Understanding the Power of Evolutionary Computation for GPU Code Optimization

Jhe-Yu Liou[1], Muaaz Awan[2], Steven Hofmeyr[2], Stephanie Forrest[1], Carole-Jean Wu[1]

[1]Arizona State University

[2]Lawrence Berkeley National Laboratory

Ira A. Fulton Schools of Engineering
ARIZONA STATE UNIVERSITY

BERKELEY LAB

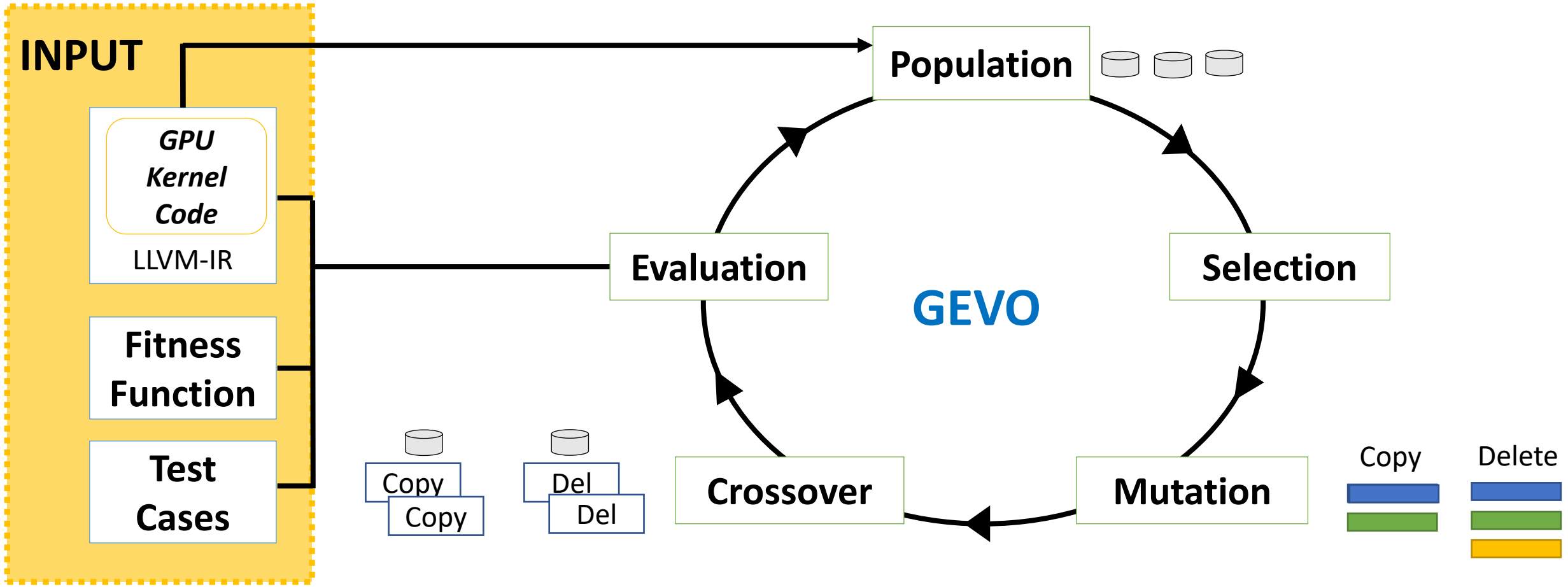
Introduction

- GPU speeds up important applications but is hard to be fully utilized
- Automated code optimization
  - GEVO: **G**PU Code **Evo**lution
- Research questions
  - What optimizations can such a method find?
  - How well does it perform on hand-tuned applications?
  - How can the method inform GPU developers?
- Deploying GEVO on
  - Highly computation-intensive workloads
  - Different development stages in workloads

Outline

- Optimization Tool: GEVO
- Workloads: ADEPT, SIMCoVGPU
- Experimental Setup and Results
- Edit and Optimization Analysis
- Limitation and Conclusion

