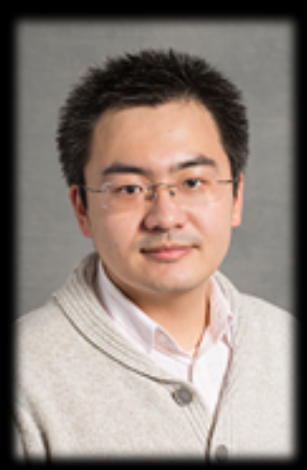
Grow and Serve

Growing Django Citation Services Using SBSE



Yue Jia

Mark Harman

William B. Langdon

Alexandru Marginean

University College London, CREST centre, UK

Python APIs

Google Scholar

Test data

GP

Citation Services

Can we grow useful functionality for readers?

Background

Inspired by **GI (Genetic Improvement)**, a recent research trend in SBSE

GI uses existing code as “genetic material” that helps to automatically improve existing software systems.

source code (text file)

bytecode (executable file)

abstract syntax trees

GI Genetic Improvements

Broken functionality repair

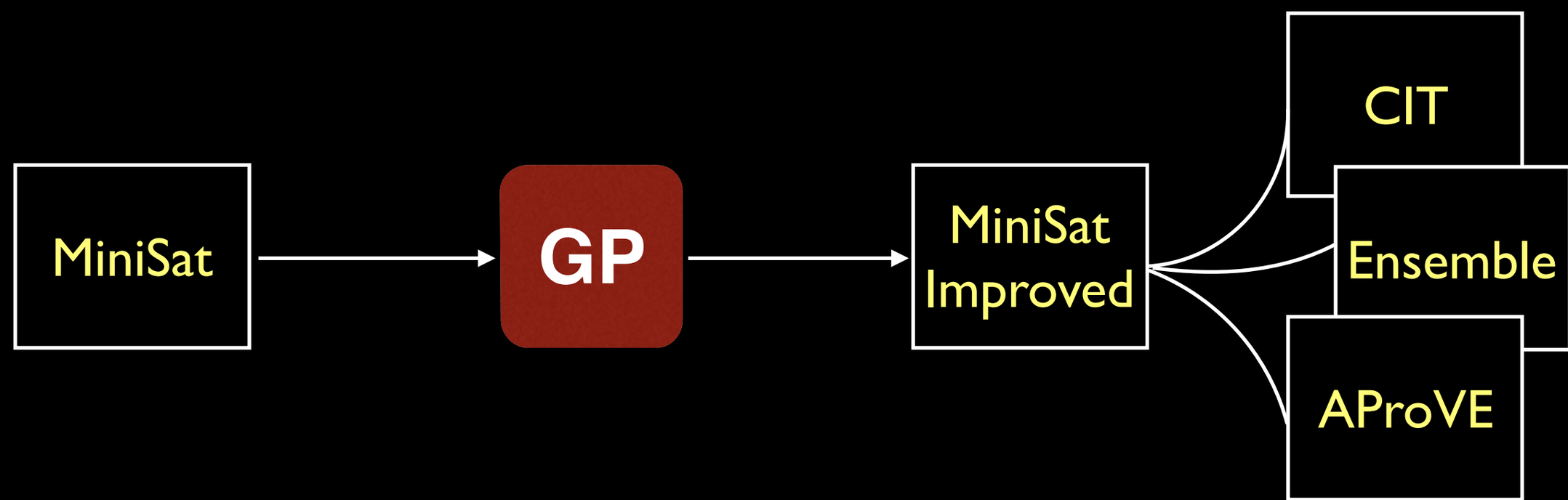
Performance improvement

Non-functional property improvement

Software migration

Software transplantion

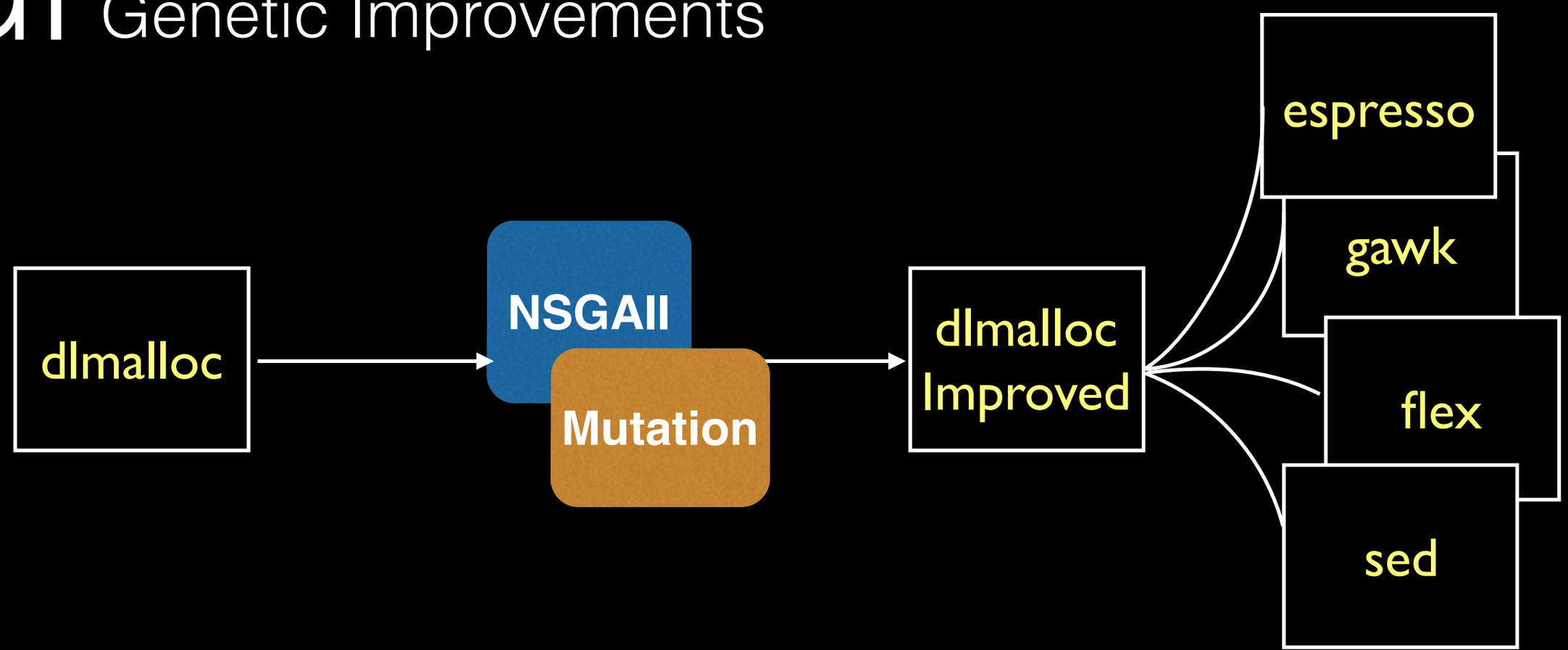
GI Genetic Improvements



"GI can reduce energy consumption by up to 25%"

Bobby R. Bruce, Justyna Petke and Mark Harman
Reducing Energy Consumption Using Genetic Improvement (GECCO'15)

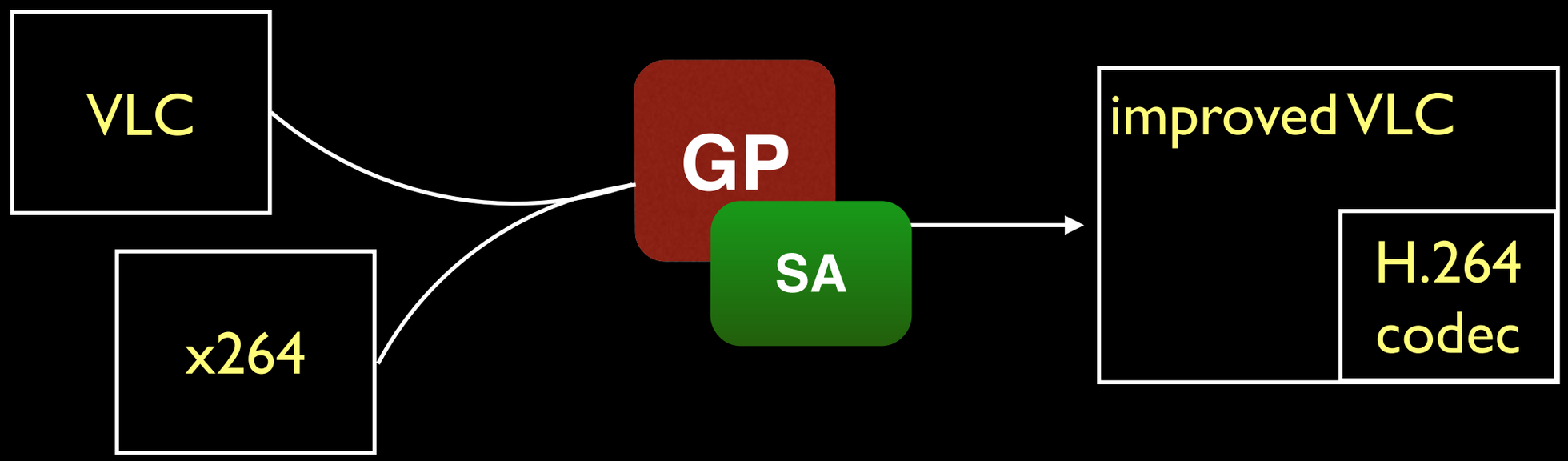
GI Genetic Improvements



"GI can reduce memory consumption by up to 21%"

Fan Wu, Westley Weimer, Mark Harman Yue Jia and Jens Krinke
Deep Parameter Optimisation (GECCO'15)

