GETTING IT RIGHT: SYSTEM TESTING OF SCIENTIFIC SOFTWARE

Myra Cohen

https://www.cs.iastate.edu/~mcohen

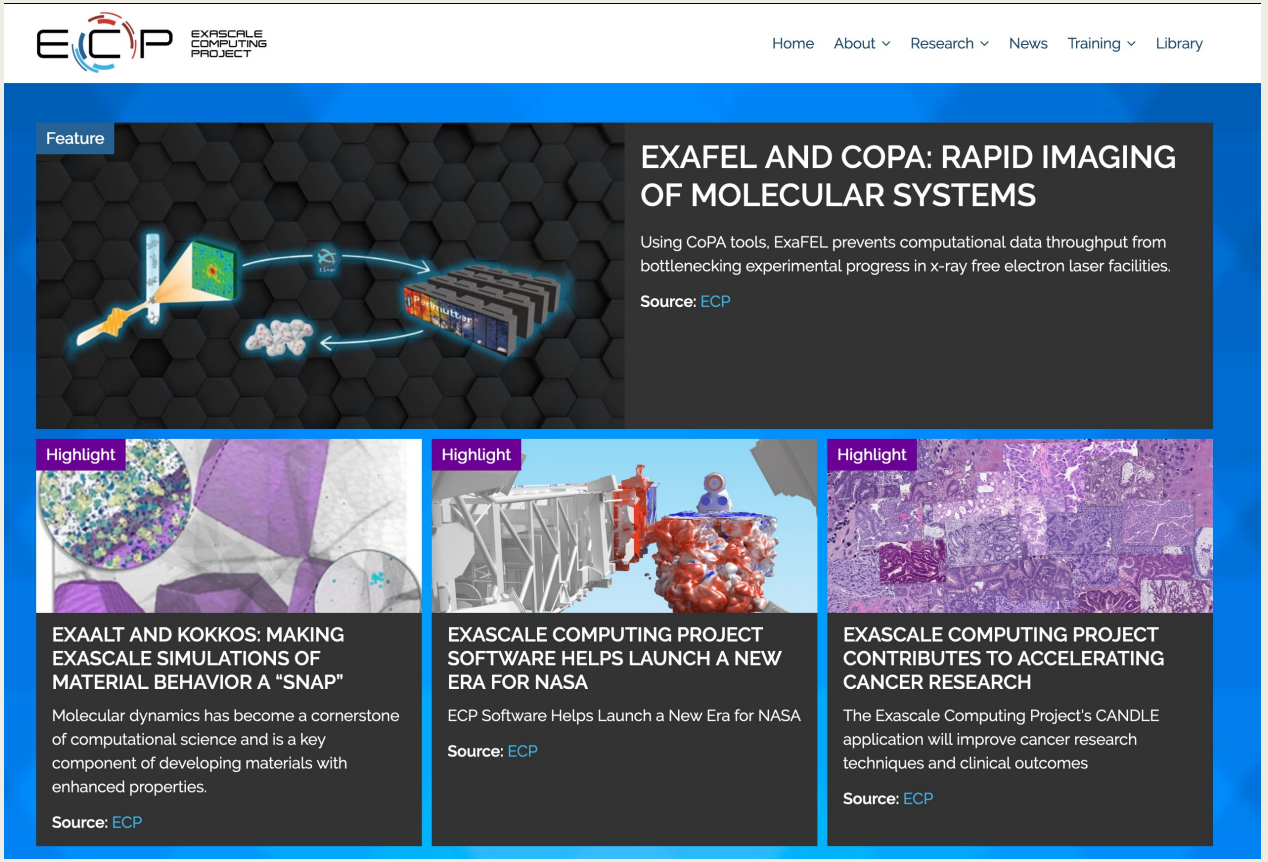
mcohen@iastate.edu

IOWA STATE UNIVERSITY
OF SCIENCE AND TECHNOLOGY

better scientific software

Laboratory for Variability-Aware Assurance and Testing of Organic Programs
LaVA-OPs

Scientific Software is Everywhere



Yet it sometimes Fails..

