Biodesign for Computing

Stephanie Forrest

Biodesign Institute and School of Computing

Arizona State University

Santa Fe Institute

May, 2023

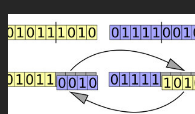
My Interdisciplinary Trajectory

Arizona State U., 2017 - present

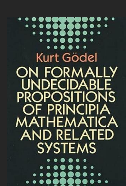
University of New Mexico, 1990 - 2017

Santa Fe Institute, 1990 - present

Center for Nonlinear Studies, LANL, 1988 - 1990



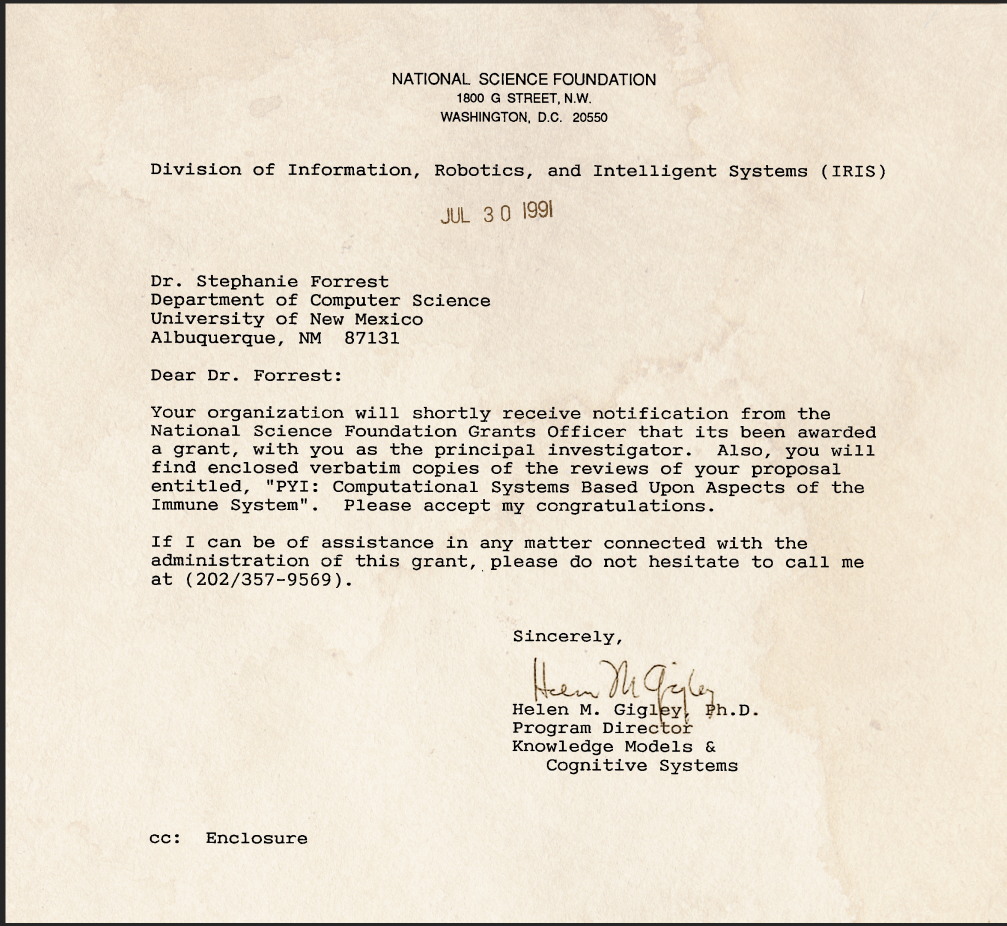
Univ. of Michigan, 1985



St. John’s College, 1977

The Role of NSF

- Lucky breaks
  - PYI Award letter “Computational aspects of the immune system” (1991)
  - Interdisciplinary research becomes socially acceptable
  - The web
- 30 years of NSF funding, rarely large grants
- Goal for talk: Make the case for strong connections between biology and computation, beyond neurons



NATIONAL SCIENCE FOUNDATION
1800 G STREET, N.W.
WASHINGTON, D.C. 20550

Division of Information, Robotics, and Intelligent Systems (IRIS)

JUL 30 1991

Dr. Stephanie Forrest
Department of Computer Science
University of New Mexico
Albuquerque, NM 87131

Dear Dr. Forrest:

Your organization will shortly receive notification from the National Science Foundation Grants Officer that its been awarded a grant, with you as the principal investigator. Also, you will find enclosed verbatim copies of the reviews of your proposal entitled, "PYI: Computational Systems Based Upon Aspects of the Immune System". Please accept my congratulations.

If I can be of assistance in any matter connected with the administration of this grant, please do not hesitate to call me at (202/357-9569).