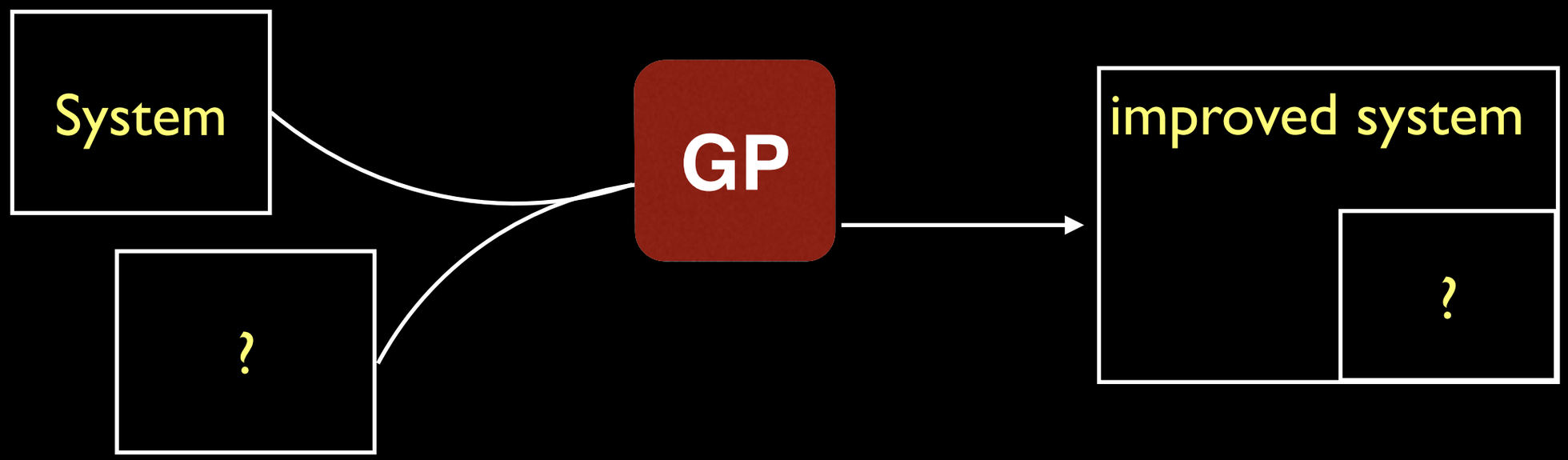
GI Genetic Improvements



It took the automated system 26 hours to complete the transplant, while VLC's manual addition of the code happened over a period of 20 days - wired.co.uk

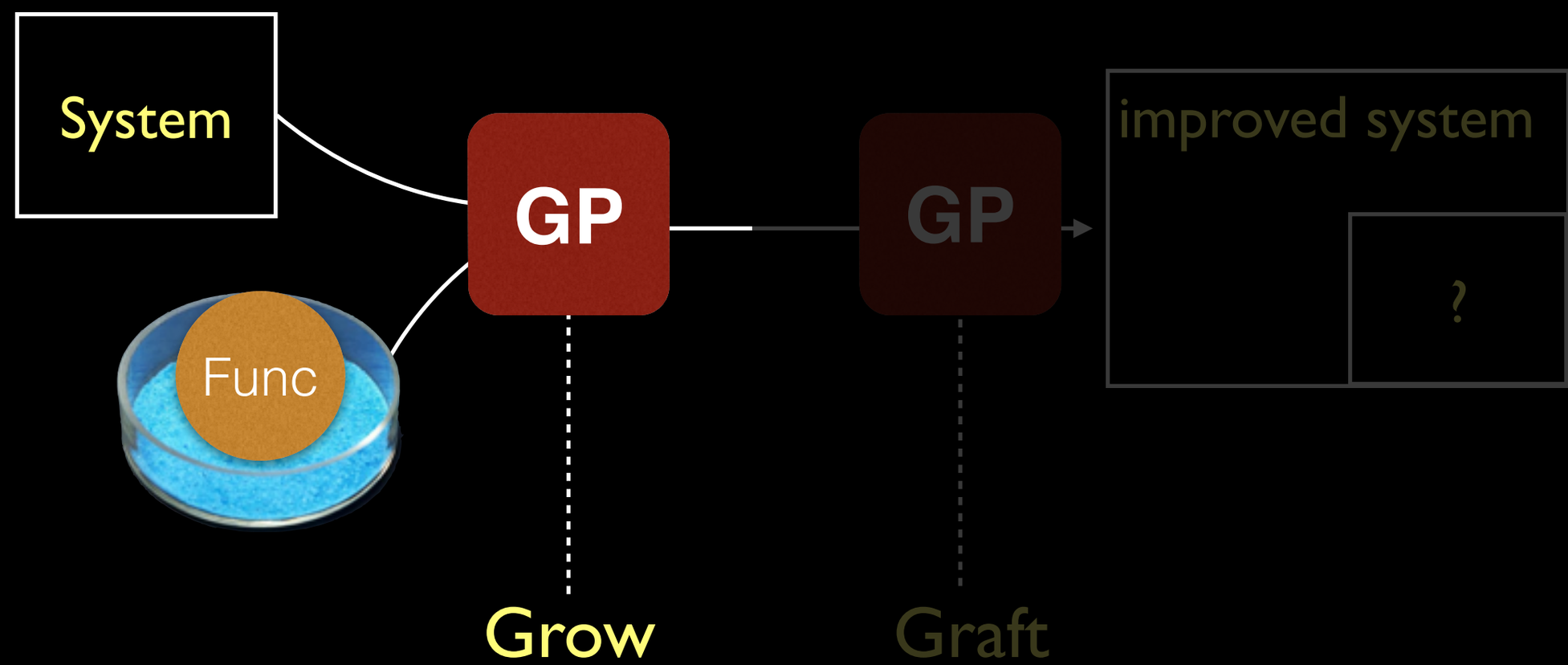
Earl Barr,Mark Harman,Yue Jia,Alexandru Marginean, Justyna Petke
Automated Software Transplantation (ISSTA'15)

