number of states visited by all VTs, including duplicates

Estimating Unique States Visited 2/2

- **Goal:** Observe a swarm's progress in state space coverage
  - How close are we to 100 % state space coverage?
  - How does the exploration of low-connectivity models perform?

Estimating Unique States Visited 2/2

- **Goal:** Observe a swarm's progress in state space coverage
  - How close are we to 100 % state space coverage?
  - How does the exploration of low-connectivity models perform?
- **Challenge:** Distributed counting of distinct states across all VTs
  - VTs are independent of each other
  - VTs may overlap in explored state space
  - Each VT visits up to 2^{18} states, we have >100 000 VTs

Estimating Unique States Visited 2/2

- **Goal:** Observe a swarm's progress in state space coverage
  - How close are we to 100 % state space coverage?
  - How does the exploration of low-connectivity models perform?
- **Challenge:** Distributed counting of distinct states across all VTs
  - VTs are independent of each other
  - VTs may overlap in explored state space
  - Each VT visits up to 2^{18} states, we have >100 000 VTs
- **Approach:** Probabilistic Cardinality Estimation using HyperLogLog++
  - Estimation with fixed error
  - Example: Estimate cardinalities $> 10^9$ with a standard error of 2 % using 1.5 kB of memory

Start Overs 1/2

- **Observation:** All VTs start at the initial state and terminate after $2^{18} = 262\,144$ states due to a full hash table
- **Challenge:** How can we reach deeper states?
- **Approach:** Start Overs as an extension to the Grapple breadth-first search

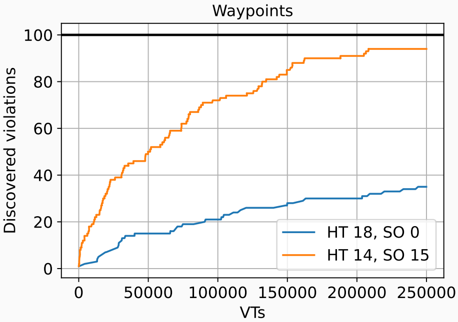
Start Overs 2/2

- Self-Contained within a VT
- Keep the last few visited states
- When the VT's search terminates, use last visited states as new initial states
- **Result:** We can use an even smaller hash table by adding start overs

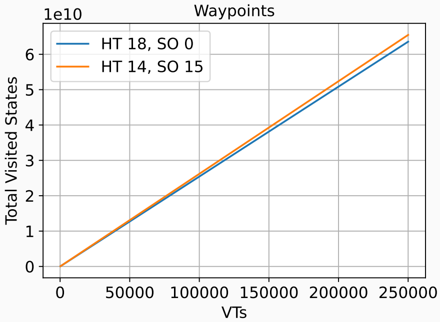
Do start overs increase the state space coverage?

Experimental Results

Start Overs on Waypoints and Low-Connectivity Models 1/2



(a) Comparison of discovered waypoints



(b) Comparison of total visited states

Figure 5: State space exploration with start overs. 250 000 VTs

HT: Hash table size, as a power of two. SO: Number of start overs

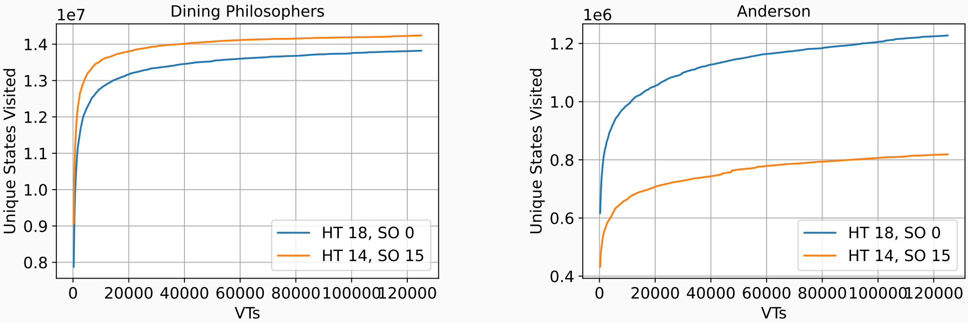


Figure 6: Start over strategy on low-connectivity models. 125 000 VTs

HT: Hash table size, as a power of two. SO: Number of start overs

Unique and Total States Visited on Low-Connectivity Models

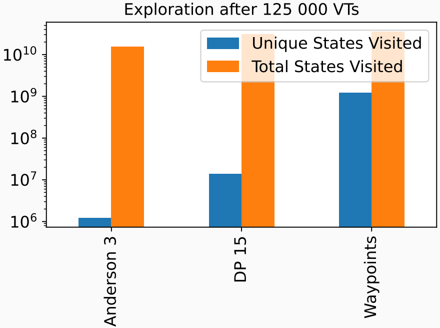


Figure 7: Exploration of low-connectivity models

- **Question:** Do Low-Connectivity models behave differently in terms of USV and TSV?
- **Logarithmic scale!**
- Anderson 3: Less than half the total visited states as the waypoints model
- Both low-connectivity models: Unique State Hit-Rate <0.1 %