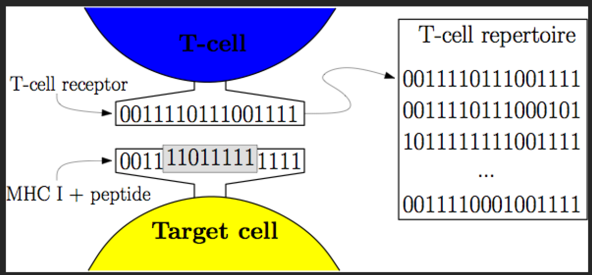
Sincerely,

Helen M. Gigley, Ph.D.
Program Director
Knowledge Models &
Cognitive Systems

cc: Enclosure

The Biology of Computation

- Defending complex systems from malicious behavior
  - Vaccine design, cancer, other evolving pathogens
  - Ch 1. Computer immune systems
- Engineering and evolution of software
  - Ch 2. Micro-level: Evolutionary computation methods
  - Ch 3. Macro-level: Inadvertent evolution



Computer Immune System

Evolving Software

networkworld.com

Information Processing in the Immune System

- Learned distinction between self and other
- Primary response to new foreign antigen
- Evolved biases towards common pathogens

- Secondary response
- Cross-reactive memory

- $10^{11} – 10^{16}$ different foreign patterns from ~25,000 genes

An activated macrophage phagocytosing bacteria upon contact
Photo: courtesy of Dennis Kunkel

Edward Jenner's first smallpox vaccine performed on James Phipps in 1796

http://www.history.com/news/vaccines diseases forgotten

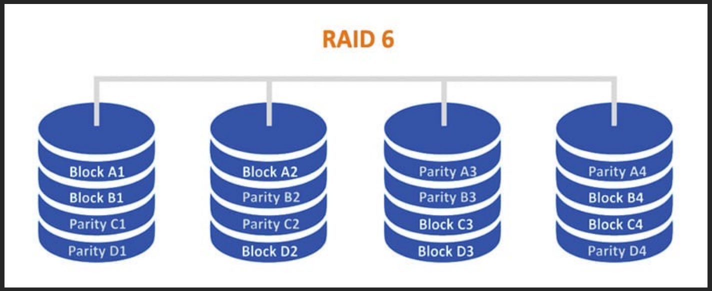
Cybersecurity Recapitulates Biology

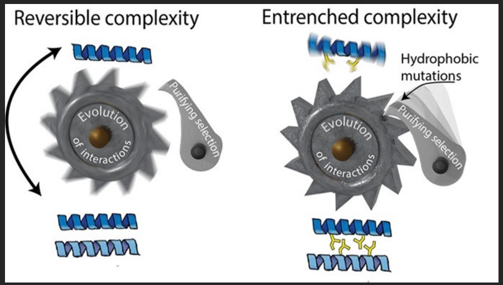
- Anomaly intrusion detection, signature detection

- Address space randomization
- Natural diversity for N-variant systems

- Two-factor authentication

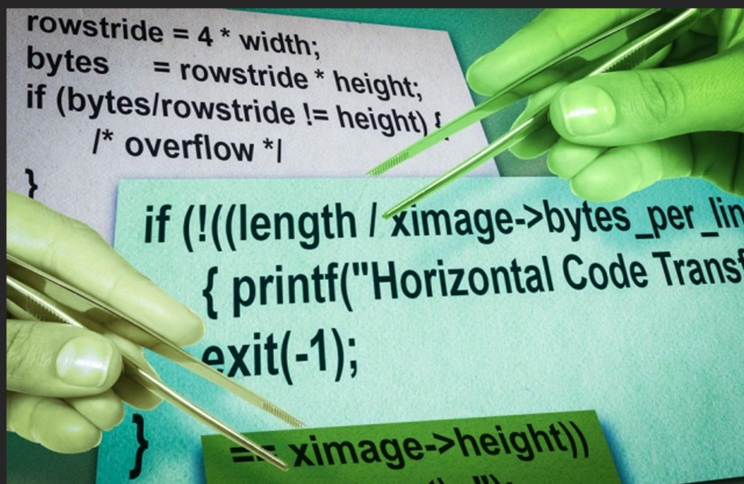
- Ratchets, constructive neutral evolution
- Limits to defense-in-depth?





