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

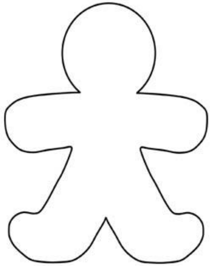
Key Idea

Many common programming tasks are

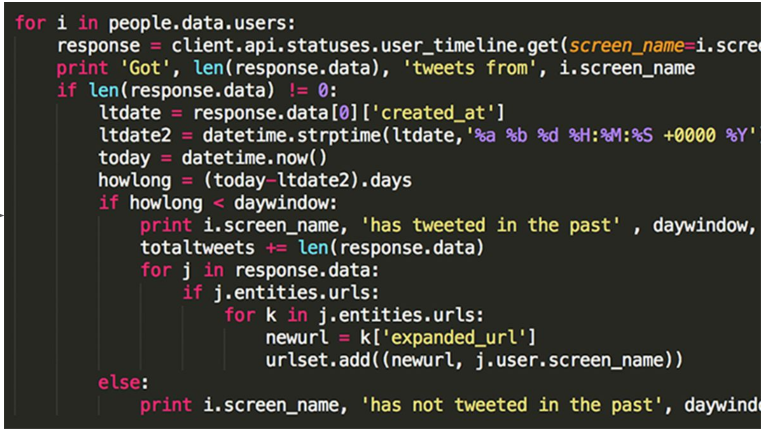
1. Surprisingly complex, such that even expert programmers may struggle
2. Easily specified (at least to a good approximation) in English
3. Nuanced enough that different programmers will fail in different ways

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

Conventional Programming



Blood, sweat, and tears

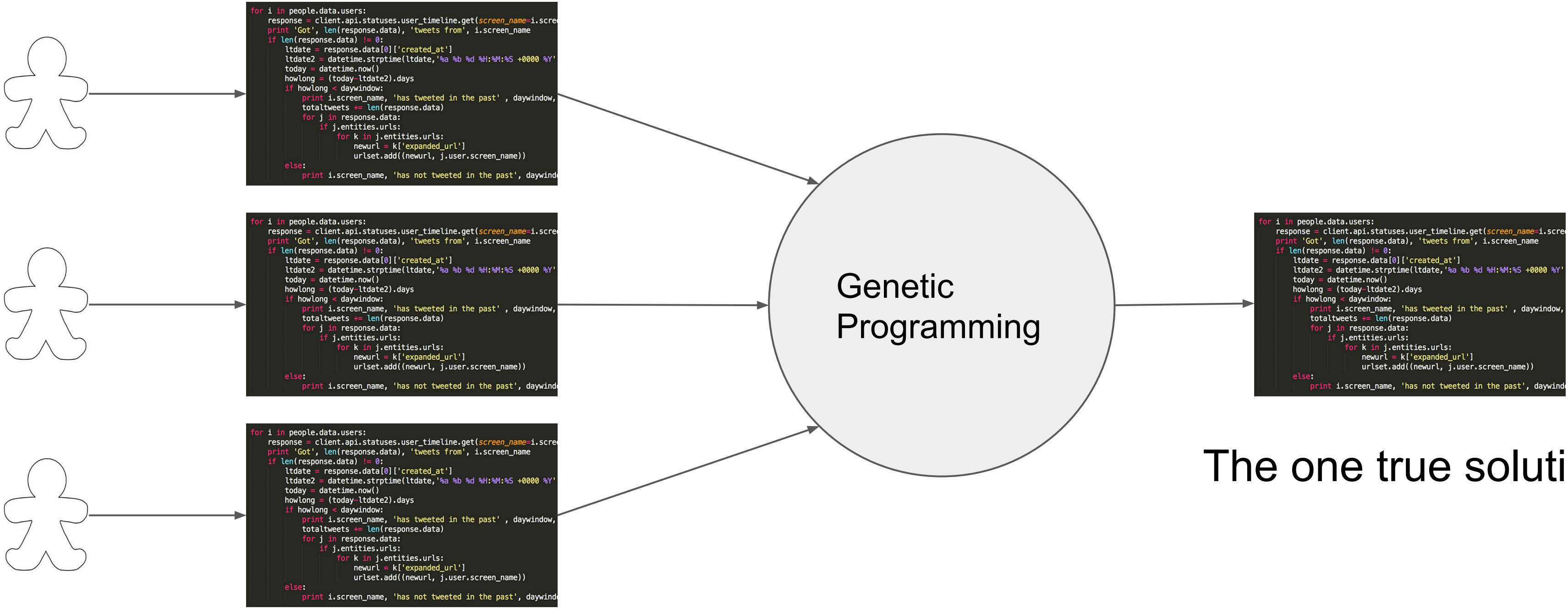


```
for i in people.data.users:
    response = client.api.statuses.user_timeline.get(screen_name=i.scre
    print 'Got', len(response.data), 'tweets from', i.screen_name
    if len(response.data) != 0:
        ltdate = response.data[0]['created_at']
        ltdate2 = datetime.strptime(ltdate,'%a %b %d %H:%M:%S +0000 %Y'
        today = datetime.now()
        howlong = (today-ltdate2).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', daywind
```

Mostly working program

Image sources - 4H, The Economist

Program Boosting



```
for i in people.data.users:
    response = client.api.statuses.user_timeline.get(screen_name=i.scre
    print 'Got', len(response.data), 'tweets from', i.screen_name
    if len(response.data) != 0:
        ltdate = response.data[0]['created_at']
        ltdate2 = datetime.strptime(ltdate,'%a %b %d %H:%M:%S +0000 %Y'
        today = datetime.now()
        howlong = (today-ltdate2).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', daywindo
```

```
for i in people.data.users:
    response = client.api.statuses.user_timeline.get(screen_name=i.scre
    print 'Got', len(response.data), 'tweets from', i.screen_name
    if len(response.data) != 0:
        ltdate = response.data[0]['created_at']
        ltdate2 = datetime.strptime(ltdate,'%a %b %d %H:%M:%S +0000 %Y'
        today = datetime.now()
        howlong = (today-ltdate2).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', daywindo
```

```
for i in people.data.users:
    response = client.api.statuses.user_timeline.get(screen_name=i.scre
    print 'Got', len(response.data), 'tweets from', i.screen_name
    if len(response.data) != 0:
        ltdate = response.data[0]['created_at']
        ltdate2 = datetime.strptime(ltdate,'%a %b %d %H:%M:%S +0000 %Y'
        today = datetime.now()
        howlong = (today-ltdate2).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', daywindo
```

Genetic Programming

```
for i in people.data.users:
    response = client.api.statuses.user_timeline.get(screen_name=i.scre
    print 'Got', len(response.data), 'tweets from', i.screen_name
    if len(response.data) != 0:
        ltdate = response.data[0]['created_at']
        ltdate2 = datetime.strptime(ltdate,'%a %b %d %H:%M:%S +0000 %Y'
        today = datetime.now()
        howlong = (today-ltdate2).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', daywindo
```