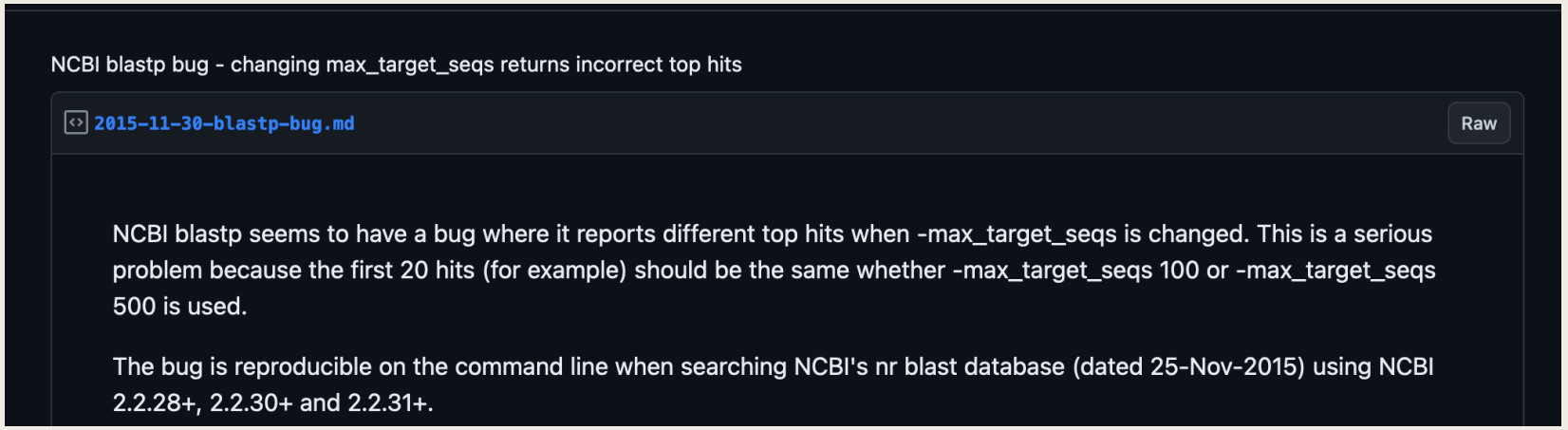
```
Error: NCBI C++ Exception:
T0 "/tmp/BLAST/2.11.0/gompi-2020b/ncbi-blast-2.11.0+-src/c++/src/serial/objistrasnb.cpp
T0 "/tmp/BLAST/2.11.0/gompi-2020b/ncbi-blast-2.11.0+-src/c++/src/serial/member.cpp", l
```

Yet it sometimes Fails..

```
Error: NCBI C++ Exception:
T0 "/tmp/BLAST/2.11.0/gompi-2020b/ncbi-blast-2.11.0+-src/c++/src/serial/objistrasnb.cp
T0 "/tmp/BLAST/2.11.0/gompi-2020b/ncbi-blast-2.11.0+-src/c++/src/serial/member.cpp", l
```

```
ial/objistrasnb.cpp", line 499: Error: (CSerialException::eOverflow) byte 132: overf
ial/member.cpp", line 767: Error: (CSerialException::eOverflow) ncbi::CMemberInfoFur
```

Yet it sometimes Fails..



Yet it sometimes Fails..

NCBI blastp bug - changing max_target_seqs returns incorrect top hits

2015-11-30-blastp-bug.md

Raw

NCBI blas… serious
problem l… et_seqs
500 is us…

The bug … g NCBI
2.2.28+, …

Bioinformatics, 35(9), 2019, 1613–1614
doi: 10.1093/bioinformatics/bty833
Advance Access Publication Date: 24 September 2018
Letter to the Editor

OXFORD

Sequence analysis

Misunderstood parameter of NCBI BLAST impacts the correctness of bioinformatics workflows

Nidhi Shah[1], Michael G. Nute[2], Tandy Warnow [3] and Mihai Pop [1,*]

[1]Department of Computer Science, University of Maryland College Park, MD 20742, USA, [2]Department of Statistics and [3]Department of Computer Science, University of Illinois Urbana-Champaign, Champaign, IL 61820, USA

*To whom correspondence should be addressed.
Associate Editor: John Hancock
Contact: mpop@umd.edu
Received and revised on August 13, 2018; editorial decision on September 19, 2018; accepted on September 21, 2018

Sometimes it seems to fail..