GI Genetic Improvements



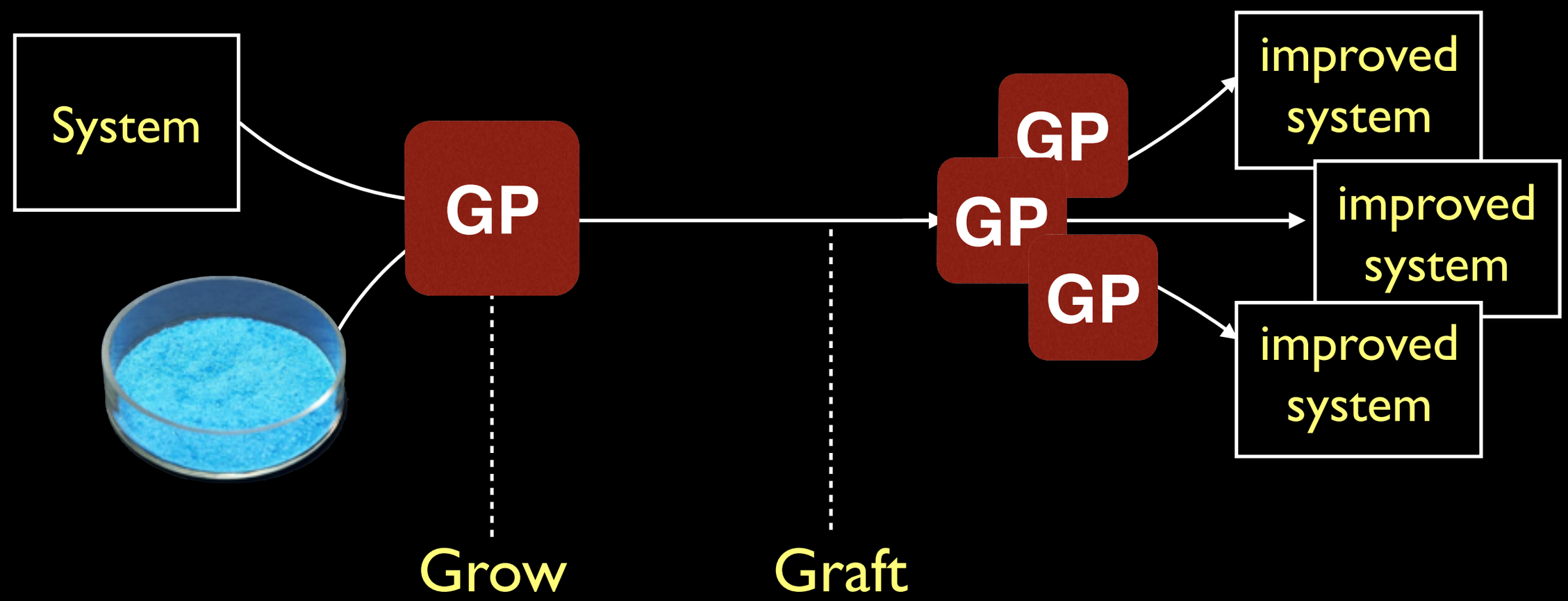
What if we want to transplant a new functionality, which we could not find it in any existing donors?

GGGI Grow and Graft for Genetic Improvements



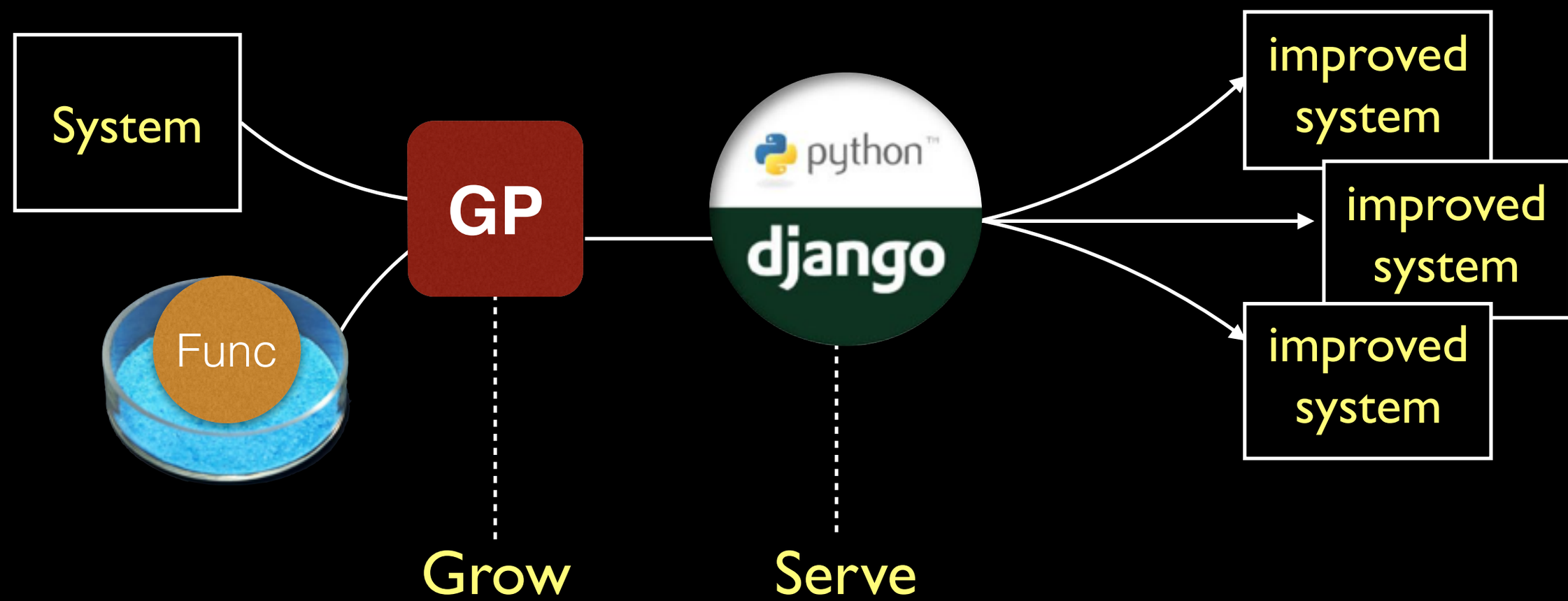
Grow code for new functionality, rather than to improve existing non-functional properties of the system.

GGGI Grow and Graft for Genetic Improvements



Grow code for new functionality, rather than to improve existing non-functional properties of the system.