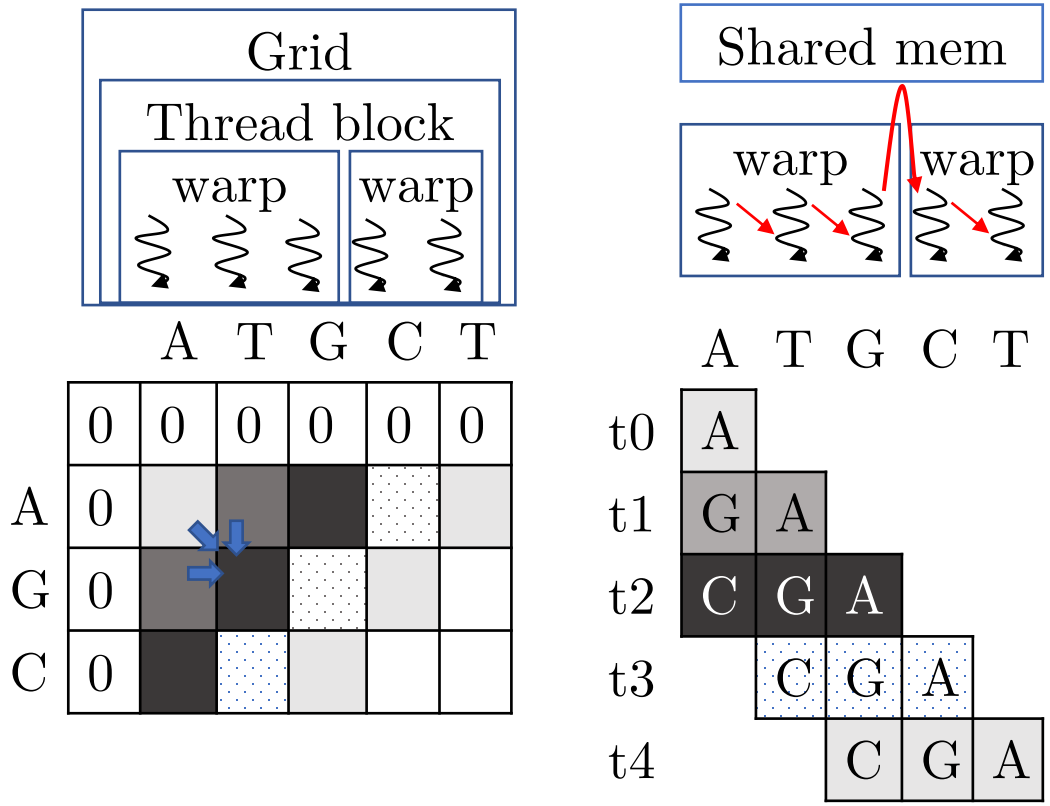
Automatic GPU Code Optimization Tool: GEVO[1]



[1] J.-Y. Liou, X. Wang, S. Forrest, and C.-J. Wu, “GEVO: Gpu code optimization using evolutionary computation,” ACM Trans. Archit. Code Optim (TACO), 2020

Workloads: ADEPT[1]
Gene sequence alignment

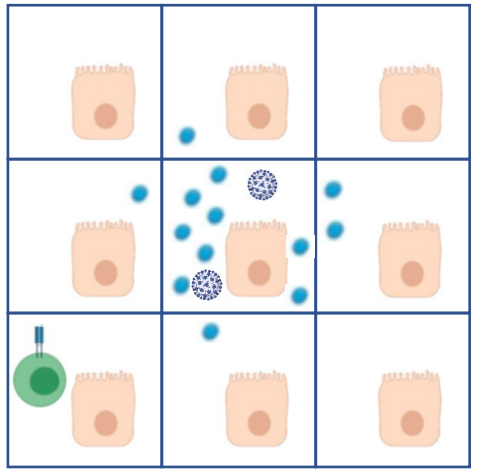
- Smith-Waterman algorithm
  - Dynamic programming for computing the score matrix
  - Exhaustively search all possible local alignment
- Per-cell score computation depends on 3 prior computations
  - Use both the private registers and the shared memory to exchange data