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

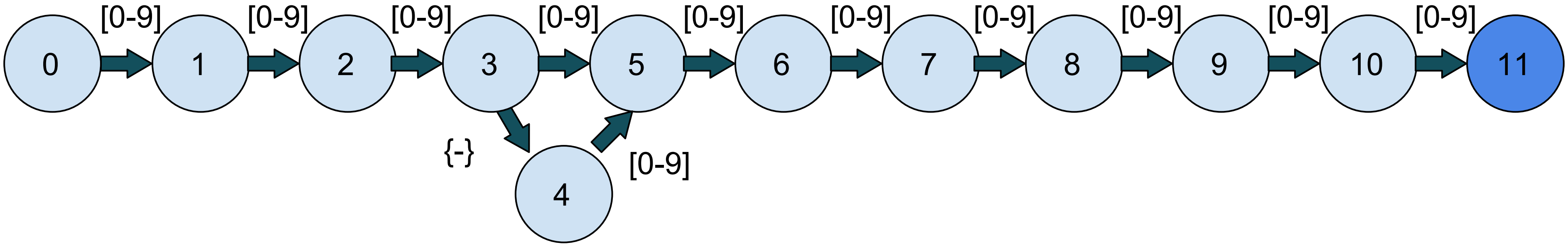
Fitness - Evaluate candidate programs

Animation

Click Me

Source: Rafael Matsunaga

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

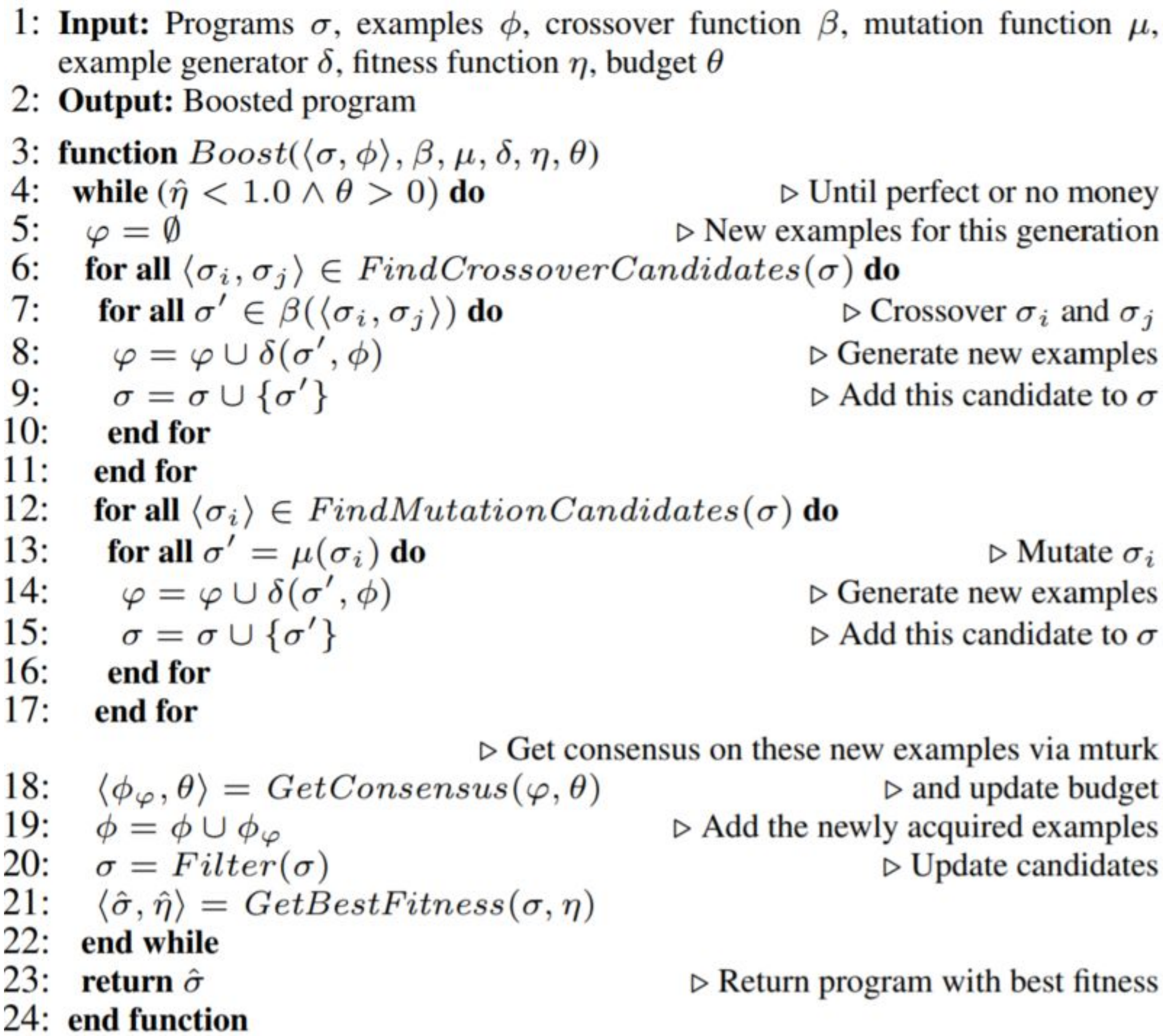
Method Overview

Crossover

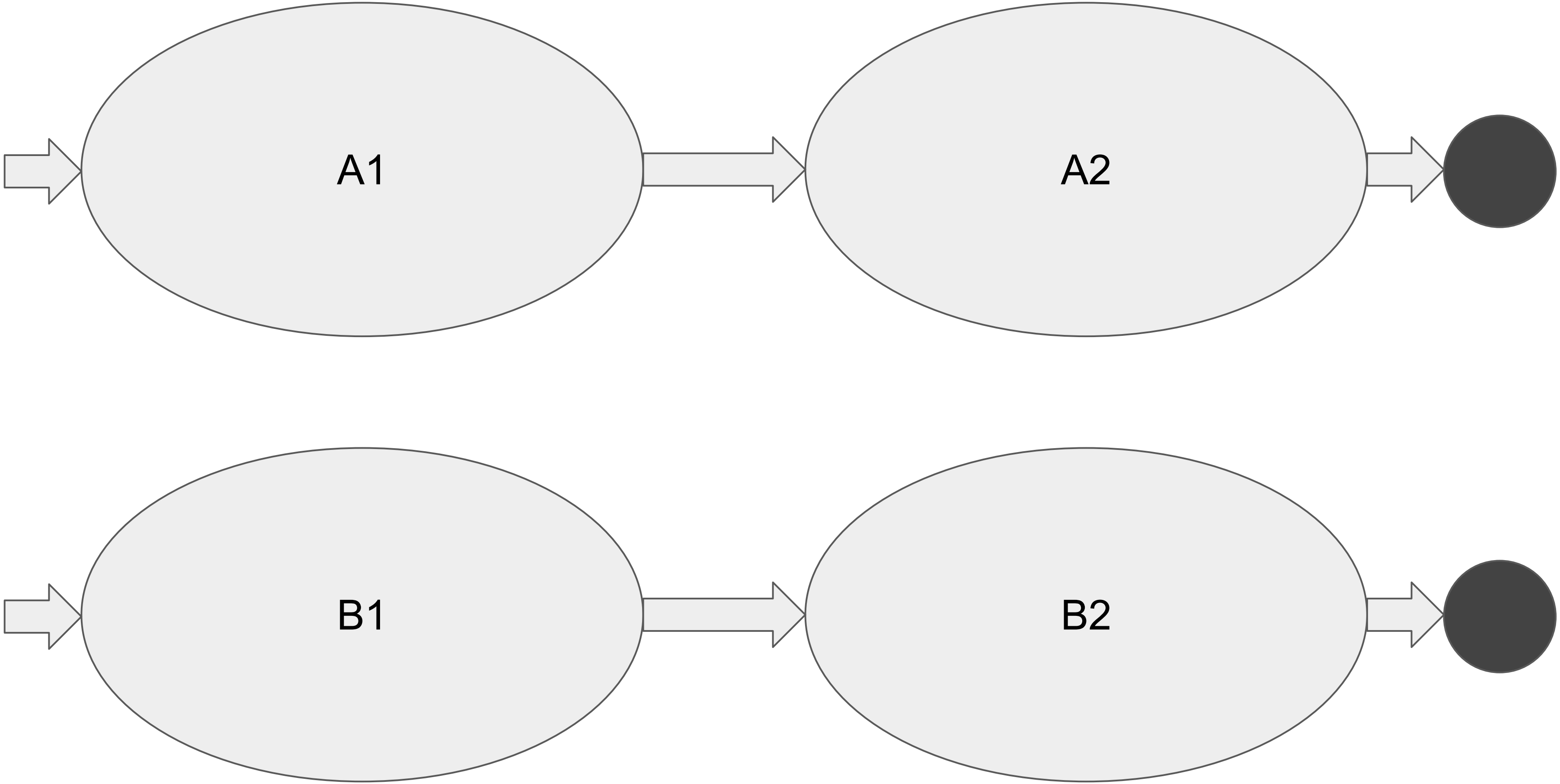
Mutation

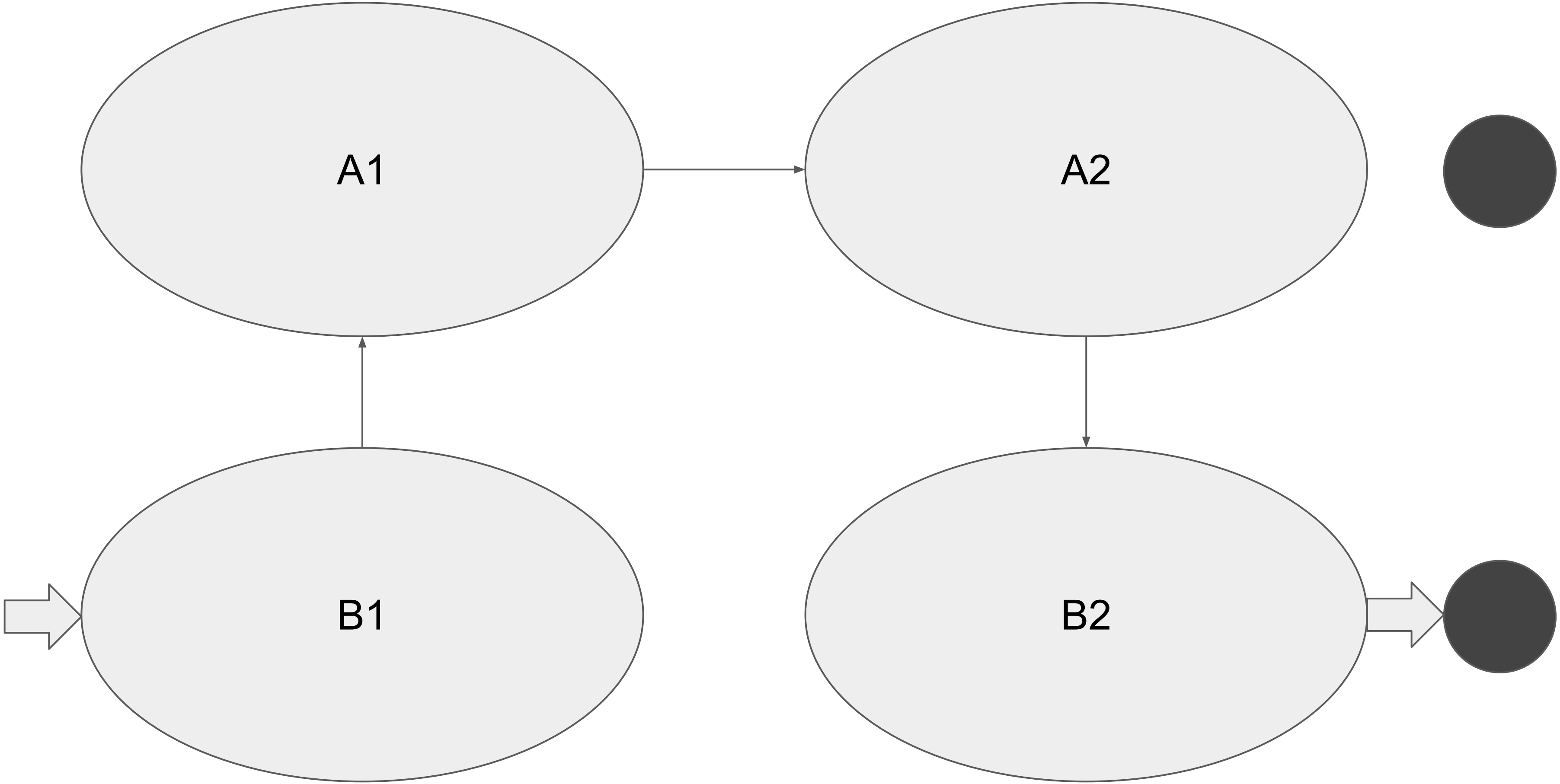
Crowdsource the fitness

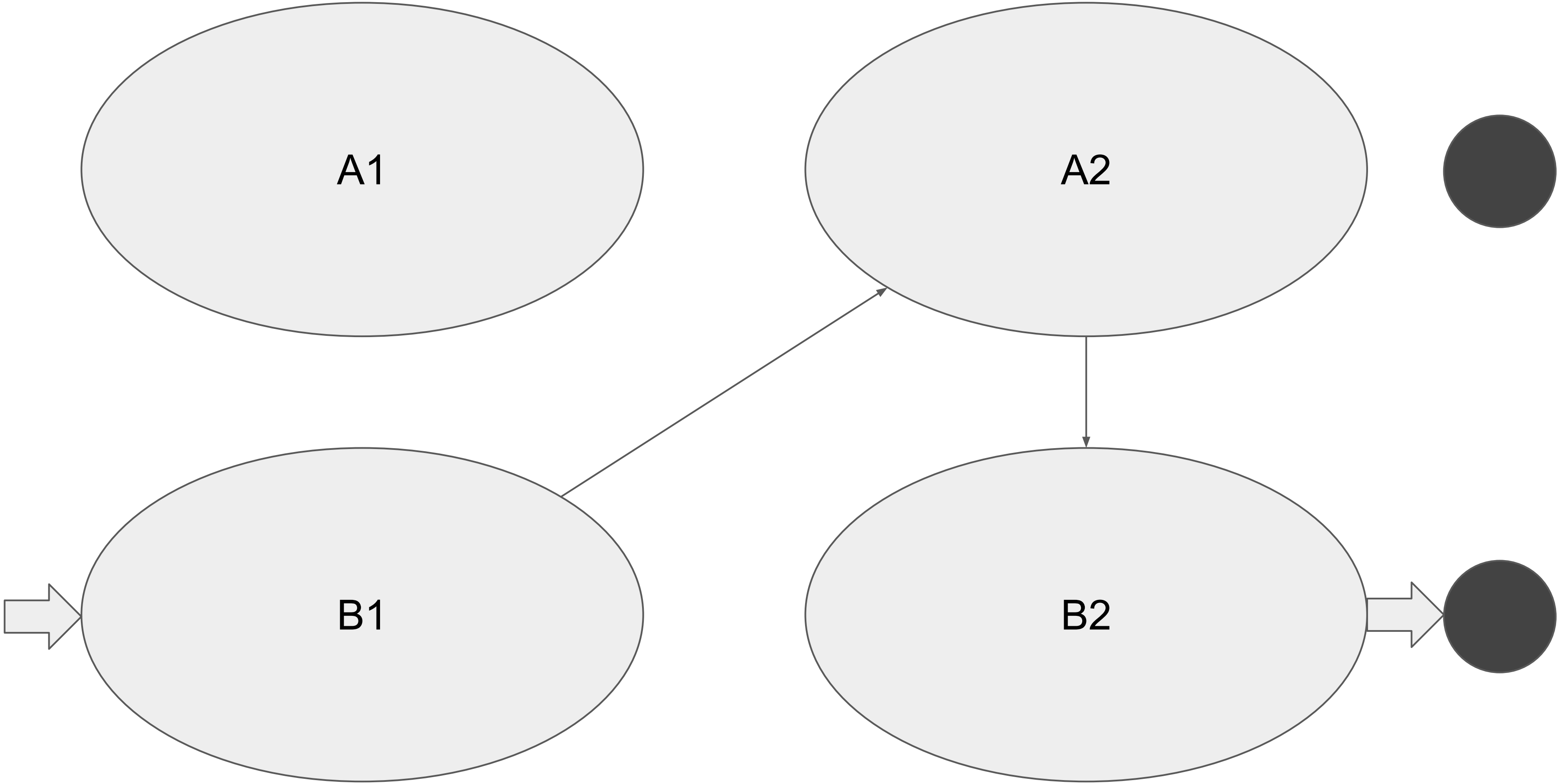
Evaluate New Gen

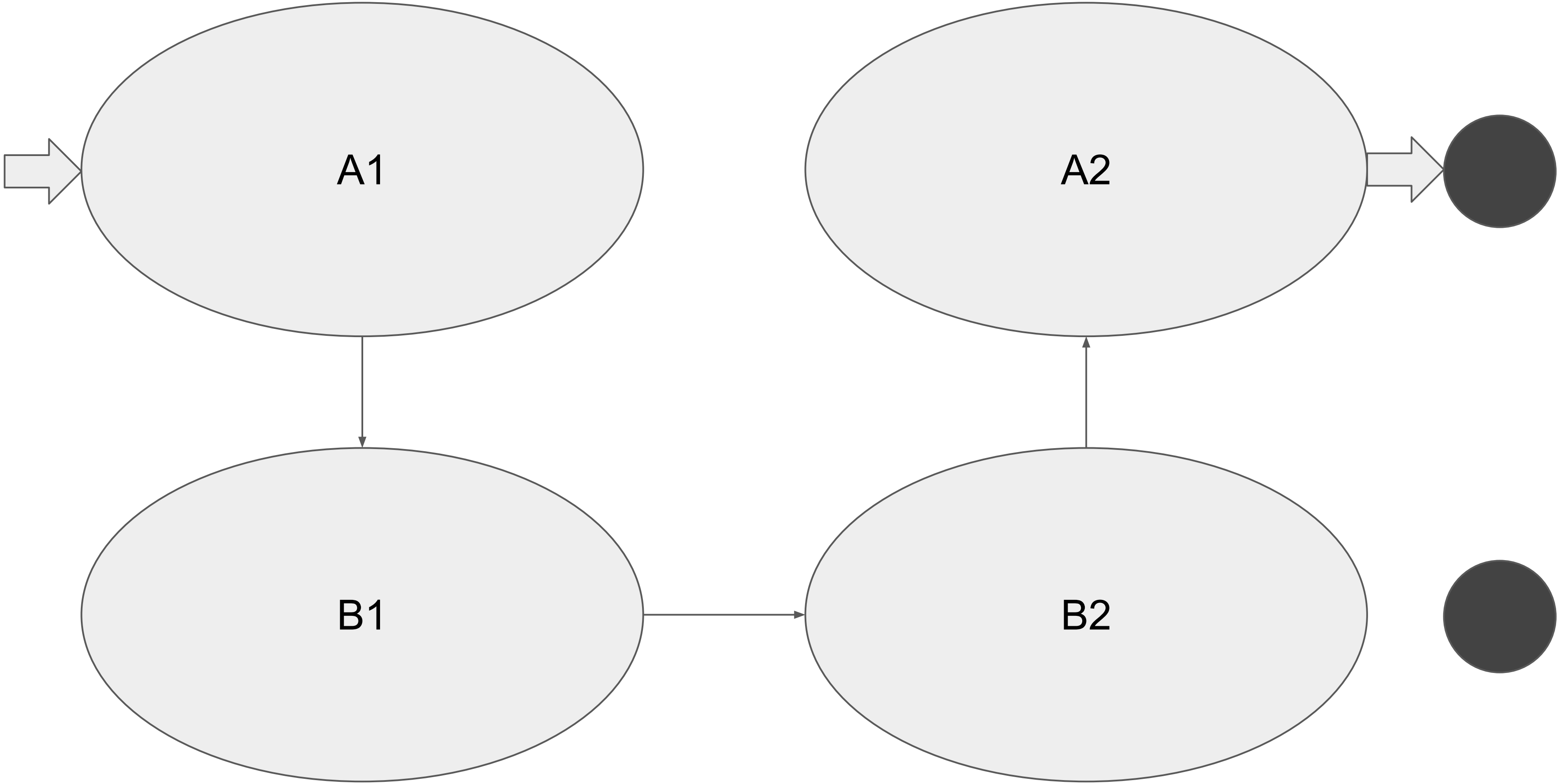


```
Input: Programs $\sigma$, examples $\phi$, crossover function $\beta$, mutation function $\mu$,
    example generator $\delta$, fitness function $\eta$, budget $\theta$
Output: Boosted program
function $Boost(\langle\sigma, \phi\rangle, \beta, \mu, \delta, \eta, \theta)$
  while $(\hat{\eta} < 1.0 \wedge \theta > 0)$ do                          ▷ Until perfect or no money
    $\varphi = \emptyset$                                                    ▷ New examples for this generation
    for all $\langle\sigma_i, \sigma_j\rangle \in FindCrossoverCandidates(\sigma)$ do
      for all $\sigma' \in \beta(\langle\sigma_i, \sigma_j\rangle)$ do       ▷ Crossover $\sigma_i$ and $\sigma_j$
        $\varphi = \varphi \cup \delta(\sigma', \phi)$                       ▷ Generate new examples
        $\sigma = \sigma \cup \{\sigma'\}$                                   ▷ Add this candidate to $\sigma$
      end for
    end for
    for all $\langle\sigma_i\rangle \in FindMutationCandidates(\sigma)$ do
      for all $\sigma' = \mu(\sigma_i)$ do                                   ▷ Mutate $\sigma_i$
        $\varphi = \varphi \cup \delta(\sigma', \phi)$                       ▷ Generate new examples
        $\sigma = \sigma \cup \{\sigma'\}$                                   ▷ Add this candidate to $\sigma$
      end for
    end for
                                                 ▷ Get consensus on these new examples via mturk
    $\langle\phi_\varphi, \theta\rangle = GetConsensus(\varphi, \theta)$     ▷ and update budget
    $\phi = \phi \cup \phi_\varphi$                                          ▷ Add the newly acquired examples
    $\sigma = Filter(\sigma)$                                                ▷ Update candidates
    $\langle\hat{\sigma}, \hat{\eta}\rangle = GetBestFitness(\sigma, \eta)$
  end while
  return $\hat{\sigma}$                                                      ▷ Return program with best fitness
end function
```