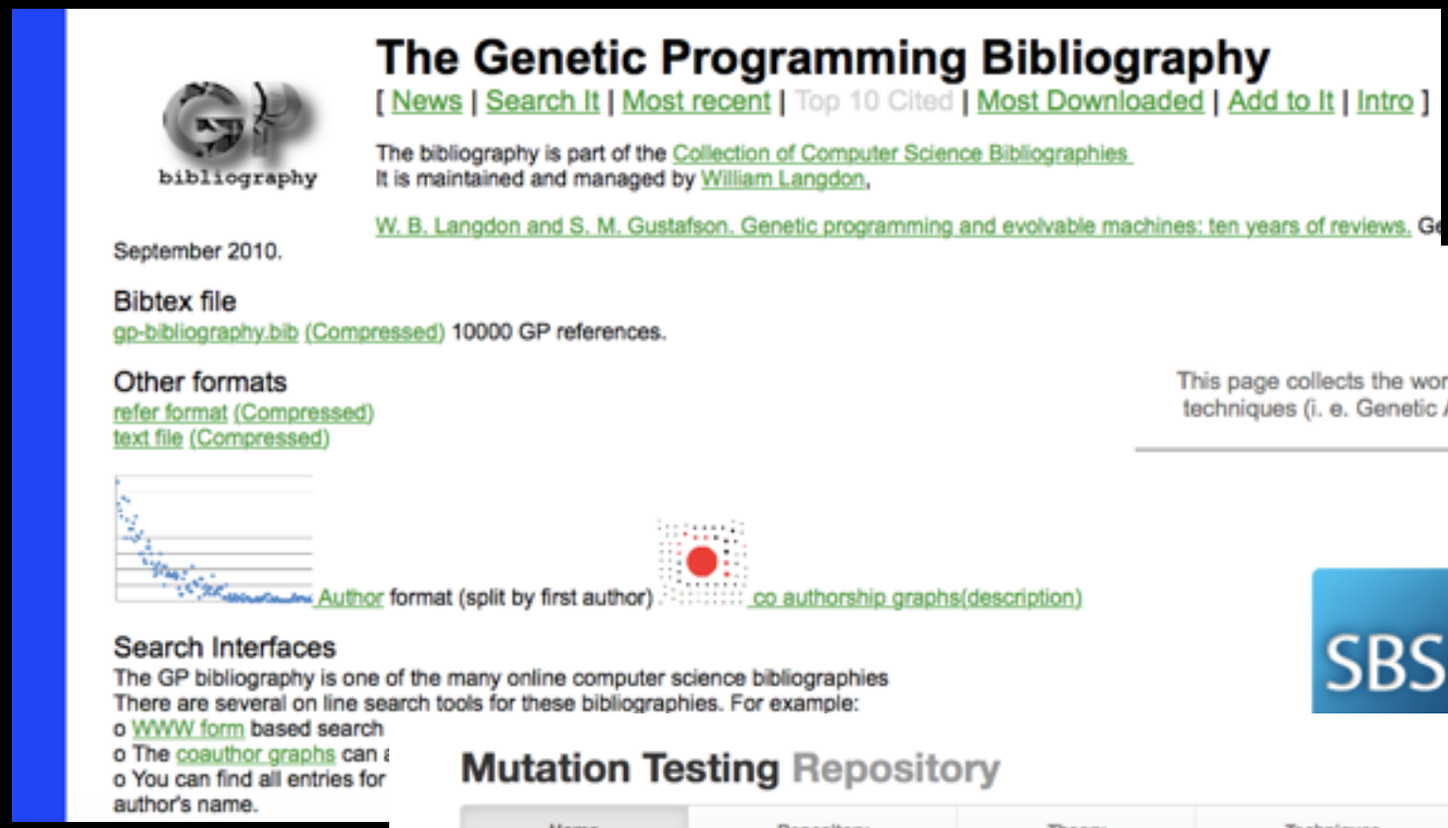
GSGI Grow and Serve for Genetic Improvements