[1] M. G. Awan, J. Deslippe, A. Buluc, O. Selvitopi, S. Hofmeyr, L. Oliker, and K. Yelick, “Adept: a domain independent sequence alignment strategy for gpu architectures,” BMC bioinformatics, 2020

Workloads: SIMCoVGPU [1]
COVID-19 infection simulation on GPU

- Simulate virus infection and immune system response
  - Virus → epithelial cell → inflammatory signals → T-cell
  - Each thread simulates a slot on a 2D/3D grid

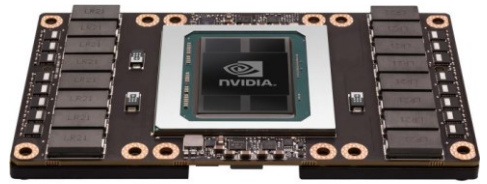


- Each thread communicates neighboring threads to
  - Accumulate the virus and inflammatory signals
  - Update status of lung epithelial cell
  - Move the t-cell

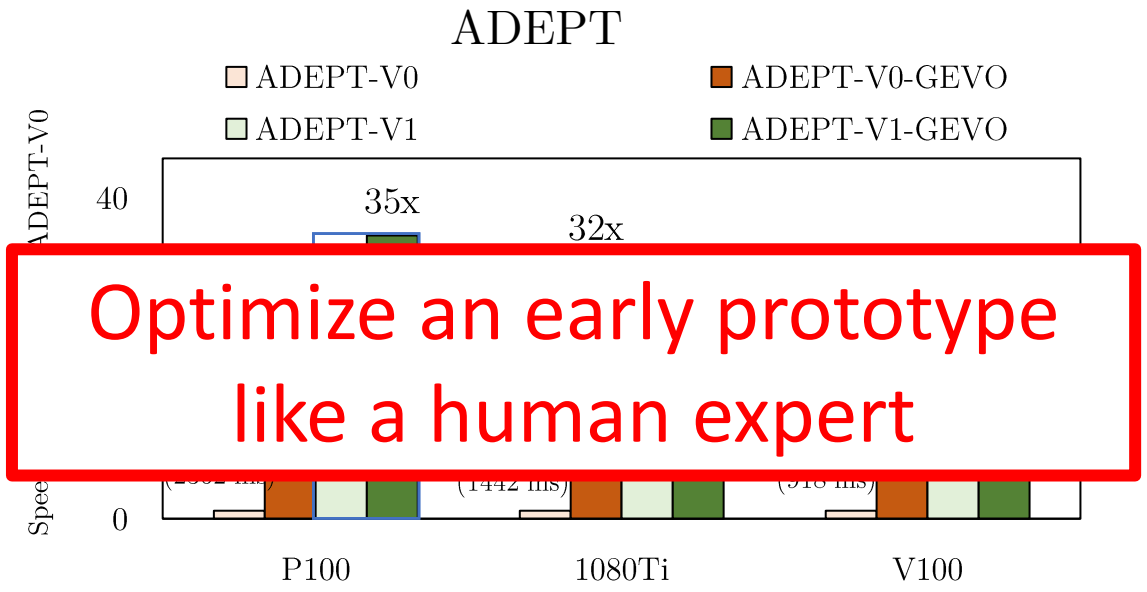
[1] M. E. Moses, S. Hofmeyr, J. L. Cannon, A. Andrews, R. Gridley, M. Hinga, K. Leyba, A. Pribisova, V. Surjadidjaja, H. Tasnim et al., "Spatially distributed infection increases viral load in a computational model of sars-cov-2 lung infection," PLoS computational biology, 2021

