

Object-Level Recombination of Commodity Applications



Carleton
UNIVERSITY

Canada's Capital University

Blair Foster

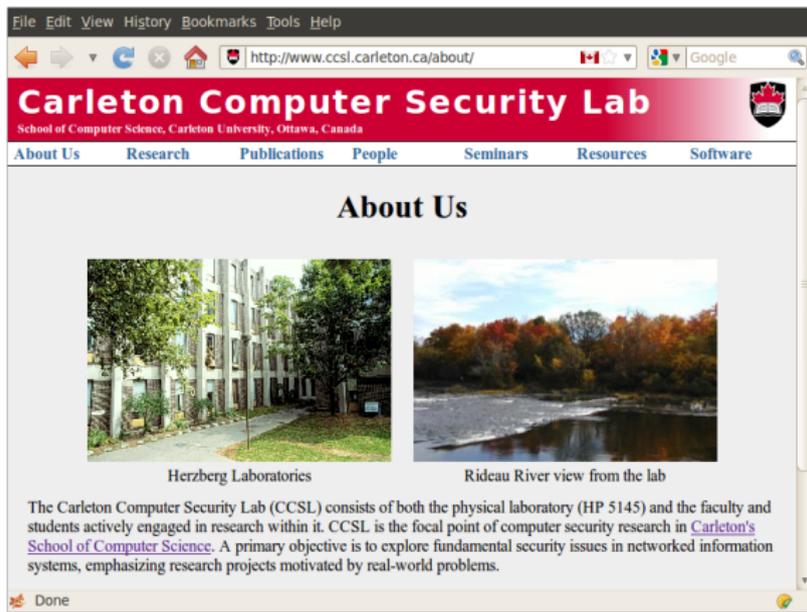
Anil Somayaji

Carleton Computer Security Lab
School of Computer Science
Carleton University

July 10, 2010

The Dream

Wouldn't it be nice to evolve **real programs** automatically?



The screenshot shows a web browser window displaying the Carleton Computer Security Lab website. The browser's address bar shows the URL <http://www.ccsl.carleton.ca/about/>. The website has a red header with the text "Carleton Computer Security Lab" and "School of Computer Science, Carleton University, Ottawa, Canada". Below the header is a navigation menu with links for "About Us", "Research", "Publications", "People", "Seminars", "Resources", and "Software". The main content area is titled "About Us" and features two images: "Herzberg Laboratories" and "Rideau River view from the lab". Below the images is a paragraph of text describing the lab's mission and objectives.

File Edit View History Bookmarks Tools Help

http://www.ccsl.carleton.ca/about/ Google

Carleton Computer Security Lab

School of Computer Science, Carleton University, Ottawa, Canada

[About Us](#) [Research](#) [Publications](#) [People](#) [Seminars](#) [Resources](#) [Software](#)

About Us



Herzberg Laboratories



Rideau River view from the lab

The Carleton Computer Security Lab (CCSL) consists of both the physical laboratory (HP 5145) and the faculty and students actively engaged in research within it. CCSL is the focal point of computer security research in [Carleton's School of Computer Science](#). A primary objective is to explore fundamental security issues in networked information systems, emphasizing research projects motivated by real-world problems.

Done

The Reality of “Real Programs”

- **big**
 - millions of lines of code!
- brittle representations
 - C statements versus S-expressions
- brittle semantics
 - complex algorithms and data structures
- fitness functions?!

Hope

Commodity applications—the programs we use every day—can be recombined *with members of their own species*.

Huh?

Species

Horse \times Camel = $\emptyset \Rightarrow$

Mozilla Firefox \times Google Chrome = \emptyset

Firefox 3.6.6 \times Firefox 4.0b1 = ???

Mapping Programs to Individuals

executable → individual

.o file → chromosome

Linking → development

Our Contribution

- Recombine program variants represented as object files.
- Parents must share a common recent ancestor.
- Use linker tricks, GA search to find and create viable children.
- Use a weak fitness function that only tests for basic viability; humans choose interesting variants.

ObjRecombGA

ObjRecombGA

[-- Target Settings --]

Primary Dir:

Secondary Dir:

Target Binary:

Primary Build String:

Secondary Build String:

Common Build String:

Object Restrict List:

Test Script:

Avoid Convergence Force Test

[-- General Settings --]

Scripts Dir:

Recombination Dir:

Temp Dir:

Population Size: # Generations:

Selection Algorithm: **TournamentSelection** ▼

X-over Algorithm: **SinglePointXover** ▼

Tournament Prob.: Elite #: Diversity Max Prob:

Initial Population List:

<[Start]>

Interesting GA characteristics

- Bitstring representation: each bit selects an object file
- Small populations: 12-50 individuals
- Few generations: 20 or less
- Fast convergence to maximum fitness
 - Coarse fitness function that gives points for compiling, not crashing, and a few basic functionality tests.

Remember, this GA is designed to explore, not optimize!

The Linking Solution

- Allow linking against all object files: bitstring indicates linking *priority*.
- Do our own symbol resolution to choose appropriate variant:
 - Catalog symbols and precompute potential dependency chains.
 - Mangle all symbols to make them unique, remember mappings.
 - Rewrite symbol references to refer to appropriate version of code or data.
- And, search for viable offspring using a GA!

Dillo



Dillo: Dillo Web Browser :: Home Page

http://www.dillo.org

Dillo
 8 years!