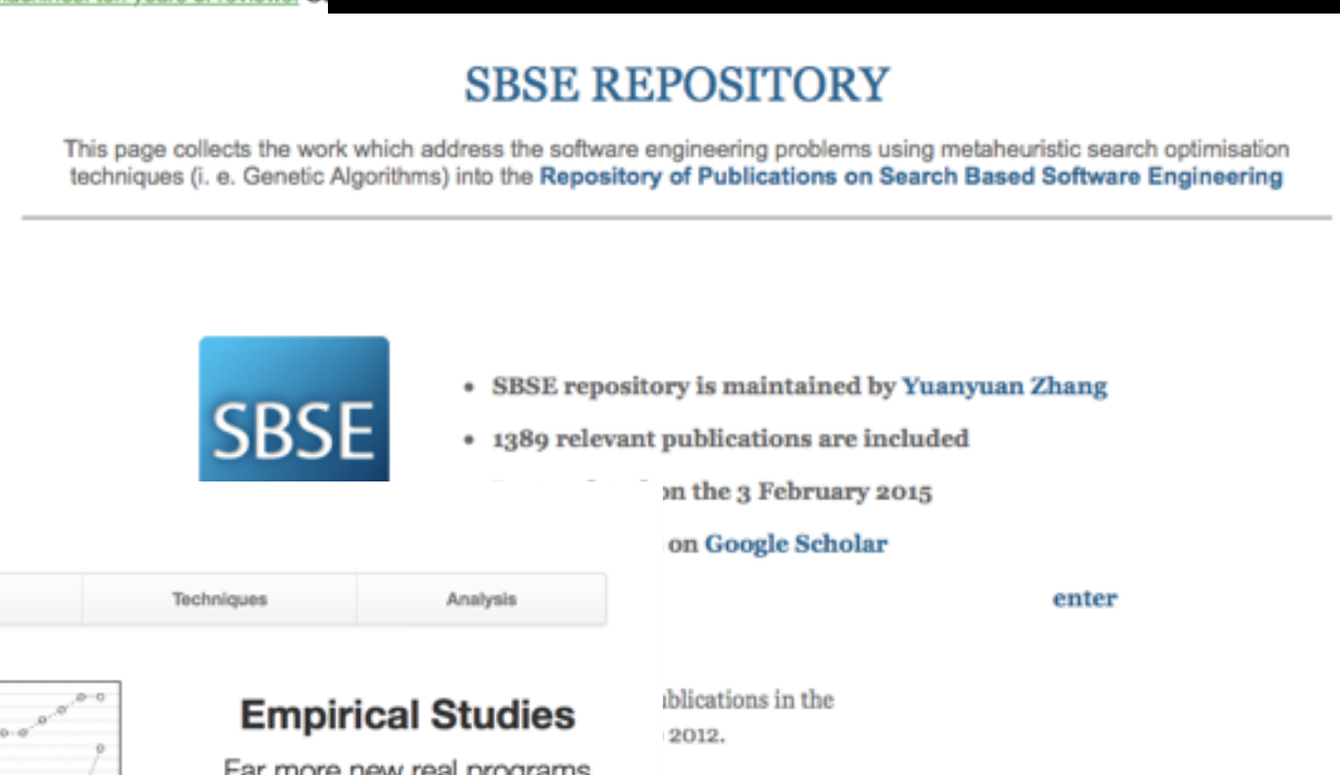


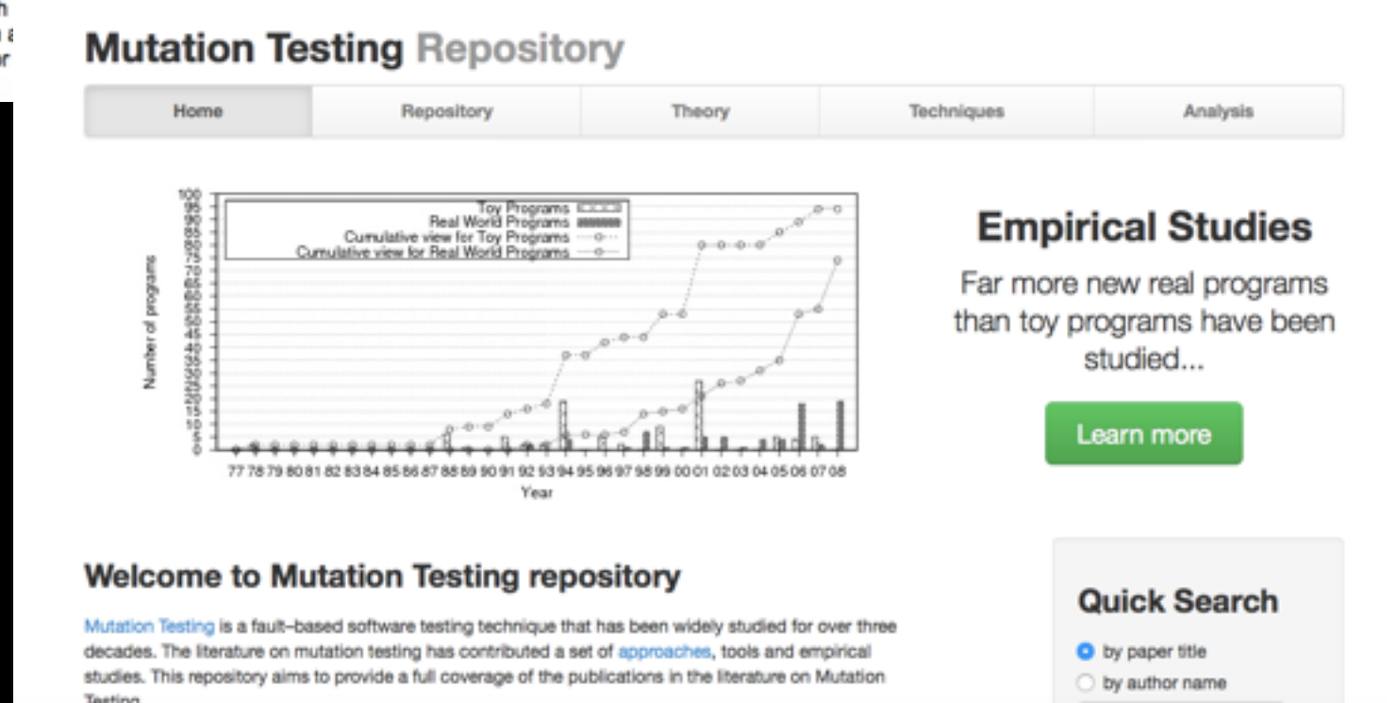
Grow code for new functionality and deploy on service-based architecture


Eating your own dog food
http://www.flickr.com/photos/booleansplit/2447000009/

Why citation services?







Why citation services?

Evolving Human Competitive Spectra-Based Fault Localisation Techniques

Created by W.Langdon from gp-bibliography.bib Revision:1.2990

```
@InProceedings{Yoo:2012:SSBSE,
  author =        "Shin Yoo",
  title =         "Evolving Human Competitive Spectra-Based Fault
                  Localisation Techniques",
  booktitle =     "4th Symposium on Search Based Software Engineering",
  year =          "2012",
  editor =        "Gordon Fraser and Jerffeson {Teixeira de Souza} and
                  Angelo Susi",
  volume =        "7515",
  series =        "Lecture Notes in Computer Science",
  pages =         "244--258",
  address =       "Riva del Garda, Italy",
  month =         sep # " 28-30",
  publisher =     "Springer",
  keywords =      "genetic algorithms, genetic programming, SBSE",
  isbn13 =        "978-3-642-33118-3",
  URL =           "http://www.cs.ucl.ac.uk/staff/s.yoo/papers/
  doi =           "doi:10.1007/978-3-642-33119-0_18",
  size =          "15 pages",
  abstract =      "Spectra-Based Fault Localisation (SBFL) aims to a
                  debugging by applying risk evaluation formulas
                  (sometimes called suspiciousness metrics) to progr
                  spectra and ranking statements according to the
                  predicted risk. Designing a risk evaluation formul
                  often an intuitive process done by human software
```

Mutation Testing Publications

Search results for "**JiaH08b**", found 1 papers, 1 pages

Papers per page 10 ← Prev Current Page 1 Next →

1 Yue Jia and Mark Harman
Constructing Subtle Faults Using Higher Order Mutation Testing cited by 90
Proceedings of the 8th International Working Conference on Source Code Analysis and Manipulation (SCAM'08)Beijing, China, 28-29 September 2008.