Bioinformatics, 35(15), 2019, 2699–2700
doi: 10.1093/bioinformatics/bty1026
Advance Access Publication Date: 24 December 2018
Letter to the Editor

OXFORD

Sequence analysis

Reply to the paper: Misunderstood parameters of NCBI BLAST impacts the correctness of bioinformatics workflows

Thomas L. Madden*, Ben Busby and Jian Ye

National Center for Biotechnology Information, National Library of Medicine, National Institutes of Health, Bethesda, MD 20894, USA

*To whom correspondence should be addressed.

Associate Editor: John Hancock

Contact: madden@ncbi.nlm.nih.gov

Received and revised on November 30, 2018; editorial decision on December 10, 2018; accepted on December 19, 2018

Sometimes it seems to fail..

Dear Editor,

A recent letter by Shah *et al.* (2018) addressed the use of a command-line parameter in BLAST (Altschul *et al.*, 1997; Camacho *et al.*, 2009). BLAST is a very popular tool, so it is not surprising that this topic has provoked a great deal of interest. The authors have, however, conflated three different issues. One is a bug that will be fixed in the BLAST+ 2.8.1 release due out in December 2018, another is simply how BLAST works and the third might be viewed as a shortcoming of our implementation of composition-based statistics (CBS). Here, we address these issues and describe some new documentation about the BLAST process.

Shah et al. (2018) did not provide their own example in the letter, but later provided one at https://github.com/shahnidhi/BLAST_maxtargetseq_analysis. At the NCBI, we examined the new example and it became clear that the demonstrated behavior was a bug, resulting from an overly aggressive optimization, introduced in 2012 for BLASTN and MegaBLAST (DNA–DNA alignments). This bug has been fixed in the BLAST+ 2.8.1 release, due out in December 2018. The aberrant behavior seems to occur only in alignments with an extremely large number of gaps, which is the case in the example provided by Shah and collaborators.

Cost of Poor Quality Software

- CISQ Consortium for Information & Software Quality 2020 report

  $2.08 trillion cost in United States

- 2002 National Institutes of Standards and Technology

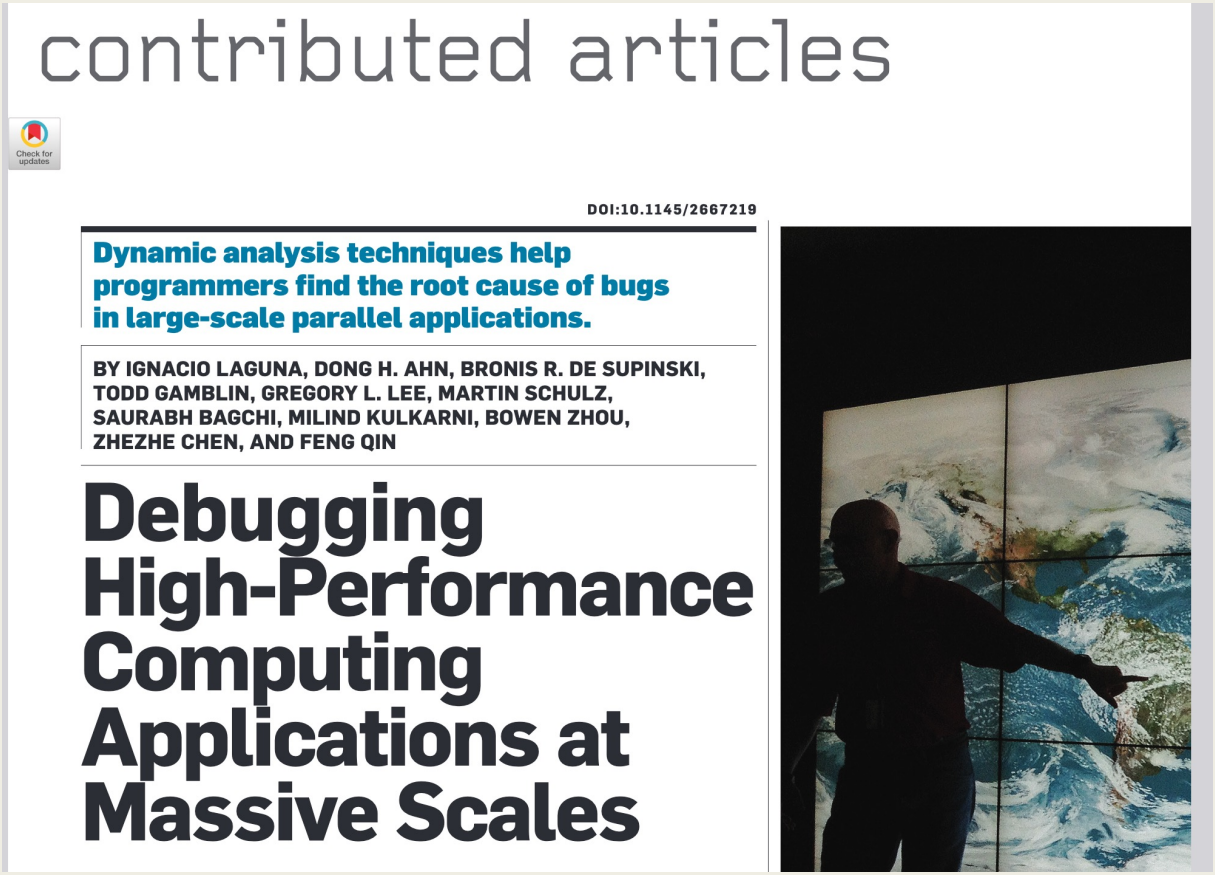
  Up to $59 Billion per year in United States