Experimental Setup

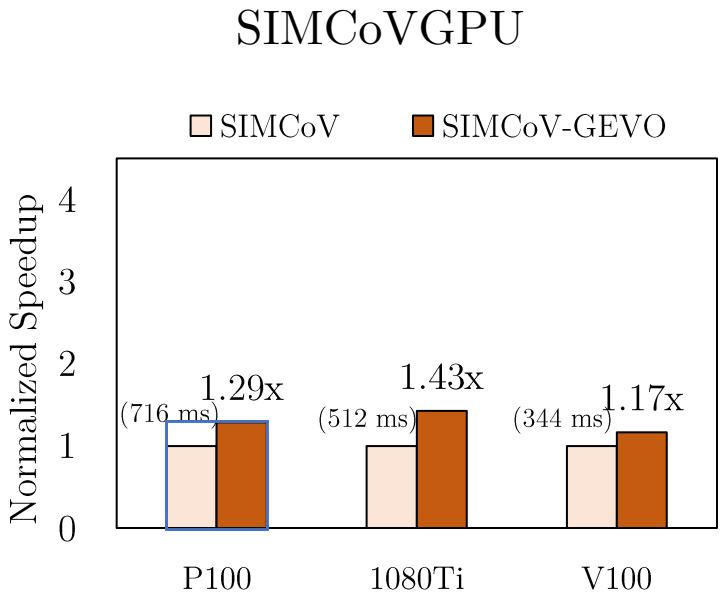
- Platform
  - GPU: Nvidia P100, 1080TI, V100
- Compilers – NVCC, Clang/LLVM
- Workloads
  - ADEPT-V0 (early prototype)
  - ADEPT-V1 (expert-optimized, 20-30x faster than V0)
  - SIMCoVGPU
- GEVO Parameters
  - Population size: 256
  - Cross rate: 80%
  - Mutation rate: 30%
  - Search time: 7 days (translates to 130 – 300 generations)