BibTeX | Abstract | URL

Abstract:
Traditional mutation testing considers only first order mutants, created by the injection of a single fault. Often these first order mutants denote trivial faults that are easily killed. This paper investigates higher order mutants (HOMs). It introduces the concept of a subsuming HOM; one that is harder to kill than the first order mutants from which it is constructed. By definition, subsuming HOMs denote subtle fault combinations. The paper reports the results of an empirical study into subsuming HOMs, using six benchmark programs. This is the largest study of mutation

Why citation services?

Google

evolving human competitive spectra-based fault localisation technique

Scholar

Articles
Case law
My library

Any time
Since 2015
Since 2014
Since 2011
Custom range...

Evolving human competitive spectra-based fault localisation techniques
S Yoo - Search Based Software Engineering, 2012 - Springer
Abstract Spectra-Based Fault L
evaluation formulæ (sometimes
ranking statements according t
Cited by 20 Related articles

Showing the best result fo

About G

Google

construct subtle faults using higher order mutation testing

Scholar

About 21,900 results (0.02 sec)

Articles
Case law
My library

Any time
Since 2015
Since 2014
Since 2011
Custom range...

Constructing subtle faults using higher order mutation testing
Y Jia, M Harman - Source Code Analysis and Manipulation, …, 2008 - ieeexplore.ieee.org
Abstract Traditional **mutation testing** considers only first **order mutants**, created by the injection of a single **fault**. Often these first **order mutants** denote trivial **faults** that are easily killed. This paper investigates **Higher Order Mutants** (HOMs). It introduces the concept of a ...
Cited by 90 Related articles All 18 versions Cite Save

Higher order mutation testing
Y Jia, M Harman - Information and Software Technology, 2009 - Elsevier
... coupled and decoupled HOMs may turn out to be harder to kill than the FOMs from which th
are **constructed**, making them ... Increased **subtlety**. ... However, by their very nature, the sub
HOMs we study in this paper are **more subtle**; they denote **faults** that **more** elaborate ...
Cited by 114 Related articles All 16 versions Cite Save More

Grow and Serve

Python APIs

Google Scholar

Test data

Citation Services

Strongly Typed GP

```
gp_function(title):
        ...
        ...
        ...
        ...
        ...
        ...
        ...
        return citations
```

GP Strongly Typed GP

```
gp_function(title):
        ...
        ...
        ...
        ...
        ...
        ...
        ...
        return citation_nubmer
```

APIs:

HTTP Request

XML Parsing

String Manipulation

List Manipulation

GP Strongly Typed GP

```
gp_function(title):
        ...
        ...
        ...
        ...
        ...
        ...
        ...
        return citation_nubmer
```

Google Scholar

"http://localhost/google_scholar/scholar?q="

"Cited by", "Cite"

GP Strongly Typed GP

Fitness function

Essential	Necessary
Inclusion	Ordering

GP Strongly Typed GP

Fitness function —— Essential

1	Characristic	Input	Expected Output
1	Full Title	'Higher Order Mutation Testing'	return 'Cited by 102'
2	Key Words	'Babel Pidgin'	return 'Cited by 5'
3	1 Citation	'Genetic Improvement for Adaptive Software Engineering'	return 'Cited by 1'
4	0 Citation	'Achievements, open problems and challenges for search based software testing'	return 'No Citation'
5	Bad Title	'sdfsdsdf sdoi jsdlkfjsdljlksdlkadslkfsadjlsdfkljsdflksd'	return 'No Citation'

Table 1. The 5 Functional Black Box Test Cased Used for Essential Fitness

GP Strongly Typed GP

Fitness function —— Inclusion

APIs:

to send html request

to parse xml

to search string

to concatenate string

to filter list

GP Strongly Typed GP

Fitness function		Ordering
concatenate string	→	send html request
to parse xml	→	to search xml
to search xml	→	to filter list

GP Strongly Typed GP

Fitness function —— Necessary

- Generate correct link
- Correct call to google scholar
- Result contains citation data

RQ: Human-machine tradeoff

Fitness function

We experimented with 8 different fitness functions, composed of subsets of 17 equally-weighted fitness components.

We categorised our fitness functions into four distinct categories: (E)ssential, (N)ecessary, (I)nclusion and (O)rdering

Results

(E)ssential, (N)ecessary, (I)nclusion and (O)rdering

Fitness Used	Successful runs in 30 trials	Time in seconds	f=Fitness evaluations
E	0 (p=N/A)	204	6,306
EI	0 (p=N/A)	281	7,400
EO	0 (p=N/A)	379	10,226
EN	0 (p=N/A)	348	9,686
EIO	1 (p=0.500)	425	10,806
ENI	0 (p=N/A)	438	11,133
ENO	9 **(p=0.020)**	443	11,633
ENIO	16 **(p=0.002)**	499	12,700

Results

(E)ssential, (N)ecessary, (I)nclusion and (O)rdering

Fitness Used	Successful runs in 30 trials	Time in seconds	f=Fitness evaluations
E	0 (p=N/A)	204	6,306
EI	0 (p=N/A)	281	7,400
EO	0 (p=N/A)	379	10,226
EN	0 (p=N/A)	348	9,686
EIO	1 (p=0.500)	425	10,806
ENI	0 (p=N/A)	438	11,133
ENO	9 **(p=0.020)**	443	11,633
ENIO	16 **(p=0.002)**	499	12,700