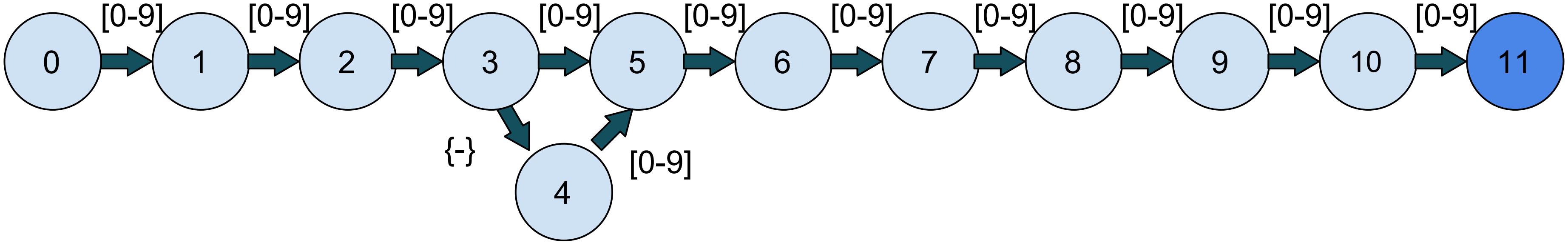

A1
A2
B1
B2

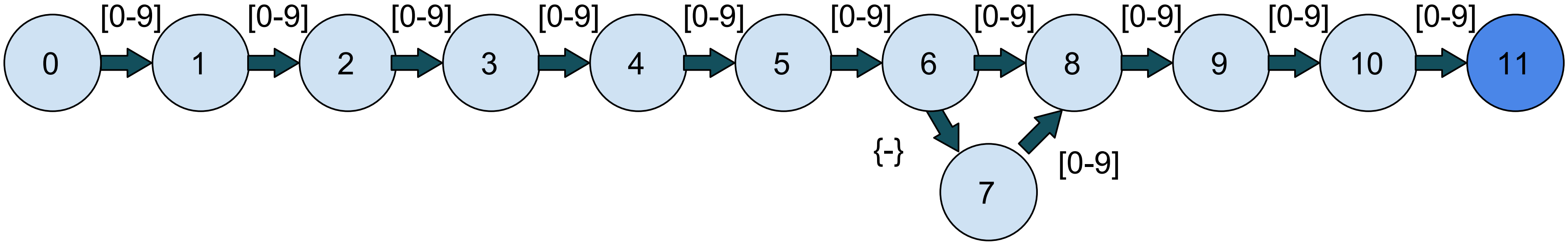

A1
A2
B1
B2


A1
A2
B1
B2

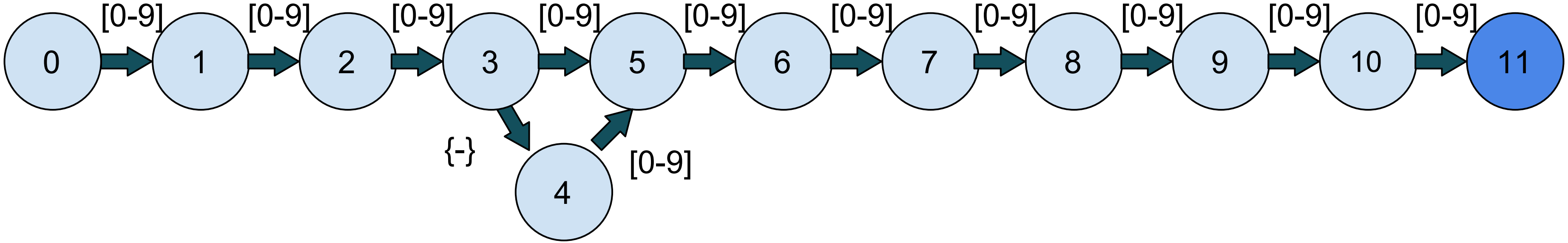

A1
A2
B1
B2

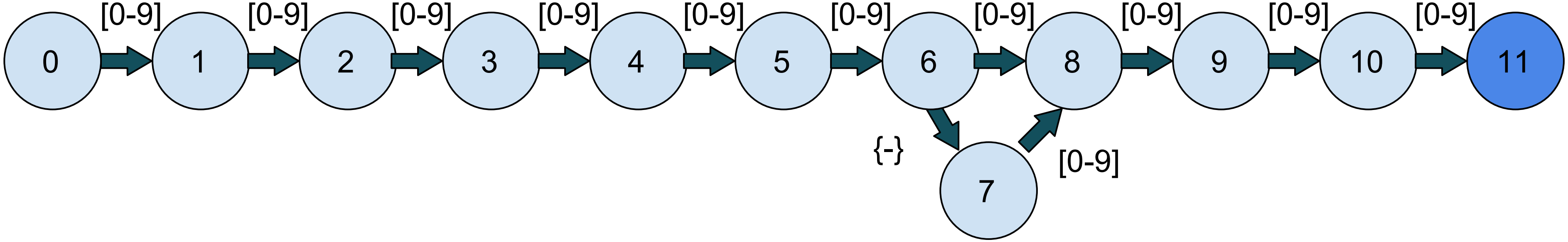
Identifying *components*



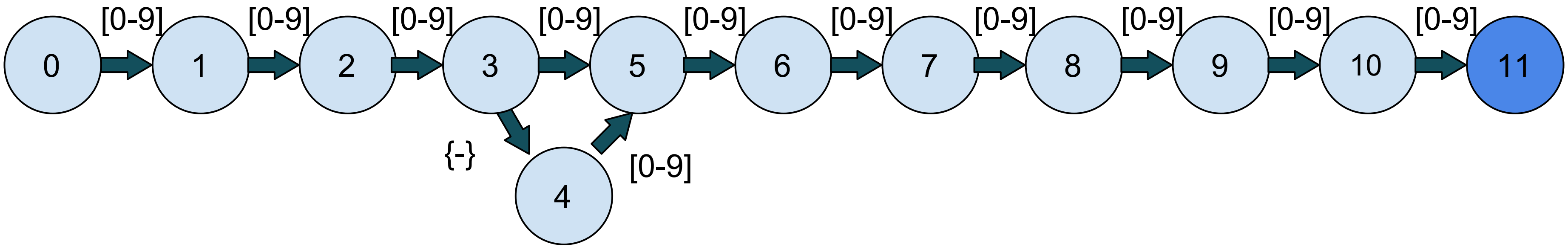


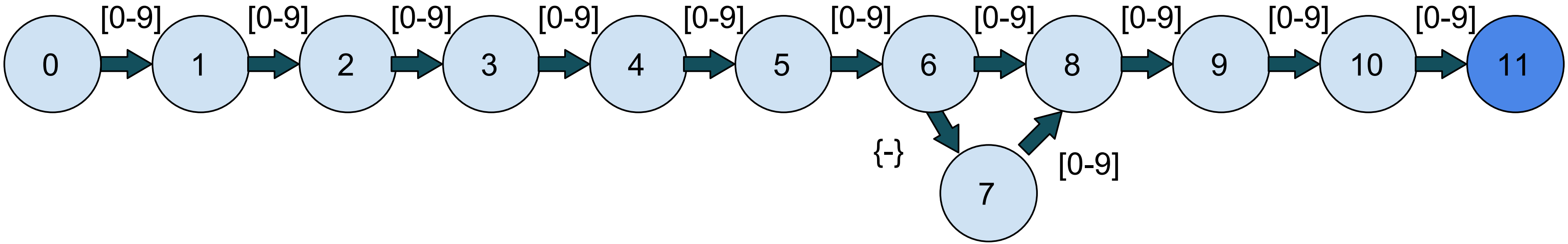
Strongly Connected Components



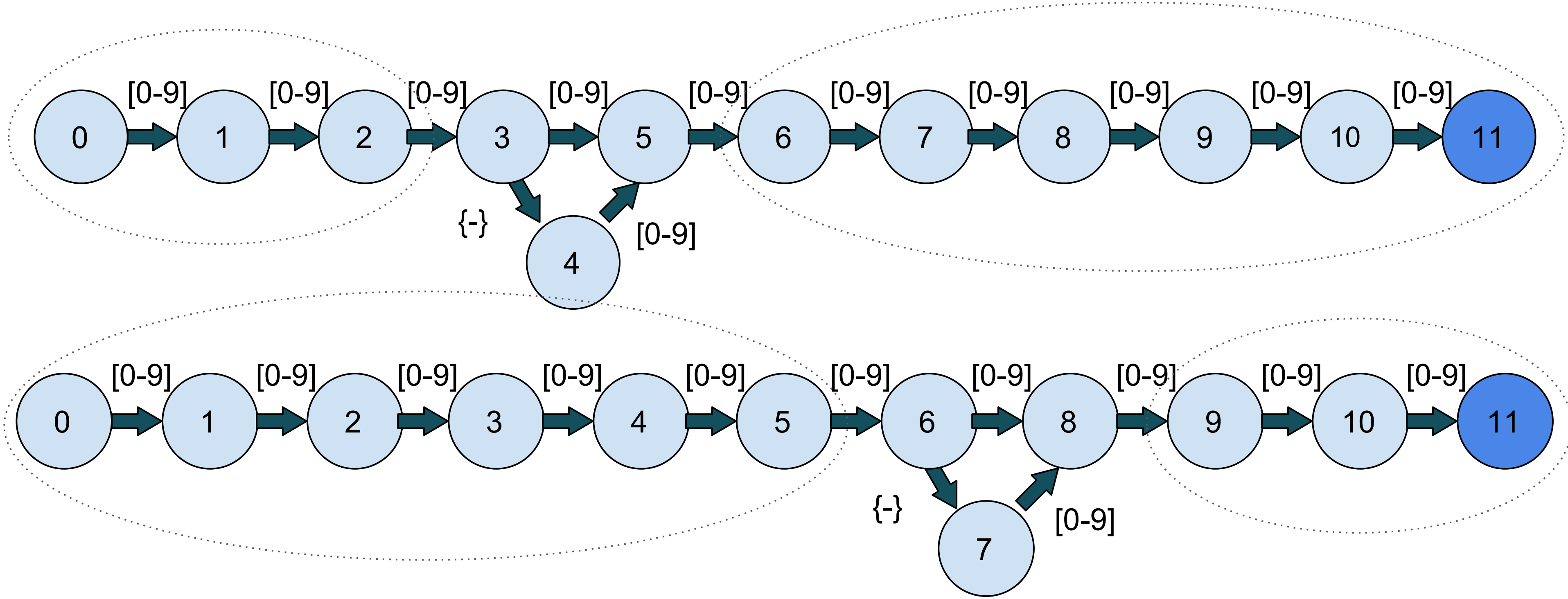


Stretches

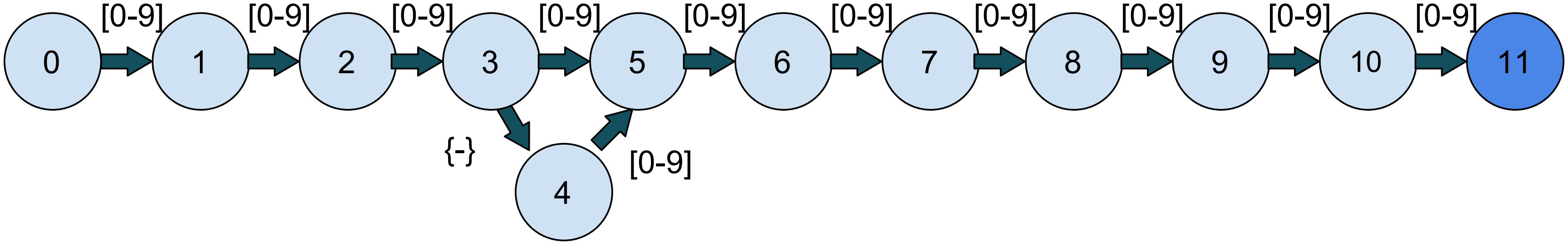

[0-9]
[0-9]