- Scientific Software

  ?

Why Test Software?

contributed articles



DOI:10.1145/2667219

Dynamic analysis techniques help programmers find the root cause of bugs in large-scale parallel applications.

BY IGNACIO LAGUNA, DONG H. AHN, BRONIS R. DE SUPINSKI, TODD GAMBLIN, GREGORY L. LEE, MARTIN SCHULZ, SAURABH BAGCHI, MILIND KULKARNI, BOWEN ZHOU, ZHEZHE CHEN, AND FENG QIN

Debugging High-Performance Computing Applications at Massive Scales

Why Test Software?

contributed articles

DOI:10.1145/3382037

An approach to reproducibility problems related to porting software across machines and compilers.

BY DONG H. AHN, ALLISON H. BAKER, MICHAEL BENTLEY, IAN BRIGGS, GANESH GOPALAKRISHNAN, DORIT M. HAMMERLING, IGNACIO LAGUNA, GREGORY L. LEE, DANIEL J. MILROY, AND MARIANA VERTENSTEIN

Keeping Science on Keel When Software Moves

the machine instructions that actually get executed. Unfortunately, such changes do affect the computed results to a significant (and often worrisome) extent. In a majority of cases, there are not easily definable a priori answers one can check against. A programmer ends up comparing the new answer against a trusted baseline previously established or checks for indirect confirmations such as whether physical properties such as energy are conserved. However, such non-systematic efforts might miss underlying issues, and the code may keep misbehaving until these are fixed.

In this article, we present real-world evidence to show that ignoring numerical result changes can lead to misleading scientific conclusions. We present techniques and tools that can help computational scientists understand and analyze compiler effects on their scientific code. These techniques are applicable across a wide range of examples to narrow down the root-causes to single files, functions within files, and even computational expressions that affect specific variables. The developer may then rewrite the code selectively and/or suppress the application of certain optimizations to regain more predictable behavior.

Going forward, the frequency of required ports of computational software will increase, given that performance gains can no longer be obtained by mere-

In this article, we present real-world evidence to show that ignoring numerical result changes can lead to misleading scientific conclusions. We present tech-

Why Test Software?

When hackers leaked thousands of e-mails from the Climatic Research Unit (CRU) at the University of East Anglia in Norwich, UK, last year, global-warming sceptics pored over the documents for signs that researchers had manipulated data. No such evidence emerged, but the e-mails did reveal another problem — one described by a CRU employee named "Harry", who often wrote of his wrestling matches with wonky computer software.