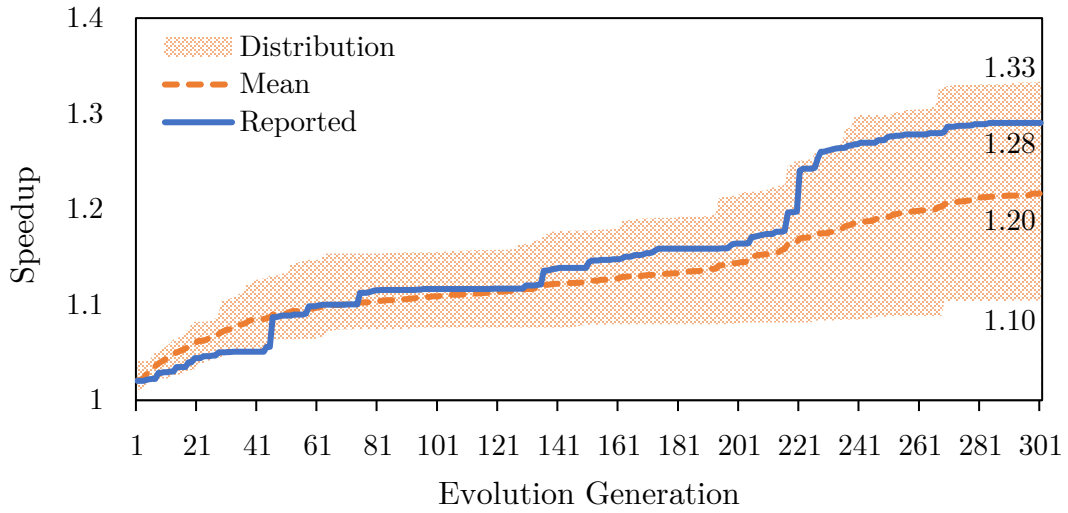


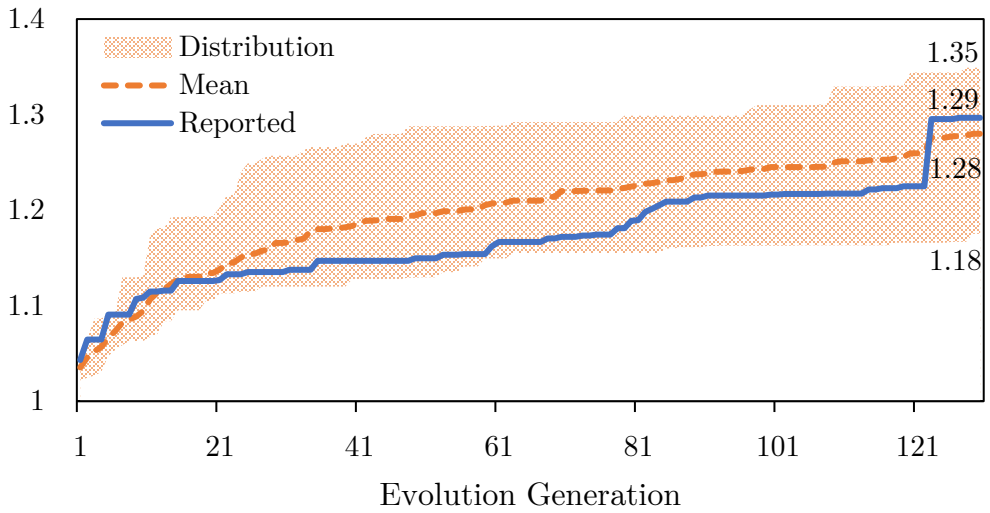
Experimental Results



Optimize an early prototype like a human expert

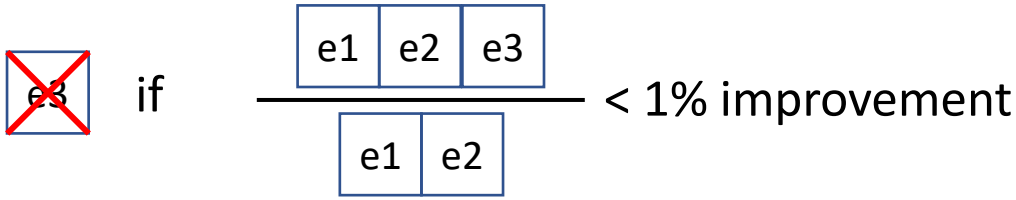