BFS Frontiers 1/4

- Breadth-first search: Produces *frontiers*
- Current frontier: Deepest states found until now
- Next frontier: Successors of current frontier that are not visited before

- Breadth-first search: Produces *frontiers*
- Current frontier: Deepest states found until now
- Next frontier: Successors of current frontier that are not visited before
- **Idea:** For each frontier: Count *visited* (= not seen before) and *failed* (= already seen) states
- **Expectation:** Visited states quickly build up from the single initial state, then decay due to the hash table filling up

BFS Frontiers 2/4: Waypoints model

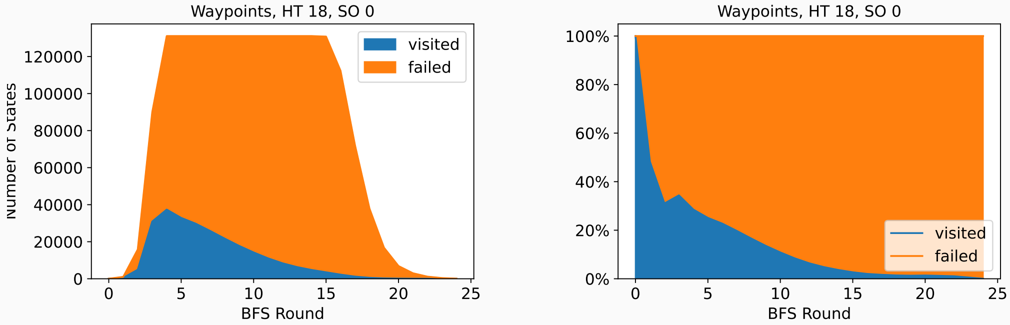


Figure 8: BFS frontier visualization of the waypoints model

visited: not seen before. failed: already seen

BFS Frontiers 3/4: Dining Philosophers

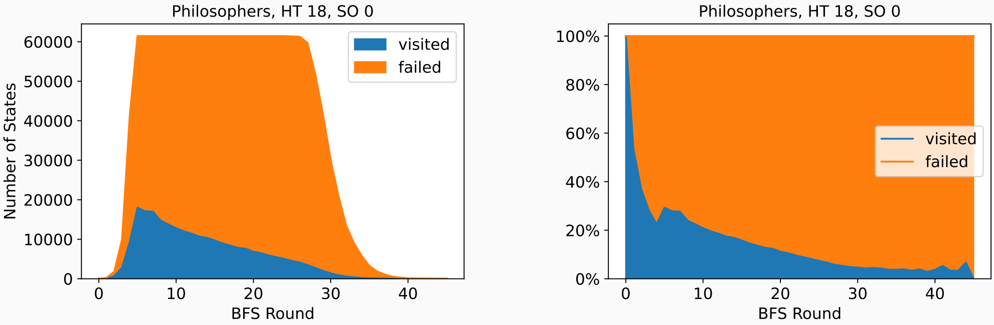


Figure 9: BFS frontier visualization of the dining philosophers model

visited: not seen before. failed: already seen

BFS Frontiers 4/4: Anderson model

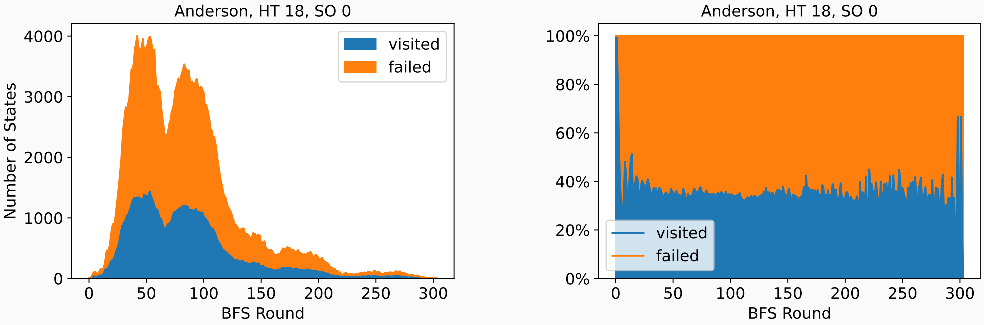


Figure 10: BFS frontier visualization of the Anderson 3 model

visited: not seen before. failed: already seen

Conclusions

Conclusions

- Successfully implemented a Grapple swarm model checker
- Successfully estimated unique states visited
- Experiments showed promising characteristics on low-connectivity models
- One notable exception: Anderson model

Thesis, Model Checker, and Experiments: github.com/barnslig/bachelor-thesis

References

- Richard DeFrancisco et al. “Swarm model checking on the GPU”. In: *International Journal on Software Tools for Technology Transfer* 22.5 (Oct. 2020), pp. 583–599. ISSN: 1433-2787. DOI: `10.1007/s10009-020-00576-x`.
- Gerard J. Holzmann, Rajeev Joshi, and Alex Groce. “Swarm Verification”. In: *2008 23rd IEEE/ACM International Conference on Automated Software Engineering*. 2008, pp. 1–6. DOI: `10.1109/ASE.2008.9`.