Program Boosting: Program Synthesis via Crowd-Sourcing

Robert A. Cochran, Loris D’Antoni, Benjamin Livshits, David Molnar and Margus Veanes

Presented by: Sam Witty, Shehzaad Dhuliawala and Samer Nashed
Outline

Introduction (Research Question, Key Ideas, Contributions)

Background (Genetic Programming and Regular Expressions)

Motivating Example

Evaluation

Discussion
Research Question

Can Crowd-Sourced solutions to programming tasks be combined automatically to boost performance?
Key Idea

Many common programming tasks are

1. Surprisingly complex, such that even expert programmers may struggle
Key Idea

Many common programming tasks are

1. Surprisingly complex, such that even expert programmers may struggle
2. Easily specified (at least to a good approximation) in English
Key Idea

Many common programming tasks are

1. Surprisingly complex, such that even expert programmers may struggle
2. Easily specified (at least to a good approximation) in English
3. Nuanced enough that different programmers will fail in different ways

These attributes make tasks prime candidates for genetic programming towards program synthesis
Conventional Programming

Blood, sweat, and tears

for i in people.data.users:
    response = client.api.statuses.user_timeline.get(screen_name=i.screen_name)
    print('Got', len(response.data), 'tweets from', i.screen_name)
    if len(response.data) != 0:
        ldate = response.data[0]['created_at']
        ldate2 = datetime.strptime(ldate, '%a %b %d %H:%M:%S +0000 %Y')
        today = datetime.now()
        howlong = (today - ldate2).days
        if howlong < daywindow:
            print(i.screen_name, 'has tweeted in the past', daywindow,
                  totaltweets += len(response.data)
        for j in response.data:
            if j.entities.urls:
                for k in j.entities.urls:
                    newurl = k['expanded_url']
                    urlset.add((newurl, j.user.screen_name))
            else:
                print(i.screen_name, 'has not tweeted in the past', daywindow)

 Mostly working program
Program Boosting

Genetic Programming

The one true solution

Flawed programs

Image sources - 4H, The Economist
Contributions

1. Proposal of new technique: Program Boosting
Contributions

1. Proposal of new technique: Program Boosting
2. Implementation of genetic programming for regular expressions using custom-designed crossover and mutation operations
Contributions

1. Proposal of new technique: Program Boosting
2. Implementation of genetic programming for regular expressions using custom-designed crossover and mutation operations
3. Proposal of a new genetic programming paradigm in which the fitness function is evolved along with the candidate programs
Contributions

1. Proposal of new technique: Program Boosting
2. Implementation of genetic programming for regular expressions using custom-designed crossover and mutation operations
3. Proposal of a new genetic programming paradigm in which the fitness function is evolved along with the candidate programs
4. Release of the tool, CROWDBOOST
Contributions

1. Proposal of new technique: Program Boosting
2. Implementation of genetic programming for regular expressions using custom-designed crossover and mutation operations
3. Proposal of a new genetic programming paradigm in which the fitness function is evolved along with the candidate programs
4. Release of the tool, CROWDBOOST
5. First use of genetic programming on automata over complex alphabets, in this case UTF-16
Contributions

1. Proposal of new technique: Program Boosting
2. Implementation of genetic programming for regular expressions using custom-designed crossover and mutation operations
3. Proposal of a new genetic programming paradigm in which the fitness function is evolved along with the candidate programs
4. Release of the tool, CROWDBOOST
5. First use of genetic programming on automata over complex alphabets, in this case UTF-16
6. Evaluation of the proposed method on 465 regular expressions
Outline

Introduction (Research Question, Key Ideas, Contributions)

Background (Genetic Programming and Regular Expressions)

Motivating Example

Evaluation

Discussion
Background: Genetic Programming

Genetic Programming is a technique wherein a computer program is evolved from some seed using a generic algorithm (often).
Background: Genetic Programming

Genetic Programming is a technique wherein a computer program is evolved from some seed using a generic algorithm (often).

Three main components
Background: Genetic Programming

Genetic Programming is a technique wherein a computer program is evolved from some seed using a generic algorithm (often).

Three main components

Crossover - Merge candidate programs
Background: Genetic Programming

Genetic Programming is a technique wherein a computer program is evolved from some seed using a generic algorithm (often).