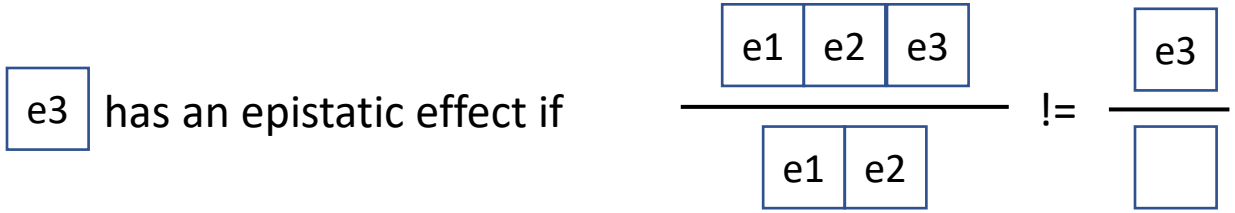


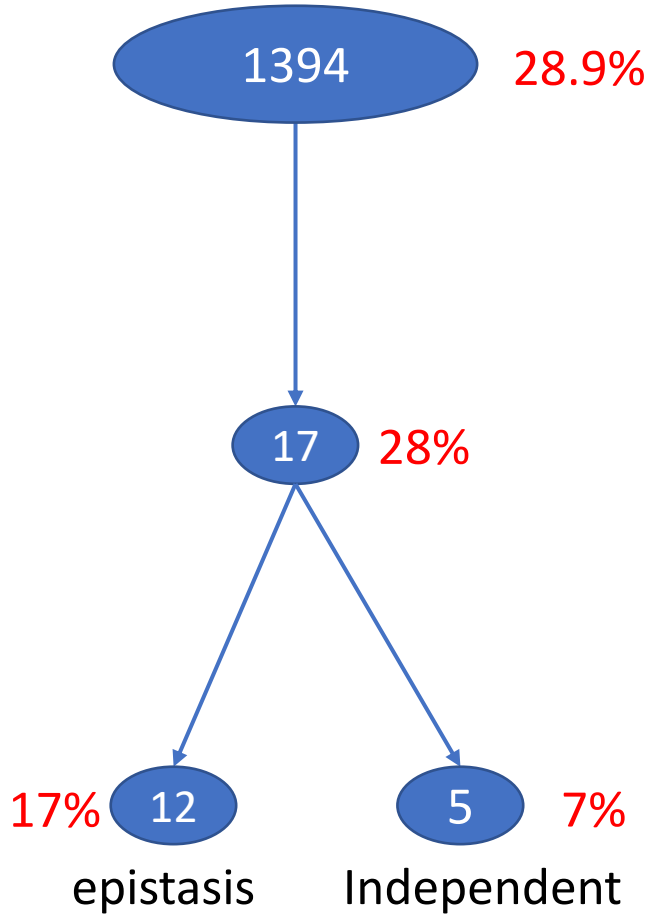
Edit analysis

- Why?
  - ADEPT-V1-GEVO has 1394 edits!
- Edit minimization (ablation)



- Detect edit interaction (epistatic)