Hochberg et al. *Nature*, 2020

Evolution in Software

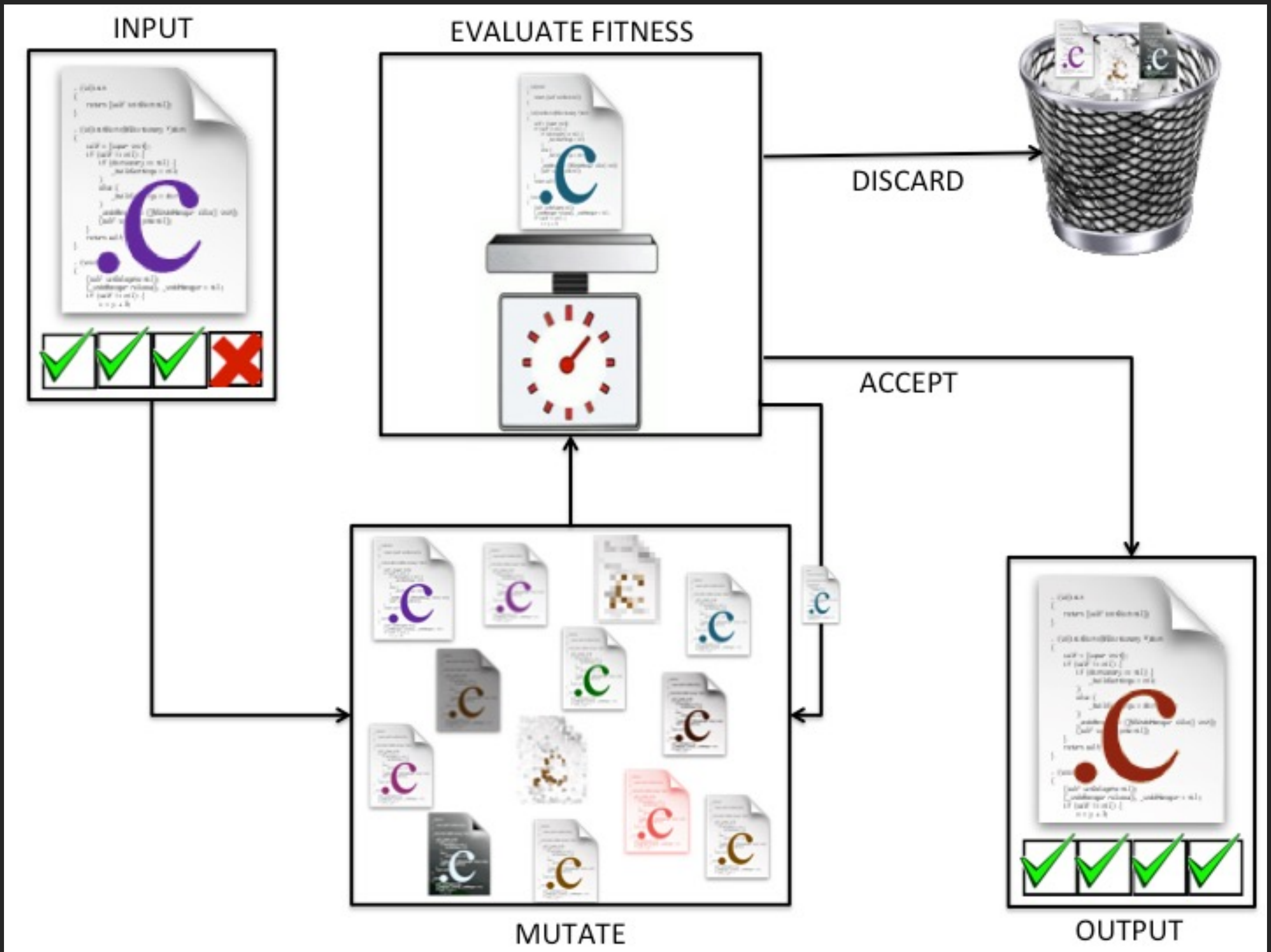


Jose Luis Olivares

networkworld.com

- Macro-level: Inadvertent evolution
- Micro-level: Evolutionary computation methods

Micro-evolution of Software



GenProg

So: C. Le Goues

ICSE '09: W. Weimer, T. Nguyen, C. Le Goues, and S. Forrest. Automatically finding patches using genetic programming. 2019: Award: Most influential paper published at the 2009 ICSE.

The Secret Sauce

- Start with a working program
- Mutations mimic human operations
  - Delete, Copy, Move/Replace
  - Don't invent new code, statement-level operations
- Restrict mutations to statements executed by failing test cases
- Most bugs are small

```
void zunebug_repair(int days) {
  int year = 1980;
  while (days > 365) {
    if (isLeapYear(year)){
      if (days > 366) {
        // days -= 366; // repair deletes
        year += 1;
      }
      else {
      }
      days -= 366;       // repair inserts
    } else {
      days -= 365;
      year += 1;
    }
  }
  printf("current year is %d\n", year);
}
```

How well does it work in practice?

- Large systematic empirical studies with many tools
  - Defects4J: Java programs (all tools---36% correct)
  - ManyBugs: Large opensource C programs (72% plausible)
- Industry transitions
- The Machine Learning tsunami
- Caveats
  - Buggy test cases
  - ‘’Overfitting” (patch vs. repair)
  - What is a correct repair ?
  - Assumptions made by tool, e.g., fault localization to a single line
  - Reproducibility
  - Difficult to know what the ML models have been trained on

Dorsa Amir @DorsaAmir · 2m

When your code is a mess but it somehow still works.

Biological Properties of Software

Eric Schulte, Joe Renzullo, Jhe-Yu Liou



- Mutational robustness
  - *Mutation testing considered helpful*
- Neutral landscapes
- Fitness distributions
  - *Where should we look for repairs?*
- Epistasis (interactions among genes)

Neutral Mutations

- Many biological mutations leave fitness unchanged
  - Buffering, genetic potential
- A neutral mutation passes the original test suite
  - It may or may not pass held-out failing test cases
  - Plentiful: ~30% of GenProg mutations are neutral!

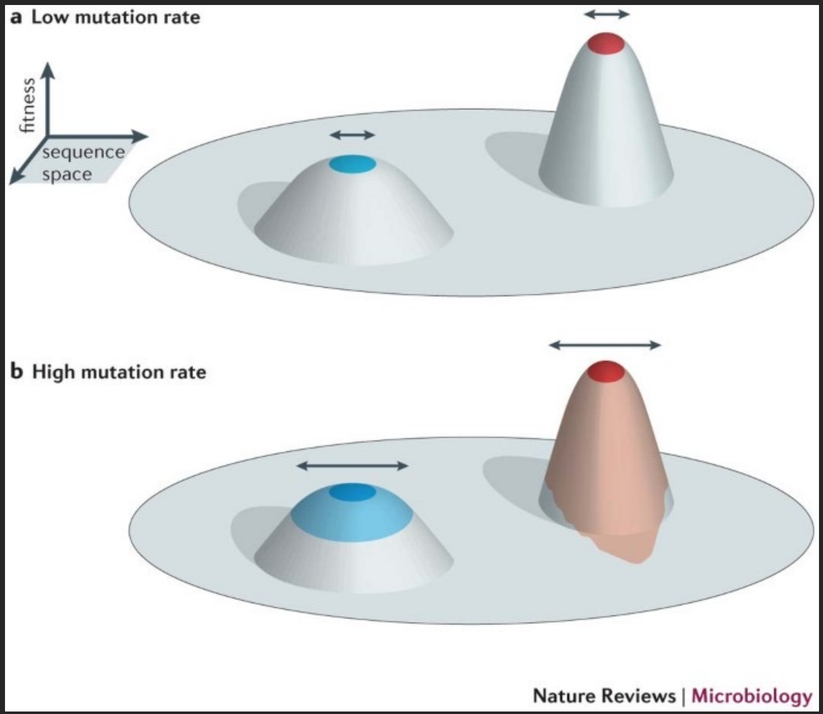
```
if (right > left) {
  // code elided
  quick(left, r)
  quick(l, right)
}
```