Three main components

Crossover - Merge candidate programs

Mutation - Stochastically alter candidate programs
Background: Genetic Programming

Genetic Programming is a technique wherein a computer program is evolved from some seed using a generic algorithm (often).

Three main components

Crossover - Merge candidate programs

Mutation - Stochastically alter candidate programs

Fitness - Evaluate candidate programs
Background: SFA and Regex

Corresponding regex: [0-9]{3}(-)?[0-9]{7}
Outline

Introduction (Research Question, Key Ideas, Contributions)

Background (Genetic Programming and Regular Expressions)

Motivating Example

Evaluation

Discussion
Motivating Example

Determine whether a string is a valid phone number.

Ex: 111-111-1111, 1111111111
Method Overview

1: Input: Programs \(\sigma\), examples \(\phi\), crossover function \(\beta\), mutation function \(\mu\),
2: example generator \(\delta\), fitness function \(\eta\), budget \(\theta\)
3: Output: Boosted program
4: function Boost(\(\sigma, \phi\), \(\beta, \mu, \delta, \eta, \theta\))
5: \(\varphi = \emptyset\) \hspace{1cm} \text{\(\triangleright\) Until perfect or no money}
6: while \((\hat{\eta} < 1.0 \wedge \theta > 0)\) do \hspace{1cm} \text{\(\triangleright\) New examples for this generation}
7:  \hspace{1cm} \text{\(\triangleright\) Crossover \(\sigma_i\) and \(\sigma_j\)}
8:  for all \(\langle \sigma_i, \sigma_j \rangle \in \text{FindCrossoverCandidates}(\sigma)\) do
9:  \hspace{1cm} \text{\(\triangleright\) Generate new examples}
10:  \hspace{1cm} \text{\(\triangleright\) Add this candidate to \(\sigma\)}
11:  end for
12: end for
13: \hspace{1cm} \text{\(\triangleright\) Generate new examples}
14: for all \(\sigma' = \mu(\sigma_i)\) do
15:  \hspace{1cm} \text{\(\triangleright\) Add this candidate to \(\sigma\)}
16:  end for
17: end for
18: \hspace{1cm} \text{\(\triangleright\) Get consensus on these new examples via mturk}
19: \hspace{1cm} \text{\(\triangleright\) and update budget}
20: \(\phi = \phi \cup \varphi\)
21: \(\sigma = \text{Filter}(\sigma)\)
22: \(\langle \hat{\sigma}, \hat{\eta} \rangle = \text{GetBestFitness}(\sigma, \eta)\) \hspace{1cm} \text{\(\triangleright\) Add the newly acquired examples}
23: end while
24: return \(\hat{\sigma}\)
25: end function
Identifying components
Strongly Connected Components
Stretches
Stretches
Single Entry - Single Exit
Single Entry - Single Exit
Resulting Regex: [0-9]{3}-?[0-9]{2}-?[0-9]{4}
Mutations

1. Diminishing
2. Augmenting
Example Mutation

Negative example: 012-456-7890
Assume numbers cannot begin with 0
Example Mutation

Negative example: 012-456-7890
Assume numbers cannot begin with 0
Fitness Function

- A simple approach to calculate fitness for regular expressions would be to calculate accuracy on the training set, but this doesn’t scale well.
Fitness Function

- A simple approach to calculate fitness for regular expressions would be to calculate accuracy on the training set, but this doesn’t scale well.
- Instead, evaluate cardinality of sets.
Fitness Function

- A simple approach to calculate fitness for regular expressions would be to calculate accuracy on the training set, but this doesn’t scale well.
- Instead, evaluate cardinality of sets.

\[
\text{Fitness}(A) = \frac{L(A \cap P) + L(N - A)}{L(P \cup N)}
\]
Evaluation

Regular expressions were pulled from Regexlib.com, blogs, Stack Overflow, and a Bountify task set by the authors.
Evaluation

Regular expressions were pulled from Regexlib.com, blogs, Stack Overflow, and a Bountify task set by the authors

In total, 465 program pairs were used for a variety of tasks (phone numbers, dates, email addresses, URLs)
Evaluation

Regular expressions were pulled from Regexlib.com, blogs, Stack Overflow, and a Bounty task set by the authors.

In total, 465 program pairs were used for a variety of tasks (phone numbers, dates, email addresses, URLs).