[0-9]
[0-9]
[0-9]
[0-9]
[0-9]
[0-9]
[0-9]
[0-9]
0
1
2
3
5
6
7
8
9
10
11
{-}
4
[0-9]

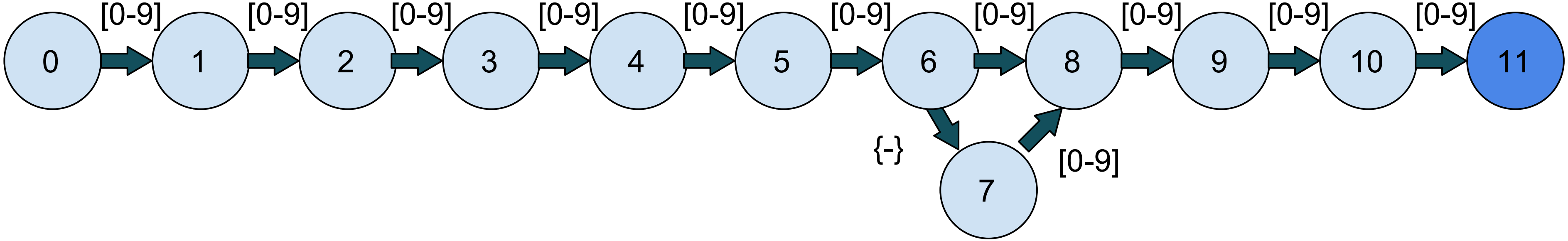

[0-9]
[0-9]
[0-9]
[0-9]
[0-9]
[0-9]
[0-9]
[0-9]
[0-9]
[0-9]
0
1
2
3
4
5
6
8
9
10
11
{-}
7
[0-9]

Stretches

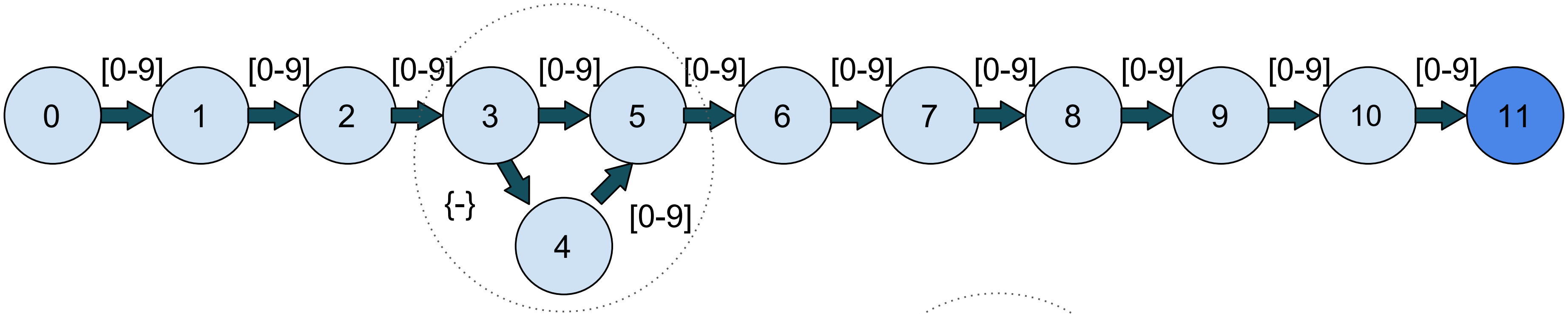

0
[0-9]
1
[0-9]
2
[0-9]
3
[0-9]
5
[0-9]
6
[0-9]
7
[0-9]
8
[0-9]
9
[0-9]
10
[0-9]
11
{-}
4
[0-9]
0
[0-9]
1
[0-9]
2
[0-9]
3
[0-9]
4
[0-9]
5
[0-9]
6
[0-9]
8
[0-9]
9
[0-9]
10
[0-9]
11
{-}
7
[0-9]