→

```
quick(l, right)
quick(left, r)
```

Schulte, et al. Software mutational robustness. *Genet. Program. Evolvable Mach.* **15**, 281–312 (2014).
Harrand, et al. A journey among Java neutral program variants. *Genet. Program. Evolvable Mach.* **20**, 531–580 (2019).

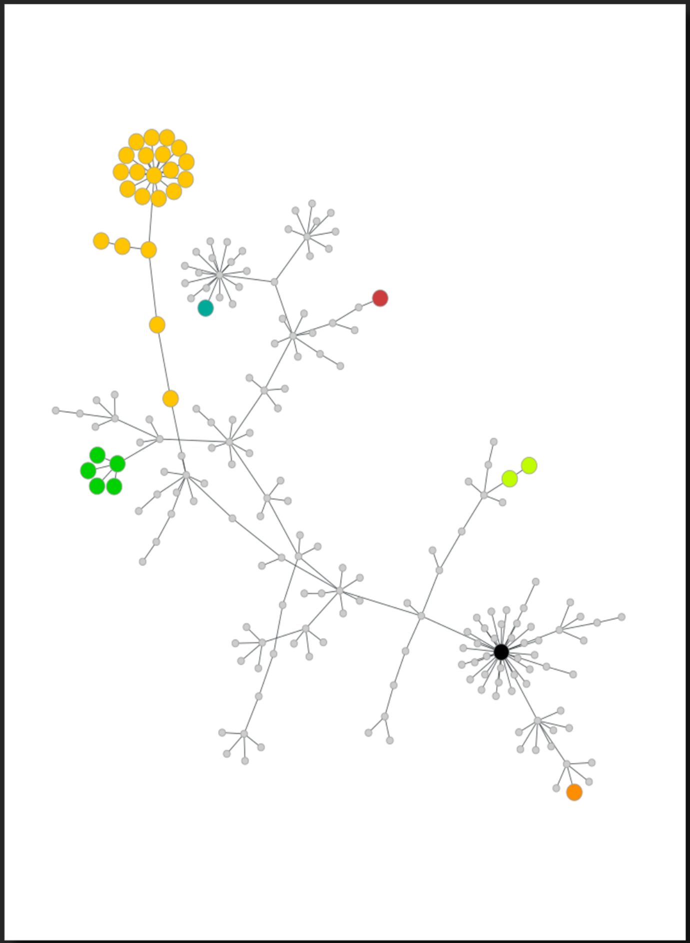
Neutral Mutations Enable Search



- Engineered diversity
- Reducing energy consumption
- For bug repairs
- For reducing GPU run-times

So: https://www.cambridge.org/core/journals/philosophy-of-science/article/abs/neutral-spaces-and-topological-explanations-in-evolutionary-biology-lessons-from-some-landscapes-and-mappings/B6F5D33A8ECB582D324CF7C8E1FC4127

Neutral Landscapes



Buffer overflow repair (look)