Mechanical turk was used to generate new examples, thus evolving the fitness function. Examples were accepted of 60% of turkers reached consensus.
Evaluation

<table>
<thead>
<tr>
<th></th>
<th>Golden set</th>
<th>Candidate regexes</th>
<th>Candidate regex source:</th>
<th>Bountify</th>
<th>Regexlib</th>
<th>Other</th>
</tr>
</thead>
<tbody>
<tr>
<td>Phone numbers</td>
<td>20</td>
<td>8</td>
<td>3</td>
<td>0</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>Dates</td>
<td>31</td>
<td>6</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>Emails</td>
<td>7</td>
<td>10</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>URLs</td>
<td>36</td>
<td>9</td>
<td>4</td>
<td>0</td>
<td>5</td>
<td></td>
</tr>
</tbody>
</table>
## Results - Accuracy

<table>
<thead>
<tr>
<th>Evaluated On...</th>
<th>Golden Set</th>
<th>Evolution Set</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Task</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Phone numbers</td>
<td>initial 0.80</td>
<td>no crowd 0.90</td>
<td>crowd 0.90</td>
</tr>
<tr>
<td>Dates</td>
<td>initial 0.85</td>
<td>no crowd 0.99</td>
<td>crowd 0.97</td>
</tr>
<tr>
<td>Emails</td>
<td>initial 0.71</td>
<td>no crowd 0.86</td>
<td>crowd 0.86</td>
</tr>
<tr>
<td>URLs</td>
<td>initial 0.67</td>
<td>no crowd 0.91</td>
<td>crowd 0.88</td>
</tr>
</tbody>
</table>
## Results - Accuracy

Crowd Boosting does not help with the Golden Set

<table>
<thead>
<tr>
<th>Evaluated On...</th>
<th>Golden Set</th>
<th>Evolved Set</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Boosted</td>
<td>Boosted</td>
</tr>
<tr>
<td>Task</td>
<td>Initial</td>
<td>No Crowd</td>
</tr>
<tr>
<td>Phone numbers</td>
<td>0.80</td>
<td>0.90</td>
</tr>
<tr>
<td>Dates</td>
<td>0.85</td>
<td>0.99</td>
</tr>
<tr>
<td>Emails</td>
<td>0.71</td>
<td>0.86</td>
</tr>
<tr>
<td>URLs</td>
<td>0.67</td>
<td>0.91</td>
</tr>
</tbody>
</table>
Results - Accuracy

Crowd Boosting does not help with the Golden Set

<table>
<thead>
<tr>
<th>Evaluated on...</th>
<th>Golden Set</th>
<th>Evolved Set</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Boosted</td>
<td>Boosted</td>
</tr>
<tr>
<td><strong>Task</strong></td>
<td>initial</td>
<td>no crowd</td>
</tr>
<tr>
<td>Phone numbers</td>
<td>0.80</td>
<td>0.90</td>
</tr>
<tr>
<td>Dates</td>
<td>0.85</td>
<td>0.99</td>
</tr>
<tr>
<td>Emails</td>
<td>0.71</td>
<td>0.86</td>
</tr>
<tr>
<td>URLs</td>
<td>0.67</td>
<td>0.91</td>
</tr>
</tbody>
</table>
Results - Runtime

Runnites are reasonable, especially for synthesizing a program
Results - Runtime

Runtimes are reasonable, especially for synthesizing a program.

This does **not** include the time required to source the RegExs.
Results - Runtime

Runtimes are reasonable, especially for synthesizing a program

This does **not** include the time required to source the RegExs

It is not clear from the paper whether this result includes time required for mechanical turkers
Results - Cost

Overall, costs are reasonable
Results - Cost

Overall, costs are reasonable

Phone and email are cheap. Why?
Outline

Introduction (Research Question, Key Ideas, Contributions)

Background (Genetic Programming and Regular Expressions)

Motivating Example

Evaluation

Discussion
Discussion Questions

1. What kind of ethical issues might arise by involving crowd sourcing in development?
Discussion Questions

2. What are some practical limitations of using large numbers of untrained workers such as mechanical turkers?
Discussion Questions

3. What are some new research questions posed by this new paradigm wherein the fitness function is evolves along with the population?
Discussion Questions

4. What are some weaknesses of genetic programming that persist even through crowd-sourcing?
5. What other open problems might you imagine applying a similar (crowdsourcing followed by some form of program synthesis) approach to?