Results

(E)ssential, (N)ecessary, (I)nclusion and (O)rdering

Fitness Used	Successful runs in 30 trials	Time in seconds	f=Fitness evaluations
E	0 (p=N/A)	204	6,306
EI	0 (p=N/A)	281	7,400
EO	0 (p=N/A)	379	10,226
EN	0 (p=N/A)	348	9,686
EIO	1 (p=0.500)	425	10,806
ENI	0 (p=N/A)	438	11,133
ENO	9 **(p=0.020)**	443	11,633
ENIO	16 **(p=0.002)**	499	12,700

Results

(E)ssential, (N)ecessary, (I)nclusion and (O)rdering

Fitness Used	Successful runs in 30 trials	Time in seconds	f=Fitness evaluations
E	0 (p=N/A)	204	6,306
EI	0 (p=N/A)	281	7,400
EO	0 (p=N/A)	379	10,226
EN	0 (p=N/A)	348	9,686
EIO	1 (p=0.500)	425	10,806
ENI	0 (p=N/A)	438	11,133
ENO	9 **(p=0.020)**	443	11,633
ENIO	16 **(p=0.002)**	499	12,700

Results

```
def gp_function(title):
    try:
    var_return = '' # String
    var_0 = '' # String
    var_1 = None # etree
    var_2 = list() # List
    var_3 = False # Bool
    var_4 = None # Response
    var_5 = 0 # Int
    title = concat (‘http://localhost/google_scholar/scholar?q=', title )
    var_4 = requests_get (title )
    var_return = response_get_text (var_4 )
    var_1 = get_tree_from_html_string (var_return )
    var_2 = tree_get_xpath ('//a/text()', var_1 )
    var_5 = len_string (var_0 )
    var_2 = filter_list (var_2, 'Cited by' )
    var_2 = filter_list (var_2, var_0 )
    title = get_string_value_from_list (var_2, 1 )
    var_3 = string_starts_with ('Cite', var_0 )
    var_return = get_string_value_from_list (var_2, 1 )
    var_return = get_string_value_from_list (var_2, var_5 )
    return var_return
    except:
    return 'error'
```

Deployment

<img src="http://yuejia.cloudapp.net/gpcitation/img/ ** title ** /">

Deployment

http://yuejia.cloudapp.net/gpcitation/img/**title**/

http://yuejia.cloudapp.net/gpcitation/**title**/

Deployment

doi>

```
doi =          "      doi:10.1007/978-3-642-33119-0_18",
size =         "15 pages",
abstract =     "Spectra-Based Fault Localisation (SBFL) aims to assist
               debugging by applying risk evaluation formulas
               (sometimes called suspiciousness metrics) to program
               spectra and ranking statements according to the
               predicted risk. Designing a risk evaluation formula is
               often an intuitive process done by human software
               engineer. This pape
               (GP) approach for e
               The empirical evalu
               utilities produces
               by Genetic Programm
               of the human-desig
               Ochiai, Jaccard, An
               More importantly, t
               Op2, which was rece
               If-Then-Else-2 (ITE
               against other progr
notes =        "Entered 2012 HUMIE
               http://www.genetic-

               See also \cite{rn-1
               http://selab.fbk.eu
               http://www.cs.ucl.a
}
```

Genetic Programming entries for Shin Yoo

Citations by GP cited by 20

Mutation Testing Publications

Search results for "**JiaH08b**", found 1 papers, 1 pages

Papers per page 10 ← Prev Current Page 1 Next →

1 Yue Jia and Mark Harman
Constructing Subtle Faults Using Higher Order Mutation Testing cited by 90
Proceedings of the 8th International Working Conference on Source Code Analysis and Manipulation (SCAM'08)Beijing, China, 28-29 September 2008.

BibTeX | Abstract | URL

Abstract:
Traditional mutation testing considers only first order mutants, created by the injection of a single fault. Often these first order mutants denote trivial faults that are easily killed. This paper investigates higher order mutants (HOMs). It introduces the concept of a subsuming HOM; one that is harder to kill than the first order mutants from which it is constructed. By definition, subsuming HOMs denote subtle fault combinations. The paper reports the results of an empirical study into subsuming HOMs, using six benchmark programs. This is the largest study of mutation

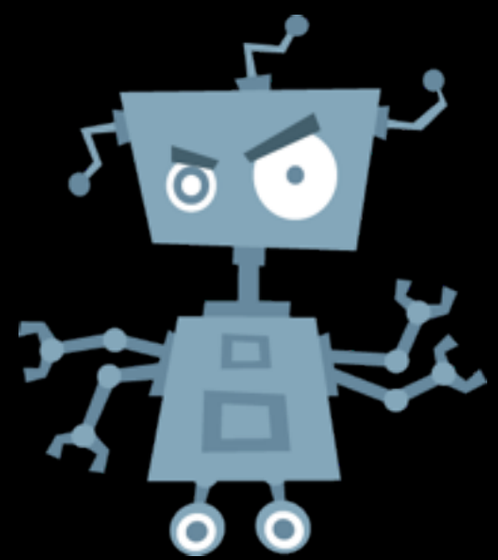
Deployment

In the first 24 hours, 369 requests from 29 countries.

Titles containing special character: & \

Papers with similar titles

Deployment



Googlebot-Image/1.0
Baiduspider/2.0
bingbot/2.0

Everyday on average receives more than 60 requests

Conclusion

We introduce a ‘grow and serve’ approach to Genetic Improvement (GI)

We report on the trade offs between varying degrees of human guidance to the GSGI process

http://yuejia.cloudapp.net/gpcitation/img/**title**/

http://yuejia.cloudapp.net/gpcitation/**title**/