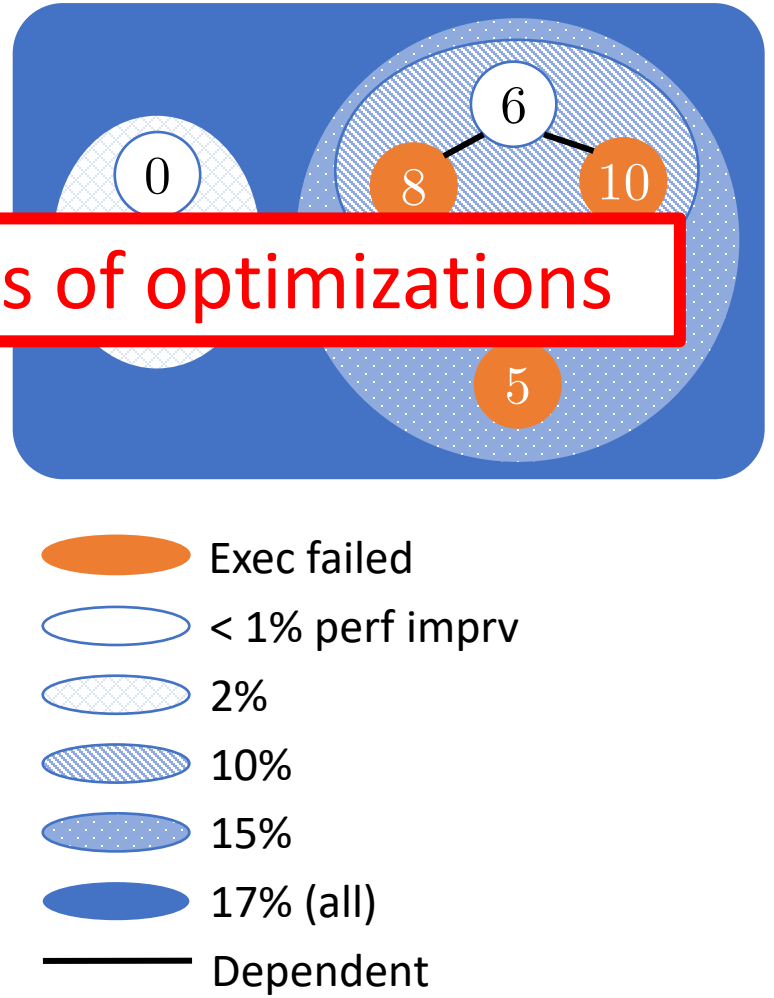


Edit analysis
Epistatic Edit Set Analysis

- Epistatic edits are rare in prior works!
  - Only on ADEPT-V1

Weak edits can be the stepping stones of optimizations

- Exhaustive search of all the combinations of the edits in the set



Optimization discovered for ADEPT

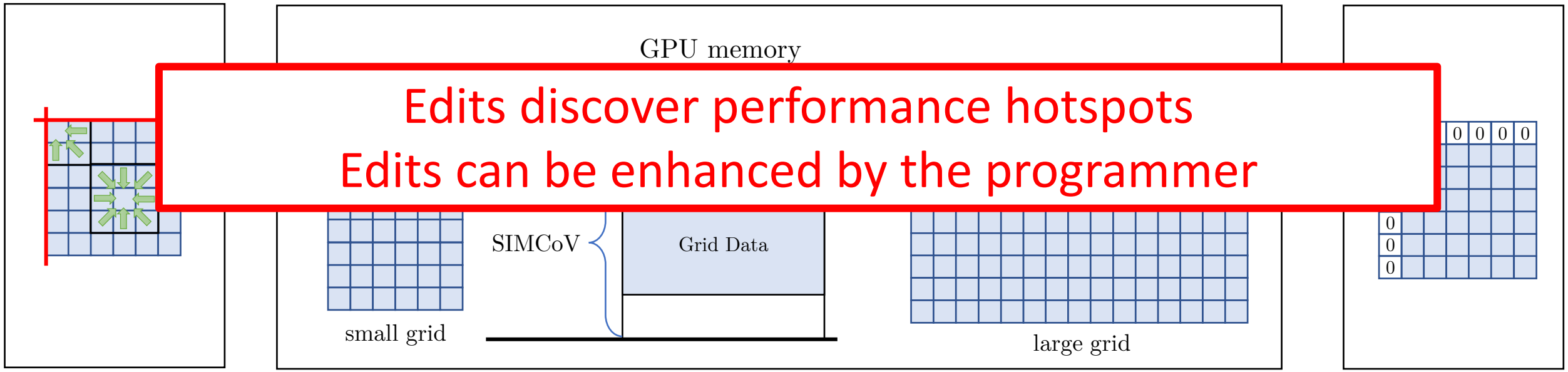
- Remove warp-level synchronization (4% ↑)
  - ADEPT-V1 on V100 GPU only
- Remove memory initialization (20X ↑)
  - ADEPT-V0
- Rearrange usage of sub-memory systems on GPU (15% ↑)
  - Private register + shared memory → shared memory only
  - ADEPT-V1

```
1   ...
2   // if (laneId == 31)
3   if (landId == 0) { // edit 5
4     sh_prev_E[warpId] = _prev_E;
5     sh_prev_prev_H[warpId] = _prev_prev_H;}
6
7   // if(diag >= maxSize)
8   if (tID < minSize) { // edit 6
9     local_prev_E[tID] = _prev_E;
10    local_prev_prev_H[tID] = _prev_prev_H; }
11
12  __syncthreads();
13
14  if (is_valid[tID] && tID < minSize) {
15    ...
16    // if(diag >= maxSize) {
17    if (is_valid[tID])  // edit 8
18      eVal = local_prev_E[tID-1] + extendGap;
19    else {
20      if (warpId != 0 && landId == 0)
21        eVal = sh_prev_E[warpId-1];
22      else // private register
23        eVal = __shfl_sync(...); }
24
25    // if(diag >= maxSize) {
26    if (is_valid[tID])  // edit 10
27      final_H = local_prev_prev_H[tID-1];
28    else {
29      if (warpId != 0 && landId == 0)
30        final_H = sh_prev_prev_H[warpId-1];
31      else // private register
32        final_H = __shfl_sync(...);
33    } ...
```

5 6 8 10

Optimization discovered for SIMCoVGPU

- boundary-check removal (20% ↑)
  - SegFault when grid size > 821x821
  - Manually padding the grid solves the SegFault issue (20% → 14%)



Limitation and Future work

- Limitation
  - No semantic correctness
    - Rely on test cases
  - Variable results
    - May need to have multiple GEVO runs
  - Analysis methods do not scale up
- Future work
  - Little edits add up

Conclusion

- ADEPT-V0 (3200%↑), ADEPT-V1 (28%↑) , SIMCoVGPU (43%↑)
  - Optimize memory synchronization and initialization
  - Optimize sub-memory usage
  - Opt-out boundary check
- Takeaways
  - GEVO is a great tool for both prototyping and hand-tuned programs
  - GEVO highlights performance hotspots and offers suggestions to developers
  - Weak edits can be the stepping stones for greater optimizations

Thanks for Yours Attention!

Understanding the Power of Evolutionary Computation for GPU Code Optimization

Acknowledgment