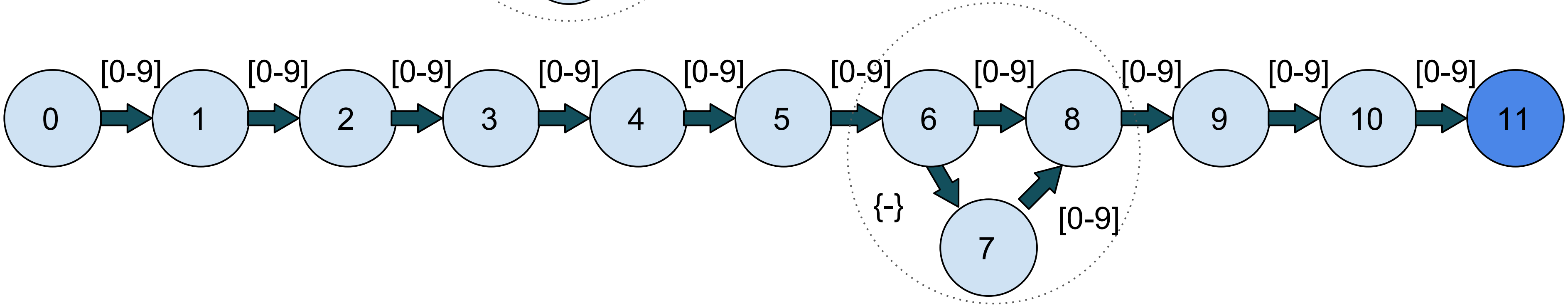
Single Entry - Single Exit

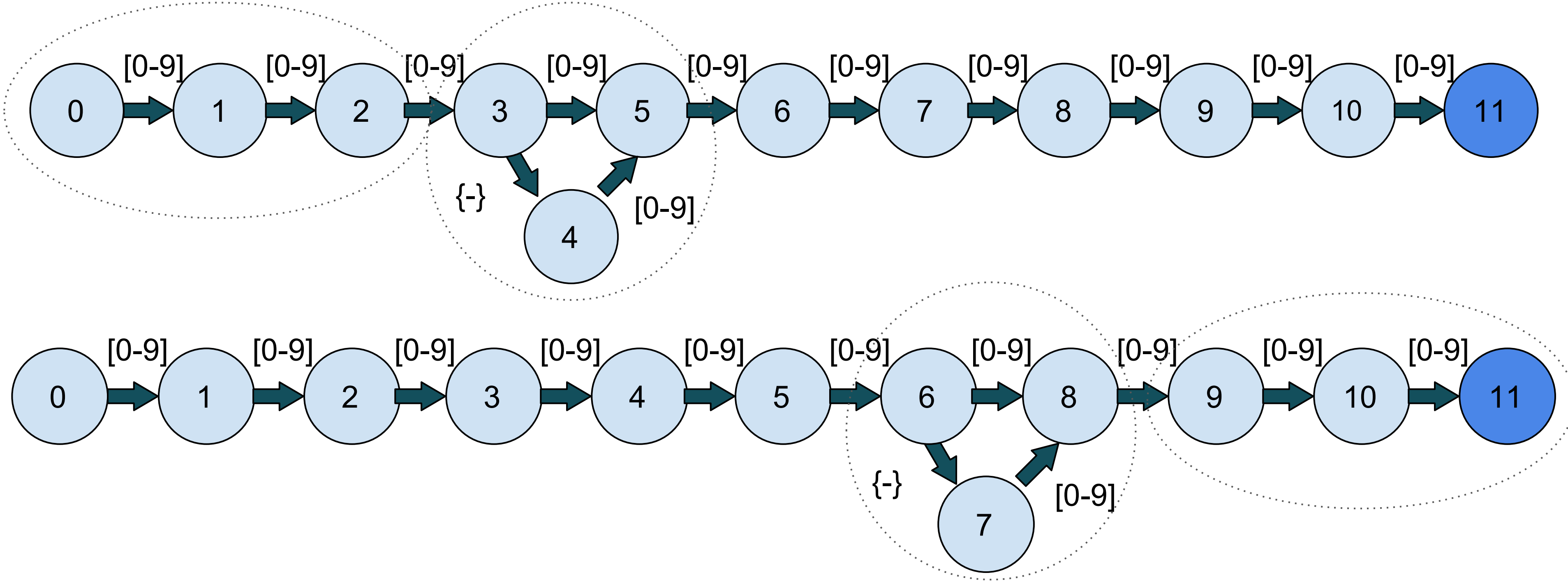


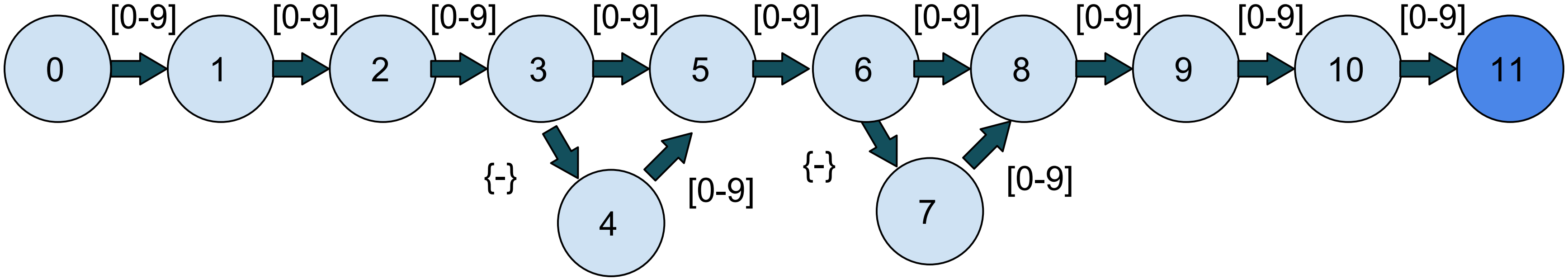


Single Entry - Single Exit






0
[0-9]
1
[0-9]
2
[0-9]
3
[0-9]
5
[0-9]
6
[0-9]
7
[0-9]
8
[0-9]
9
[0-9]
10
[0-9]
11
{-}
4
[0-9]
0
[0-9]
1
[0-9]
2
[0-9]
3
[0-9]
4
[0-9]
5
[0-9]
6
[0-9]
8
[0-9]
9
[0-9]
10
[0-9]
11
{-}
7
[0-9]

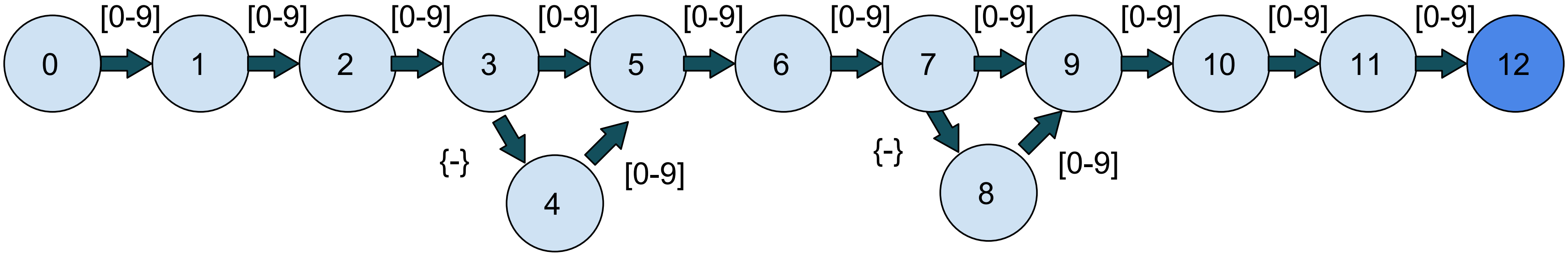


Resulting Regex: [0-9]{3}-?[0-9]{2}-?[0-9]{4}

Mutations

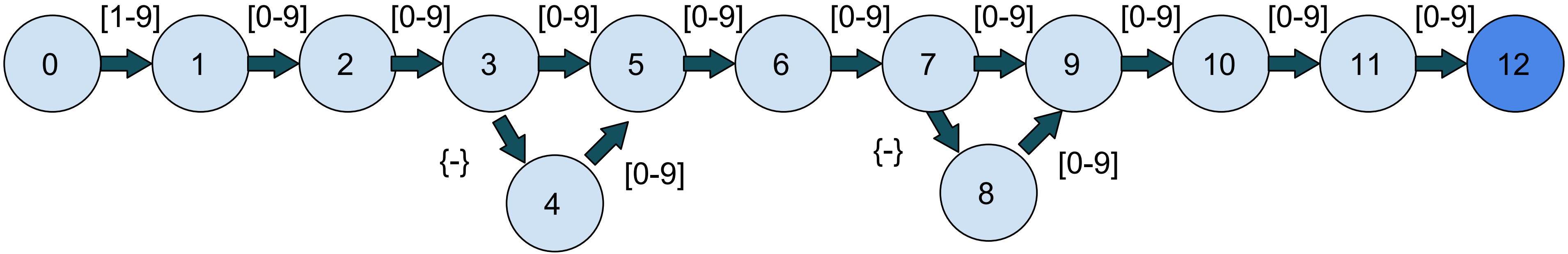
1. Diminishing
2. Augmenting

Example Mutation



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

Example Mutation



Negative example: **0**12-456-7890
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.

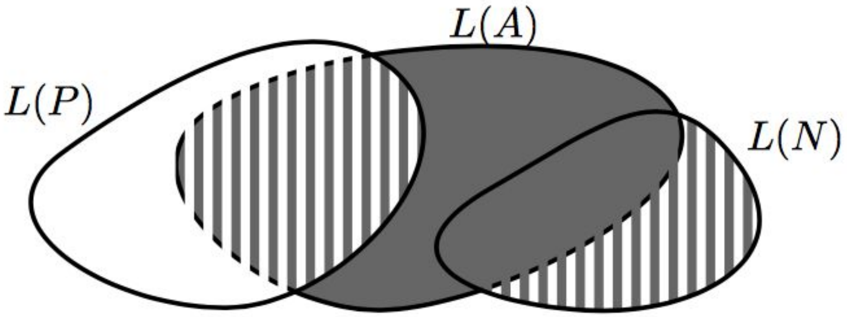
Fitness Function

- 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.

$$\text{Fitness}(A) = (L(A \cap P) + L(N - A))/ L(P \cup N)$$



Outline

Introduction (Research Question, Key Ideas, Contributions)

Background (Genetic Programming and Regular Expressions)

Motivating Example

Evaluation

Discussion

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)