- Home
- Current Plans
- Changelog
- Screenshots
- Download
- Media List
- FAQ
- Compatibility
- Donate
- Firefox 2005
- Linux 2005

Bug Track Engine

- Bug Track Intro
- New Entries
- Bug Tracker
- Volunteering

Developers

- New Developer
- Naming & Coding
- Dev. Spec
- CSS Spec

Source repository

- Authors
- Security contact

Users

- Help
- Icons
- Run meter

The Dillo Web Browser

Welcome to the Dillo Project!

What is Dillo?

- Dillo is a [multi-platform](#) graphical web browser known for its speed and small size.
- Dillo is written in C and C++.
- Dillo is based on [ELITE](#), the Fast Light Toolkit (statically-linked by default).
- Dillo is free software made available under the terms of the GNU General Public License (GPLv3).
- Dillo strives to be friendly both to users and developers.
- Dillo helps web authors to comply with web standards by using the [bug meter](#).

Project Objectives:

- The democratization of internet information access.
- Personal security and privacy.
- High software efficiency.

So, you have been using dillo for a while, but have you read about the [project objectives](#)? Please give it some time, you'll find some interesting issues there!



Dillo: Dillo Web Browser :: Home Page

The Dillo Web Browser

Dillo
 8 years!

- Home
- Current Plans
- Changelog
- Screenshots
- Download
- Media List
- FAQ
- Compatibility
- Donate
- Firefox 2005
- Linux 2005

Bug Track Engine

- Bug Track Intro
- New Entries
- Bug Tracker
- Volunteering

Developers

- New Developer
- Naming & Coding
- Dev. Spec
- CSS Spec

Source repository

- Authors
- Security contact

Users

- Help
- Icons
- Run meter

Related

- Site in the Press
- Interview

Welcome to the Dillo Project!

What is Dillo?

Project Objectives:

So, you have been using dillo for a while, but have you read about the [project objectives](#)? Please give it some time, you'll find some interesting issues there!

Current version

Dillo follows an evolving software model where each new version should be better than the previous one; there's no place for unstable releases, so just keep with the latest one: **dillo-2.0**.

Click [here](#) for the downloads page.

Code contributions

Quake: HUD, no models



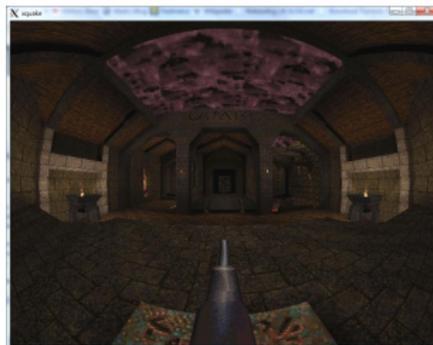
Quake: Fisheye



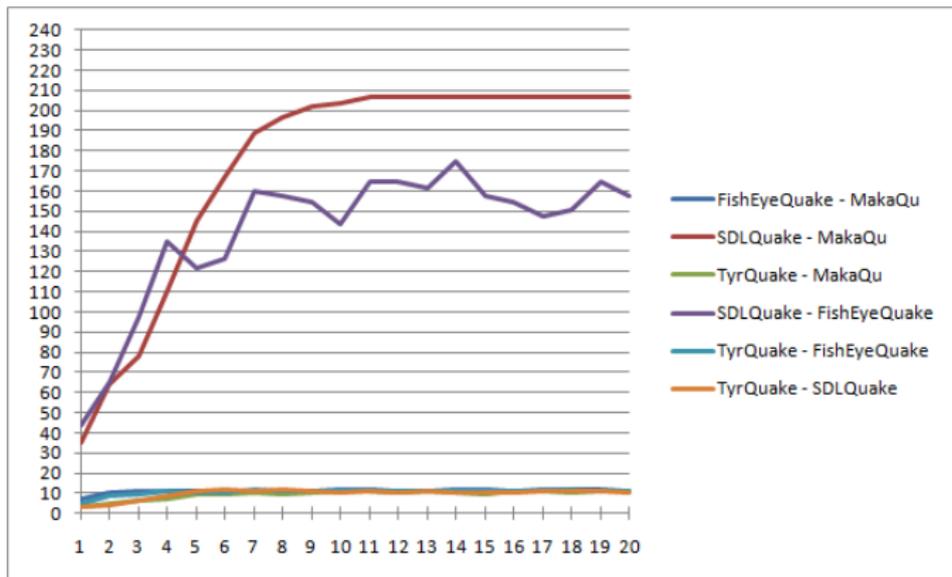
Quake: Recombined Child



Quake: Parents and Children



Fitness Convergence



Note that populations exhibit high diversity.

Limitation of current implementation

- Two ancestors only.
- Child programs cannot be used as ancestors
 - Children are effectively diploid, parents are haploid.
- Linking sometimes fails when it could have succeed.
- Primitive fitness functions.

Potential Applications

- Software testing/debugging
- Functional diversity for security
- User-directed software evolution

Conclusion

- Commodity software can be recombined using object files as the units of recombination.
- Linking problems are significant but can be overcome with linker tricks.
- Functional dependencies are addressed through GA search.
- Only works with closely related programs.
- Defines a “species” relationship amongst programs.
- Opens the door to automated evolution of the software we use everyday.

Acknowledgements

- Funded by Canada's NSERC Discovery Program
- Blair Foster (Master's student!)
- "Road to Software Evolvability," Santa Fe Institute 2005
- Stephanie Forrest (graduate advisor), David Ackley

Contact me at soma@csl.carleton.ca

Questions?