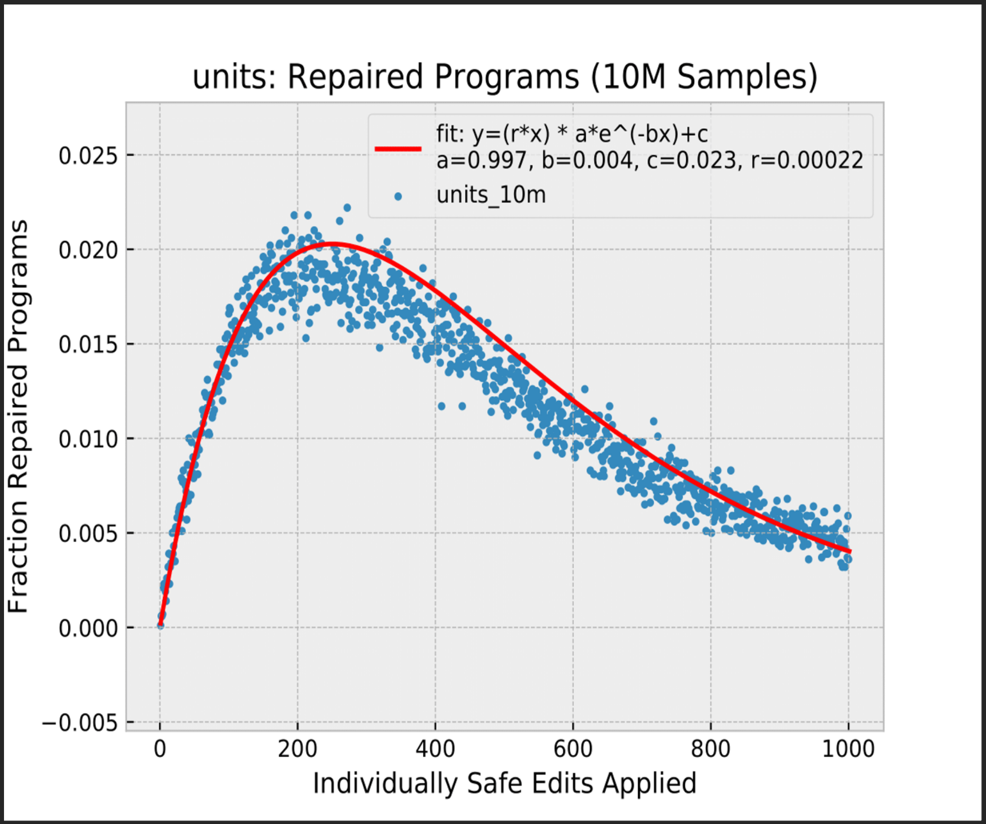
ICSE GI Workshop, 2018

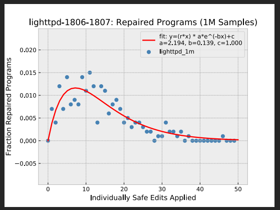
- Neutral mutations sometimes repair latent bugs
- Many semantically distinct repairs
  - Color indicates unique repairs
- Network connects diverse repairs by neutral intermediate mutations
- Insight: All repairs are neutral wrt original test suite

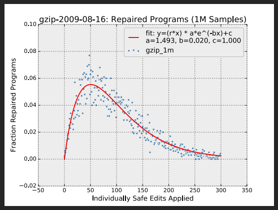
Fitness Distributions:

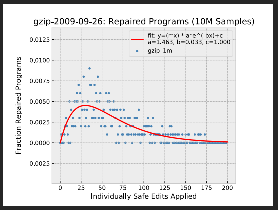
Where are the repairs in neutral space?

1. Generate large pool of neutral edits
2. Generate random subsets of pool
3. Apply each subset to original program
4. Measure repair frequency







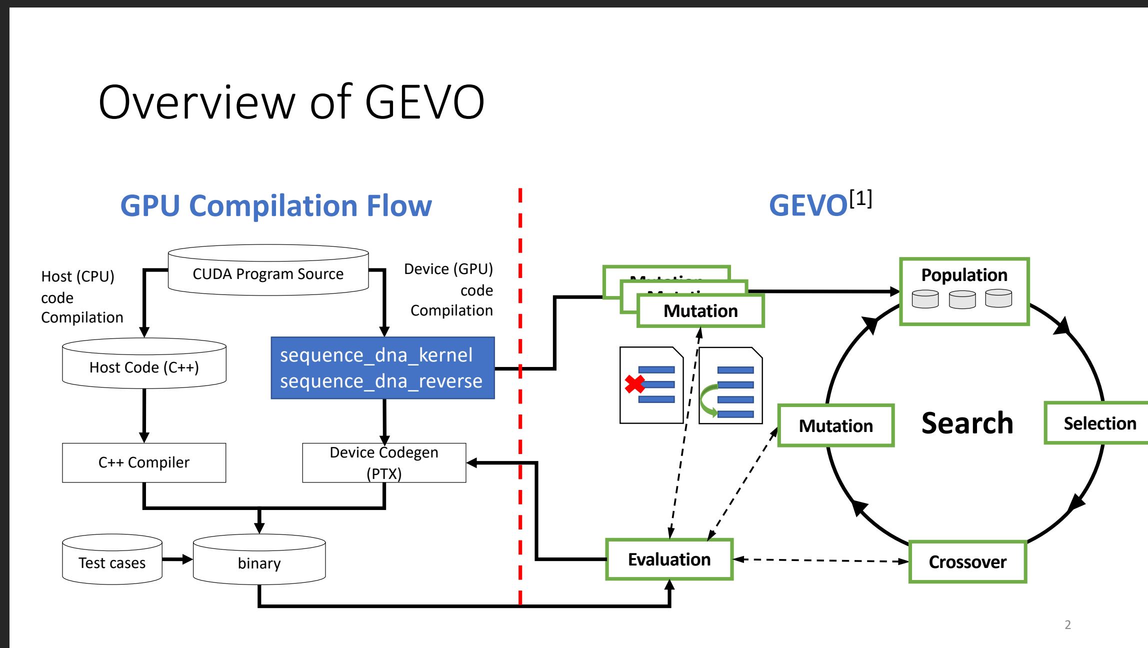


100 times more likely to find a patch at distance 200 than at distance 1

ACM TELO, in press

Evolving Faster GPU Code

J. Liou, C. Wu and S. Forrest (TACO, 2020)

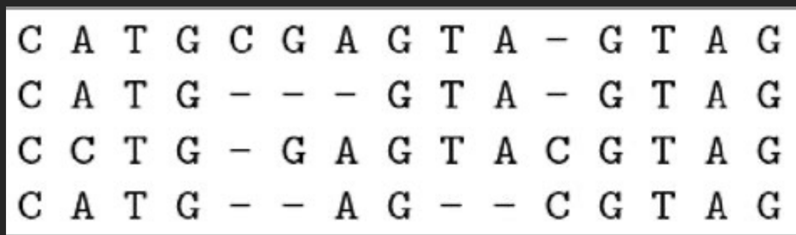


- GPUs important for ML and HPC, but challenging to optimize
- More complex mutation operators
- 49% average speedup on Rodinia benchmarks (NVIDIA Tesla P100)