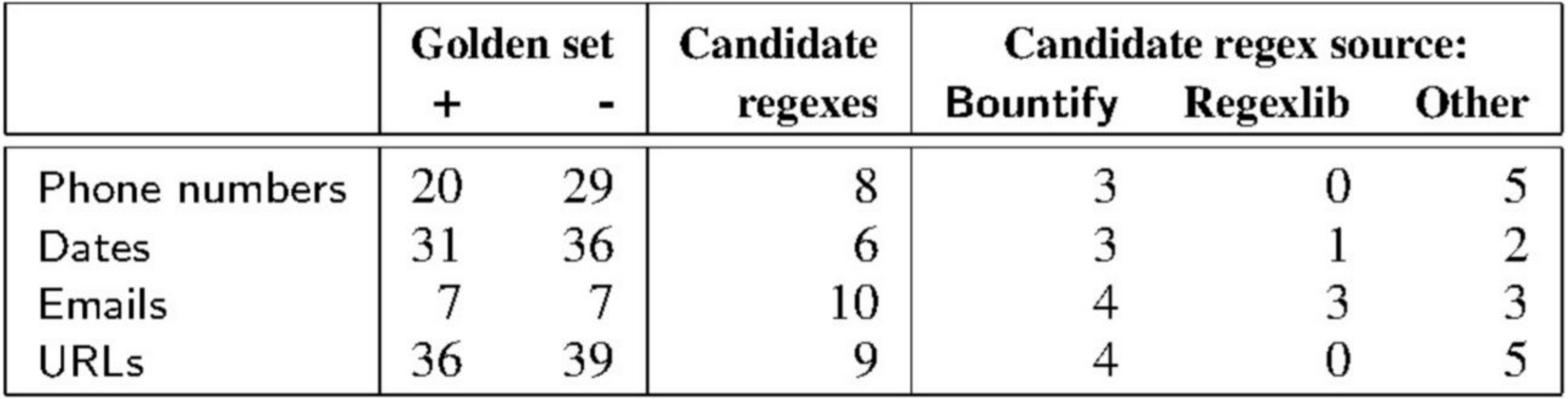
Evaluation

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

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

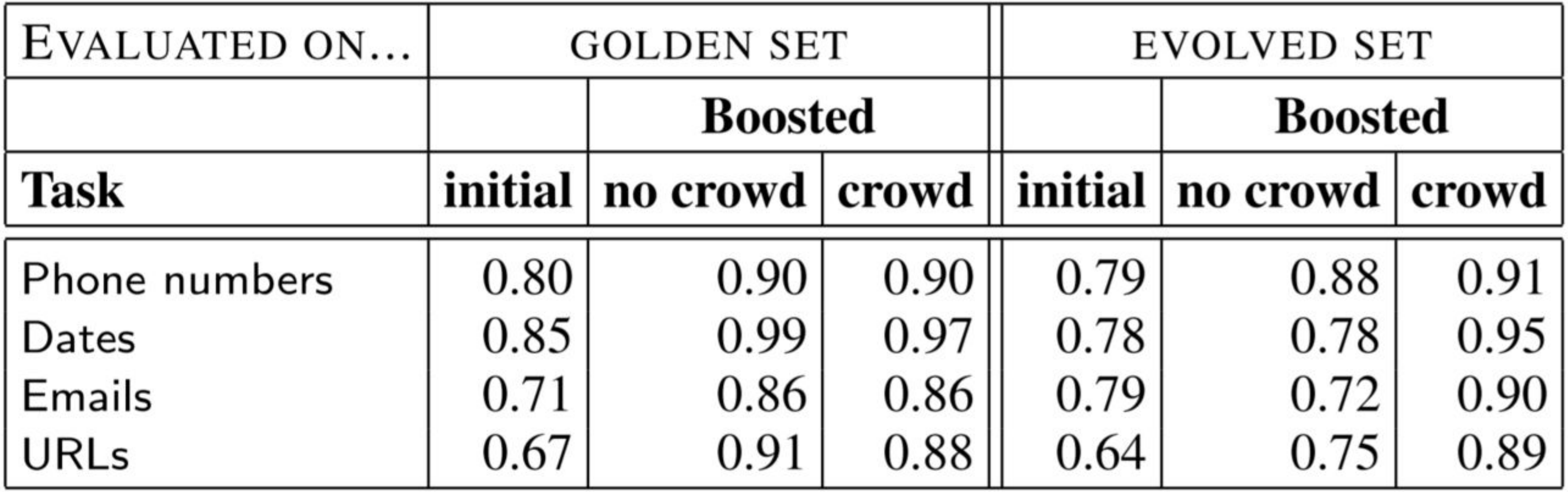
Mechanical turk was used to generate new examples, thus evolving the fitness function. Examples were accepted of 60% of turkers reached consensus

Evaluation



	Golden set		Candidate	Candidate regex source:		
	+	-	regexes	Bountify	Regexlib	Other
Phone numbers	20	29	8	3	0	5
Dates	31	36	6	3	1	2
Emails	7	7	10	4	3	3
URLs	36	39	9	4	0	5

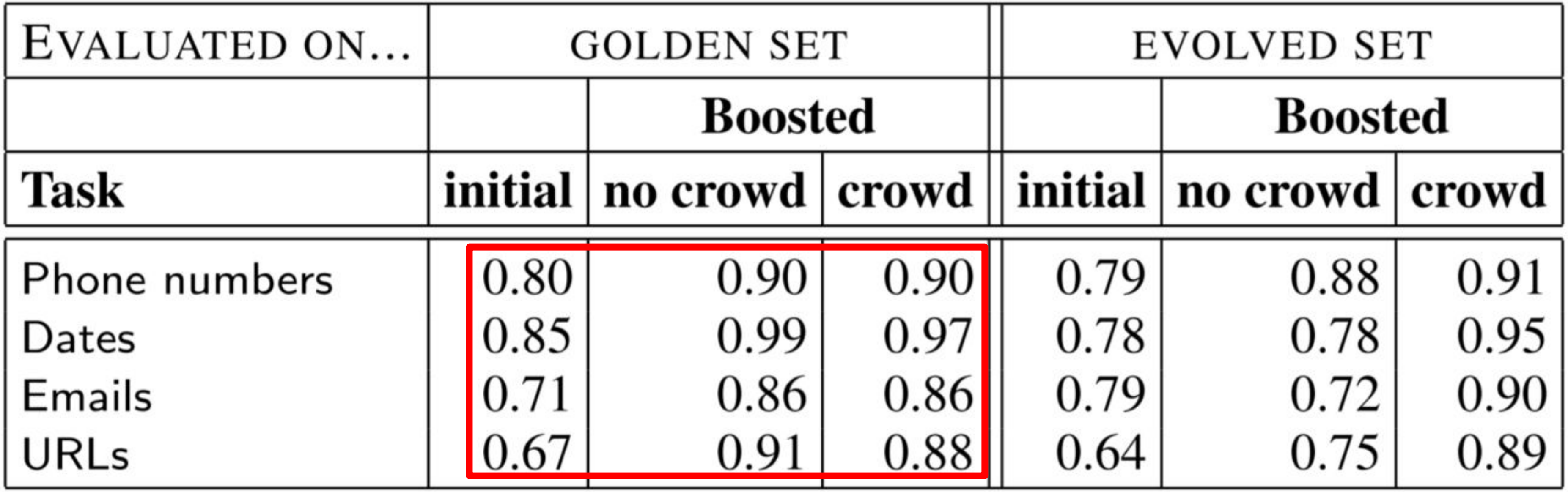
Results - Accuracy



EVALUATED ON...	GOLDEN SET			EVOLVED SET		
		Boosted			**Boosted**	
Task	**initial**	**no crowd**	**crowd**	**initial**	**no crowd**	**crowd**
Phone numbers	0.80	0.90	0.90	0.79	0.88	0.91
Dates	0.85	0.99	0.97	0.78	0.78	0.95
Emails	0.71	0.86	0.86	0.79	0.72	0.90
URLs	0.67	0.91	0.88	0.64	0.75	0.89

Results - Accuracy

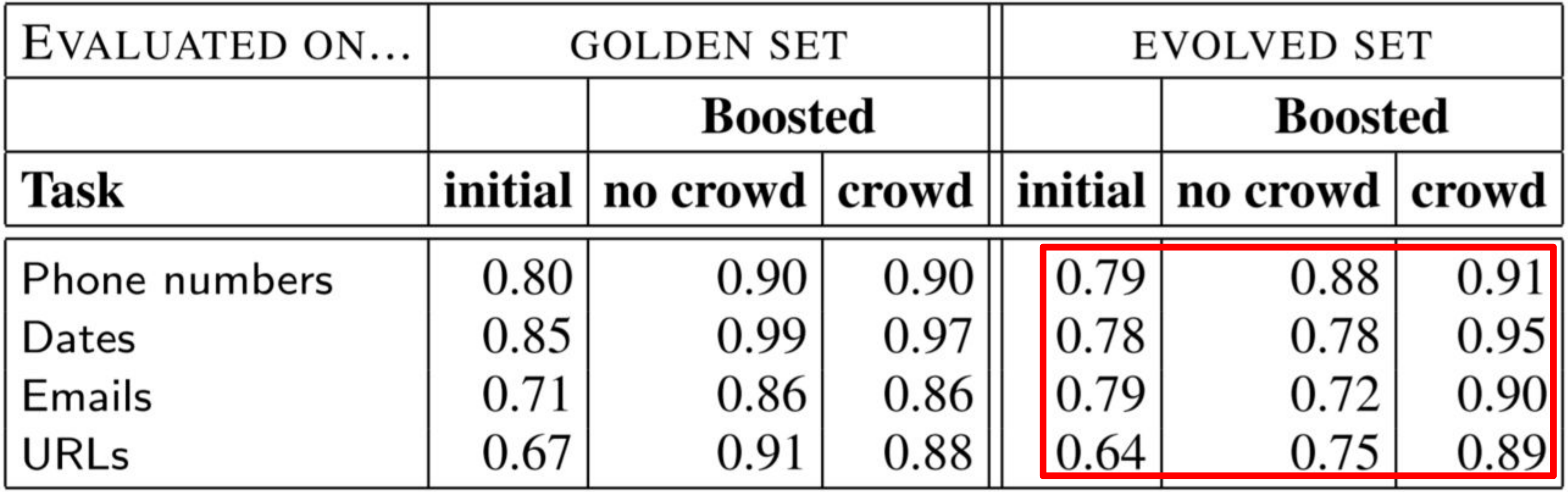
Crowd Boosting does not help with the Golden Set



EVALUATED ON...	GOLDEN SET			EVOLVED SET		
		Boosted			Boosted	
Task	**initial**	**no crowd**	**crowd**	**initial**	**no crowd**	**crowd**
Phone numbers	0.80	0.90	0.90	0.79	0.88	0.91
Dates	0.85	0.99	0.97	0.78	0.78	0.95
Emails	0.71	0.86	0.86	0.79	0.72	0.90
URLs	0.67	0.91	0.88	0.64	0.75	0.89

Results - Accuracy

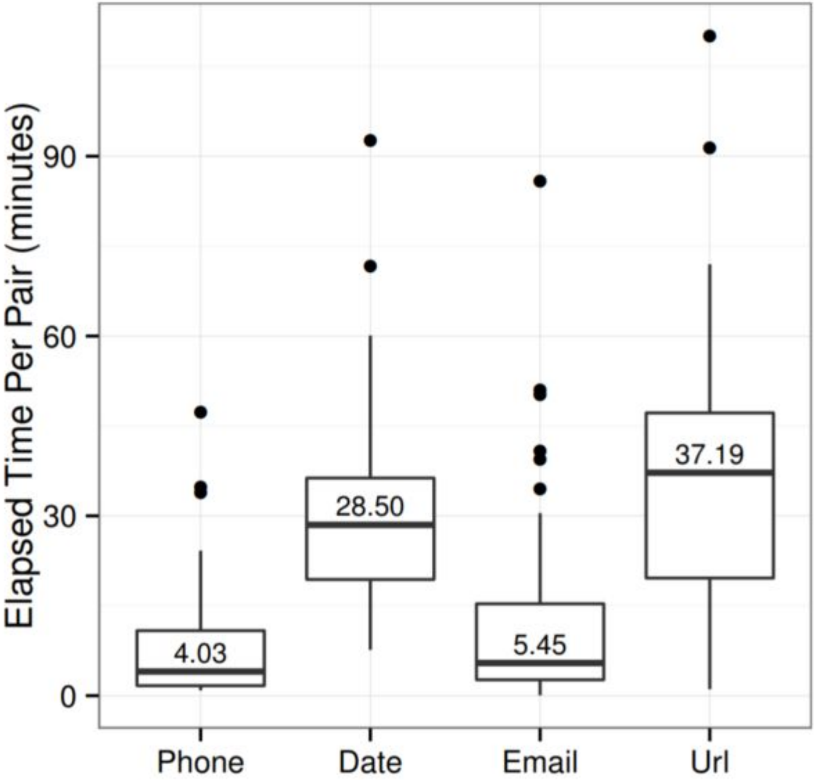
Crowd Boosting does not help with the Golden Set



