Rethinking Genetic Improvement Programming

David R. White
University of Glasgow, Scotland

Sunday 12th July 2015
GI 2015, GECCO
Genetic Programming has gone backwards.

http://www.davidrwhite.co.uk/2014/11/27/genetic-programming-has-gone-backwards/
Starting at the End
Evolutionary Improvement of Programs

David R White, Andrea Arcuri, John A Clark.

Abstract—Most applications of Genetic Programming (GP) involve the creation of an entirely new function, program or expression to solve a specific problem. In this paper we propose a new approach that applies GP to improve existing software by optimising its non-functional properties such as execution time, memory usage or power consumption. In general, satisfying non-functional requirements
Readymades (Marcel Duchamp)
How does existing code help us?

Existing code provides:

1. An Oracle
2. A starting point
3. Raw material
Existing Software as an Oracle
The Oracle of Delphi
An Oracle

- Full (non-functional optimisation) or partial (bug-fixing)

We can effectively treat the Oracle as a specification for new versions:
- N-Version programming\(^1\)
- Reverse Engineering\(^2\)

---

\(^1\)R. Feldt. Generating Multiple Diverse Software Versions with Genetic Programming. In Euromicro Conference, 1998

Why not consider software translation in a very general sense?

- Porting to new languages.
- Target to new platforms and technologies.
  - Automated parallelisation?\(^3\)
  - CUDA and GPGPU\(^4\)
- Compressing and simplifying programs.
- A solution to the problem of *legacy software*?

---


\(^4\) W B Langdon and M Harman. Evolving a CUDA kernel from an nVidia template. In *CEC*, 2010
Existing Software as a Starting Point
The number of possible trees of depth $d$ is given by:

$$c(d) = \begin{cases} 
  n_0 & \text{for } d = 1 \\
  \max \sum_{a=0}^{\max} n_a \cdot c(d-1)^a & \text{for } d > 1
\end{cases}$$  \hspace{1cm} (1)$$

$n_a$ is the number of functions in $N$ that have arity $a$. $\max$ is the maximum arity of functions in the function set.
## Example Numbers for a Simple Function Set

<table>
<thead>
<tr>
<th>Max Depth</th>
<th>Search Space Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
</tr>
<tr>
<td>3</td>
<td>202</td>
</tr>
<tr>
<td>4</td>
<td>81610</td>
</tr>
<tr>
<td>5</td>
<td>$3.5 \times 10^{20}$</td>
</tr>
</tbody>
</table>

Reasonable assumption that the solution is close to the original program.

Profiling the existing code reduces the size of the search space.

Provides the basic units of manipulation, course-grained search.
Plethora of techniques we have yet to exploit.

Simple example: profiling of memory usage to eliminate memory leaks or inefficiencies.
Existing Software as Raw Material
“In practice, a program that makes a mistake in one location often handles the situation correctly in another.”

Prefabs
Research Direction: Code Scavenging

### Ineffective Sorts

**Define HalheatedMergesort(list):**

```plaintext
IF LENGTH(list) < 2:
  RETURN LIST
Pivot = INT(LENGTH(list) / 2)
A = HalheatedMergesort(list[:Pivot])
B = HalheatedMergesort(list[Pivot:]):
  // UMMMM
RETURN[A, B] // HERE. SORRY.
```

**Define FirstBogosort(list):**

```plaintext
// AN OPTIMIZED BOGOSORT
// RUNS IN O(N*LOG N)
FOR n FROM 1 TO LOG(LENGTH(list)):
  Shuffle(list)
  IF isSorted(list):
    RETURN LIST
RETURN "KERNEL PAGE FAULT (ERROR CODE: 2)"
```

**Define JogineallyQuickSort(list):**

```plaintext
OK SO YOU CHOOSE A PIVOT
THEN DIVIDE THE LIST IN HALF
FOR EACH HALF:
  CHECK TO SEE IF IT'S SORTED
  NO WAY, IT DOESN'T MATTER
  COMPARE EACH ELEMENT TO THE PIVOT
  THE BIGGER ONES GO IN A NEW LIST
THE EQUAL ONES GO INTO OH
THE SECOND LIST FROM BEFORE
HANG ON, LET ME NAME THE LISTS
THIS IS LIST A
THE NEW ONE IS LIST B
PUT THE BIG ONES INTO LIST B
NOW TAKE THE SECOND LIST
CALL IT LIST, OH, A2
WHICH ONE WAS THE PIVOT IN?
SCRATCH ALL THAT
IT JUST RECURSIVELY CALLS ITSELF
UNTIL BOTH LISTS ARE EMPTY
RIGHT?
NOT EMPTY, BUT YOU KNOW WHAT I MEAN
AM I ALLOWED TO USE THE STANDARD LIBRARIES?
```

**Define PanicSort(list):**

```plaintext
IF isSorted(list):
  RETURN LIST
FOR n FROM 1 TO 10000:
  Pivot = RANDOM(0, LENGTH(list))
  List = LIST[pivot] + LIST[:Pivot]
  IF isSorted(list):
    RETURN LIST
  IF isSorted(list):
    // THIS CAN'T BE HAPPENING
    RETURN LIST
  IF isSorted(list):
    // COME ON COME ON
    RETURN LIST
  // OH JEEZ
  // I'M GONNA BE IN SO MUCH TROUBLE
  LIST = []
  SYSTEM("SHUTDOWN -H +5")
  SYSTEM("RM -RF /")
  SYSTEM("RM -RF ~*/")
  SYSTEM("RM -RF /")
  SYSTEM("RD /S/Q C:/*") // PORTABILITY
RETURN [1, 2, 3, 4, 5]
```

StackSort connects to StackOverflow, searches for 'sort a list', and downloads and runs code snippets until the list is sorted.
Code Scavenging: Stacksort

stacksort

In a recent xkcd's alt text, Randall Munroe suggested stacksort, a sort that searches StackOverflow for sorting functions and runs them until it returns the correct answer. So, I made it. If you like running arbitrary code in your browser, try it out.

stackoverflow_sort([8,6,7,5,3,0,9]);

Try a list of numbers, a string, a list of words or json.

Sort

var output = ;

Output from the function.

output console

http://gkoberger.github.io/stacksort/
Summary

It’s all about existing software.

1. As an Oracle.
2. As a starting point.
3. As readymades.