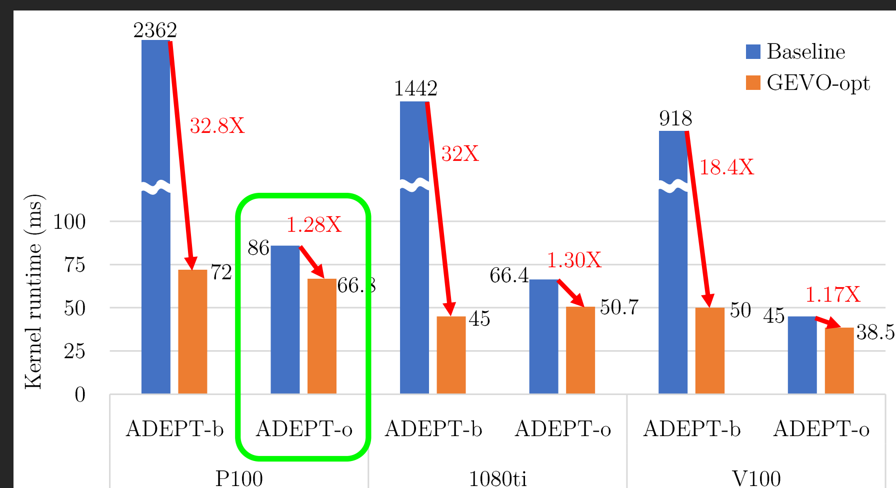
Optimizations: Application logic, architecture-specific, dataset specific

Optimizing Multiple Sequence Alignment Codes

(J. Liou, M. Gul Awan, C. Wu, S. Hofmeyr, and S. Forrest, ISWC 2022)

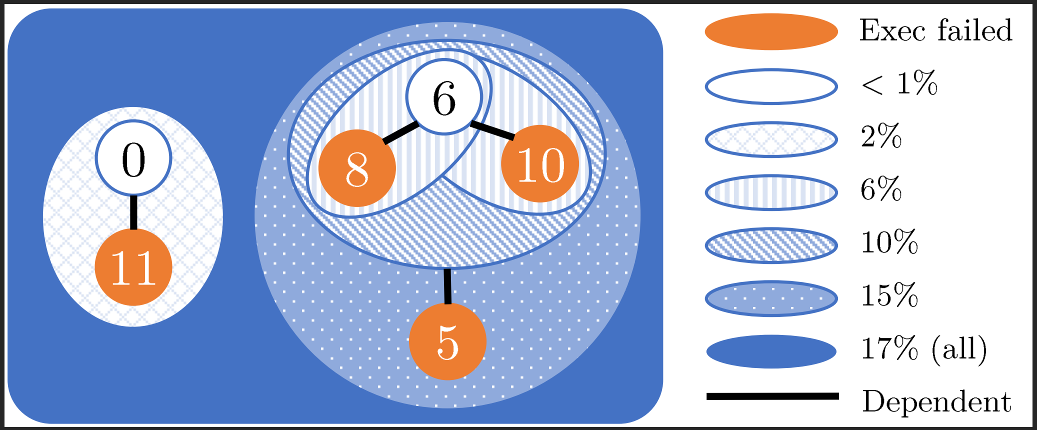


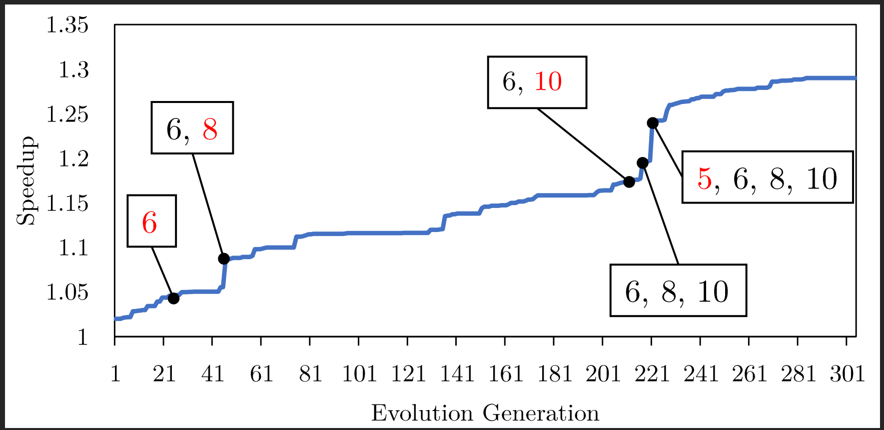
- Smith-Waterman algorithm (ADEPT)
  - State-of-the art implementation on GPU
  - Hand-optimized for GPU by human expert
- GEVO run
  - 256 pop size; 300 gens; 7 days
  - 64 mutations, 17 useful
  - 5 independent mutations (7%)
  - 12 interdependent (18% improvement)



GEVO finds 28.5% run-time improvement over expert human-optimized version

GEVO optimizations are epistatic



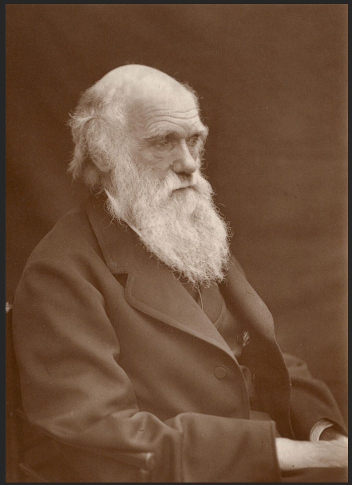


ADEPT-o on P100 GPU.