EVALUATED ON...	GOLDEN SET			EVOLVED SET		
		Boosted			Boosted	
Task	**initial**	**no crowd**	**crowd**	**initial**	**no crowd**	**crowd**
Phone numbers	0.80	0.90	0.90	0.79	0.88	0.91
Dates	0.85	0.99	0.97	0.78	0.78	0.95
Emails	0.71	0.86	0.86	0.79	0.72	0.90
URLs	0.67	0.91	0.88	0.64	0.75	0.89

Results - Runtime

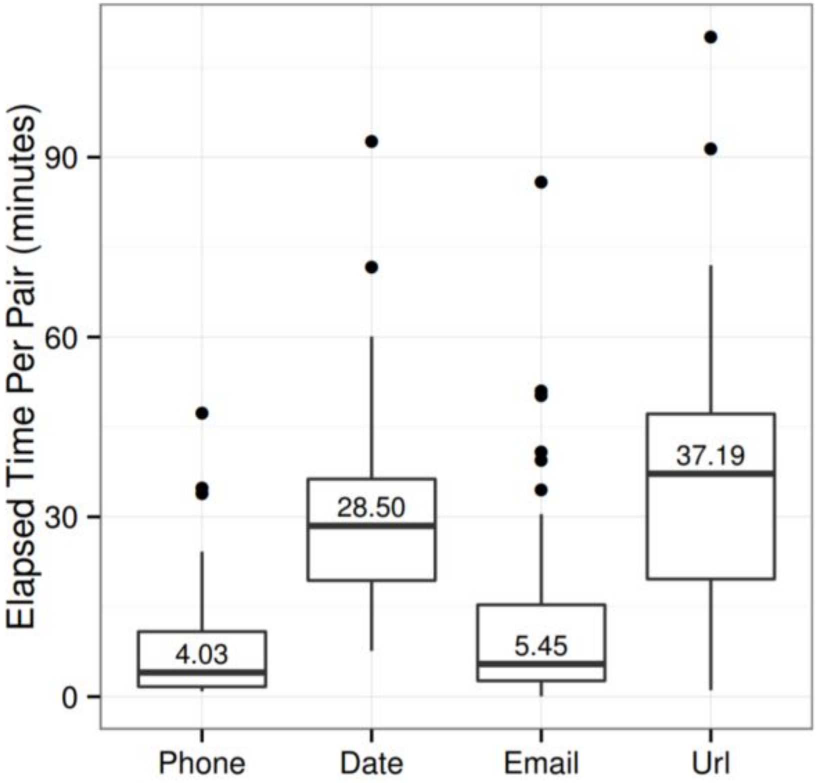
Runtimes are reasonable, especially for synthesizing a program



Results - Runtime

Runtimes are reasonable, especially for synthesizing a program

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

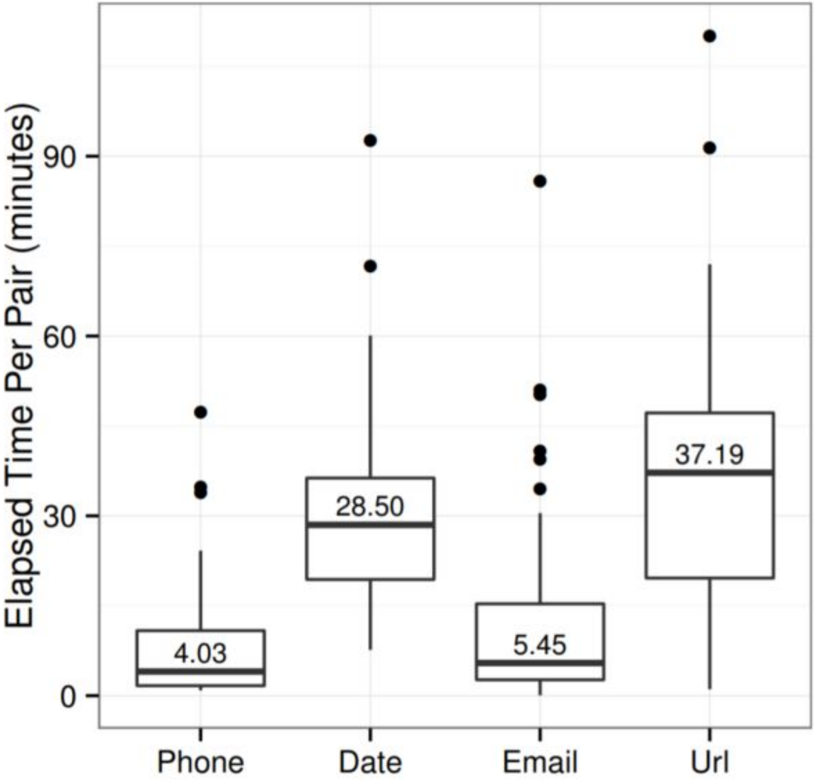


Results - Runtime

Runtimes are reasonable, especially for synthesizing a program

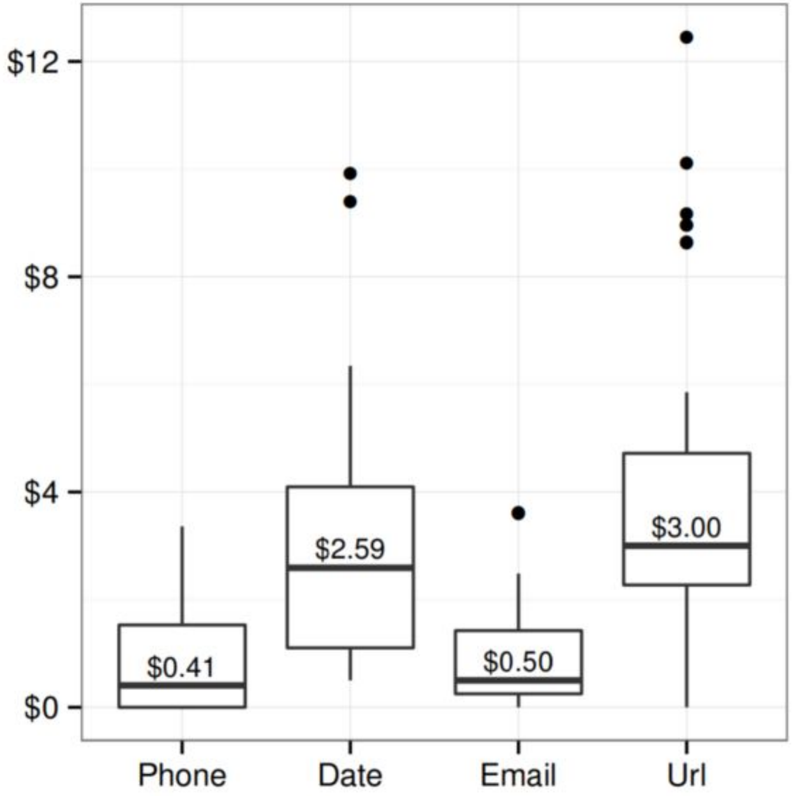
This does **not** include the time required to source the RegExs

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



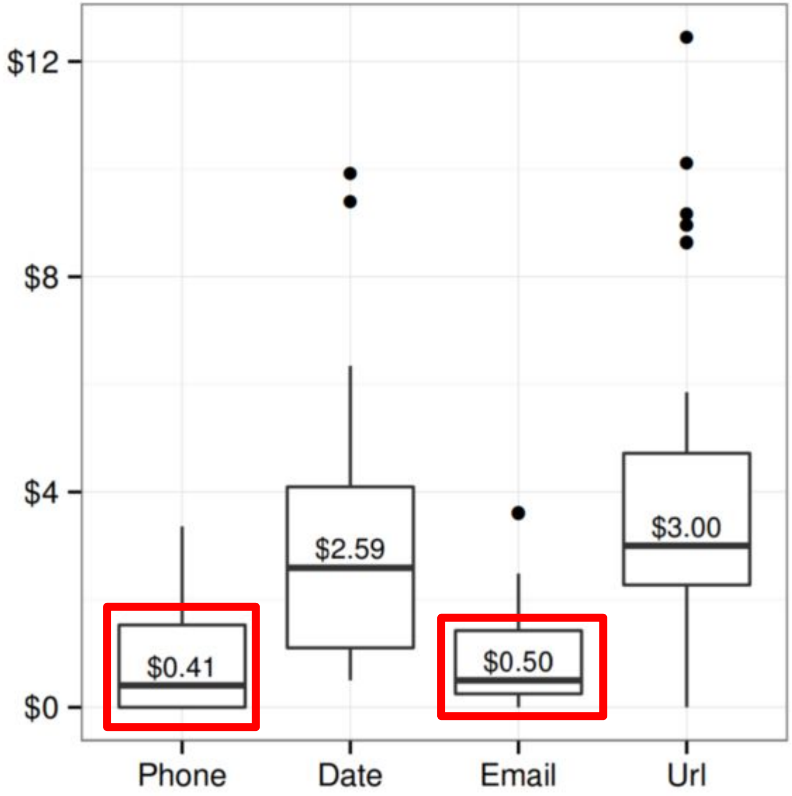
Results - Cost

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?

Discussion Questions

5. What other open problems might you imagine applying a similar (crowdsourcing followed by some form of program synthesis) approach to?