- Rearrange usage of sub-memory systems on GPU (15%)
  - Use shared memory instead of private registers
- Remove redundant synchronizations (~4%)
  - violates CUDA Programming guide
- Remove unnecessary memory initializations (30X on adept-b)

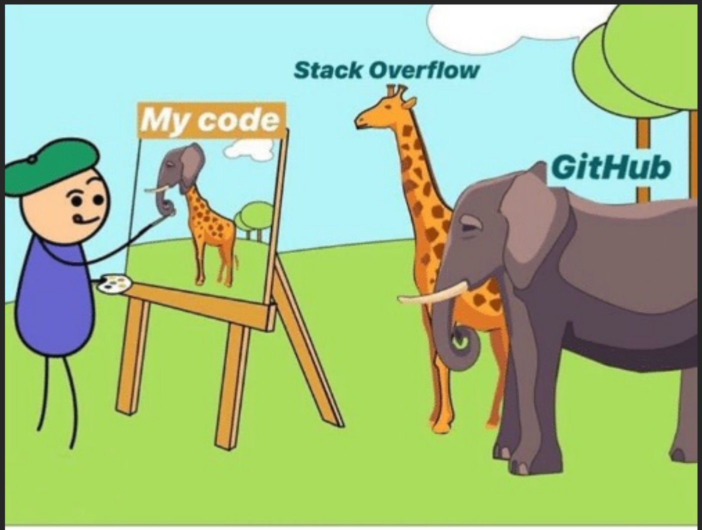
Epistatic optimizations can be hard for humans to find

The Bigger Picture



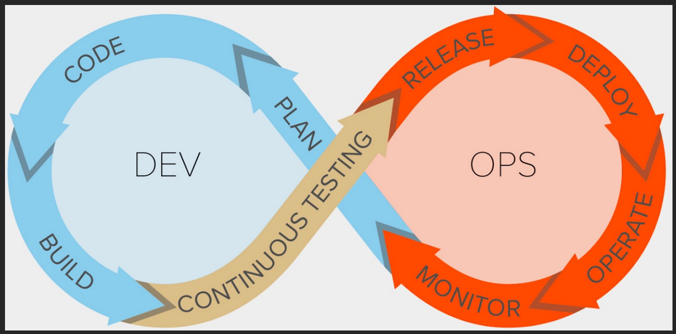
- Key ingredients of Darwinian evolution
  - Variation: Mutation and recombination
  - Natural selection
  - Inheritance
- Software
  - Selection and inheritance: Successful genes are copied: libraries, packages, code snippets, etc.
  - Variation: Programmers make small changes and recombine successful genes

Thesis: Software today is the result of many generations of inadvertent evolution



Stackoverflow + github

Macro-evolution in Software



Continuous Integration


Uber Two-factor authentication attack

Arms races

The Tinkerer and the Craftsman

Evolution

- Unplanned and openended
- Survival, relative fitness
- Ongoing process
- Incremental
- Driven by random mutation

Engineering

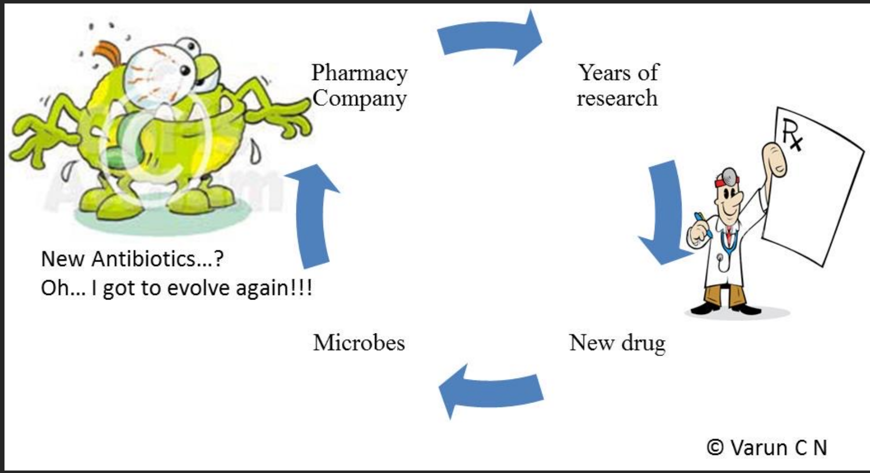
- Planned, with specifications
- Purposeful, goal-driven
- Clean slate design
- Large jumps
- Conducted by agents with foresight and intent

'Nature is a tinkerer, not an inventor'

F. Jacob

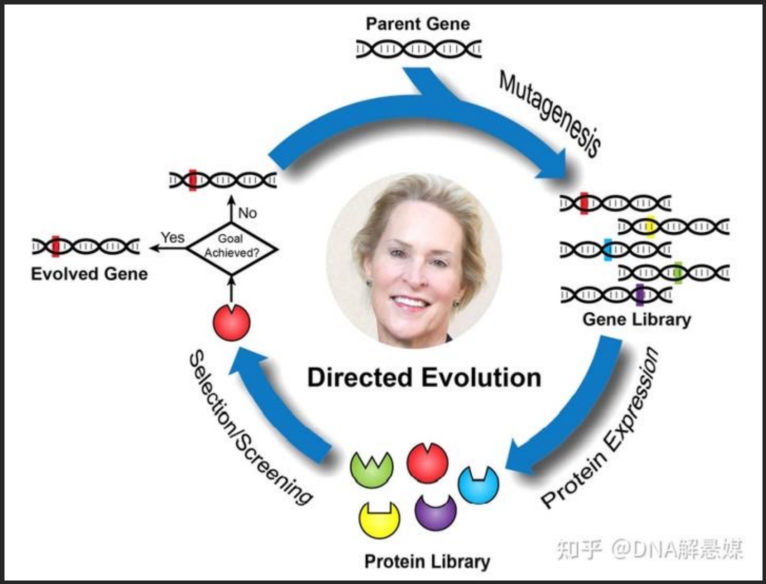
*Evolution **and** Engineering*

- **Antibiotic resistance**
- Directed evolution
- Synthetic biology
- Attack fuzzing in cybersecurity
- Large jumps in evolution
- Randomized algorithms
- Software



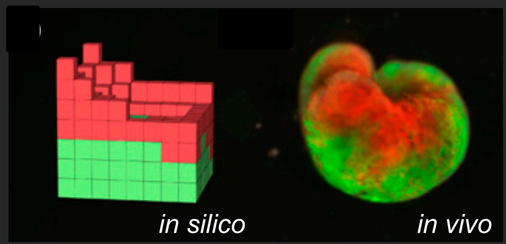
*Evolution **and** Engineering*

- Antibiotic resistance
- **Directed evolution**
- Synthetic biology
- Attack fuzzing in cybersecurity
- Large jumps in evolution
- Randomized algorithms
- Software



*Evolution **and** Engineering*

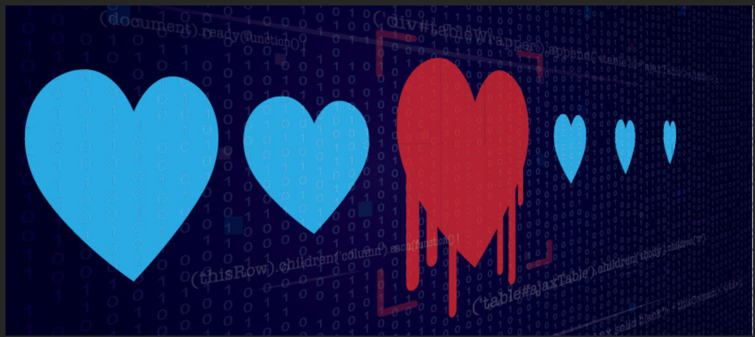
- Antibiotic resistance
- Directed evolution
- Synthetic biology, **xenobots**
- Attack fuzzing in cybersecurity
- Large jumps in evolution
- Randomized algorithms
- Software



PNAS, 2020

*Evolution **and** Engineering*

- Antibiotic resistance
- Directed evolution
- Synthetic biology
- Attack fuzzing in cybersecurity
- Large jumps in evolution
- Randomized algorithms
- **Software**



What are the best practices for engineering systems in the context of evolution?

- Claim: Software is an excellent starting point
- Co-evolution
  - Interactions with humans
  - Interactions among software components
  - Interactions with biology
- Highly optimized tolerance
  - Understanding tradeoffs between performance and robustness (Carlson and Doyle)
- Rethinking defense-in-depth and technological ratchets

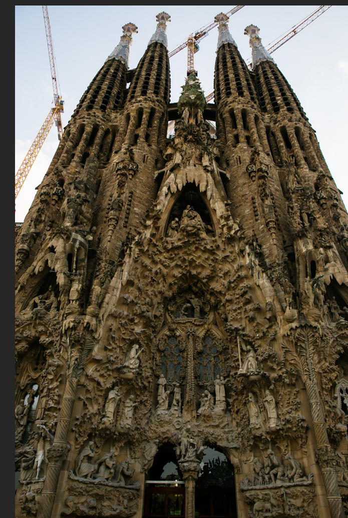
Summary

- The perspective of biology is important because it provides insight and guidance
  – Engineering (bio-inspired computing)
  – Science (biological properties of computation)

"As engineers, we would be foolish to ignore the lessons of a billion years of evolution"

Carver Mead

THANK YOU



steph@asu.edu

https://profsforrest.github.io

ASU Arizona State University NSF DARPA

References

- W. Weimer, T. Nguyen, C. Le Goues, and S. Forrest. Automatically finding patches using genetic programming. In *ICSE ’09: Proc. of the 2009 IEEE 31st Intl. Conf. on Software Engineering*, pages 364–374, Washington, DC, USA, 2009.
- C. Le Goues, M. Dewey-Vogt, S. Forrest, and W. Weimer. A systematic study of automated program repair: Fixing 55 out of 105 bugs for $8.00 each. In *ICSE ’12: Proc. of the IEEE 34th Intl. Conf. on Software Engineering*, 2012.
- E. Schulte, Z. P. Fry, E. Fast, W. Weimer, and S. Forrest. Software mutational robustness. *Genetic Programming and Evolvable Machines*, 15(3):281–312, 2014. DOI 10.1007/s10710-013-9195-.
- E. Schulte, J. Dorn, S. Forrest, and W. Weimer. Post-compiler software optimization for reducing energy. In *Nineteenth Intl. Conf. on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, 2014.
- J. Liou, X. Wang, S. Forrest, and C. Wu. Post-compiler performance tuning for general-purpose GPU kernels. *ACM Trans. on Architecture and Code Optimization*, 17(4), 2020.
- J. Liou, M. Awan, S. Hofmeyr, C. Wu, and S. Forrest. Understanding the power of evolutionary computation for GPU code optimization. In *2022 IEEE International Symposium on Workload Characterization*, in press.
- J. Renzullo, W. Weimer, and S. Forrest. Multiplicative weights algorithms for parallel automated software repair. In *35th IEEE International Parallel and Distributed Processing Symposium*, 2021.