"Yup, my awful programming strikes again," Harry lamented in one of his notes, as he attempted to correct a code analysing weather-station data from Mexico.

Merali, Zeeya. "Computational Science: ...Error." *Nature* 467, no. 7317 (Oct, 2010): 775–77

Overview

Types of Testing

Challenges

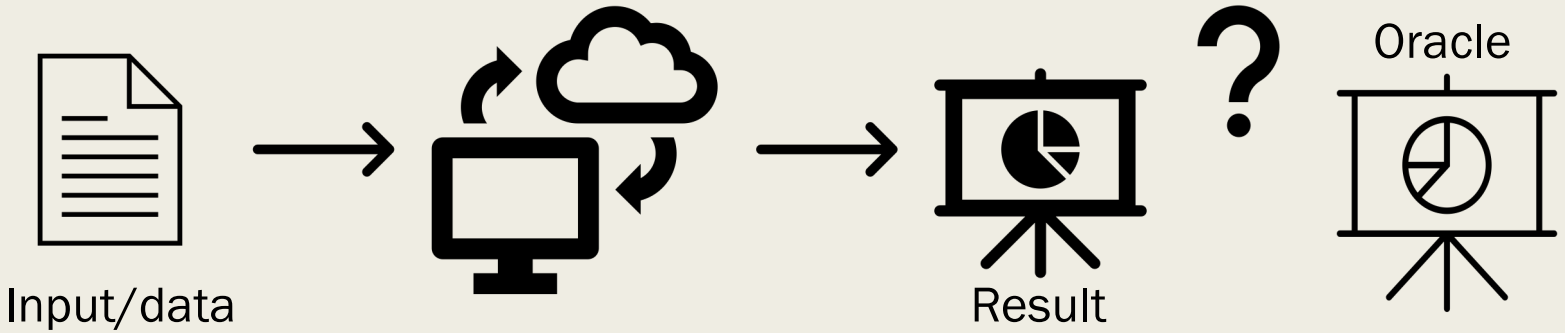
Models

Coverage

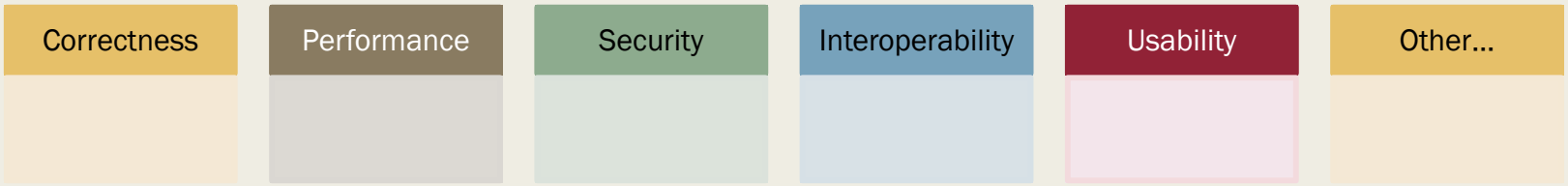
Oracles

Configurability

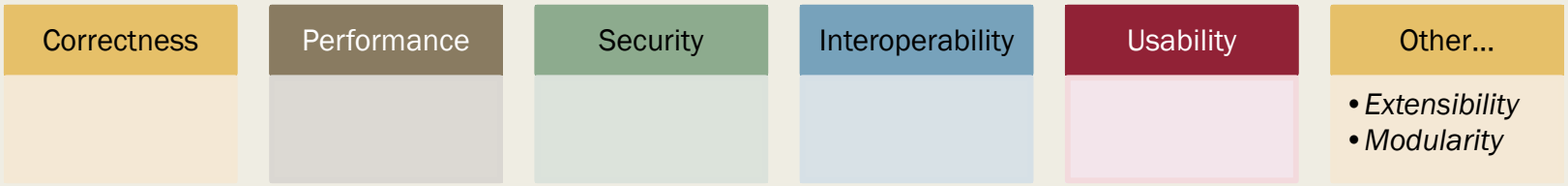
What is Testing



What Should We Test?



What Should We Test?

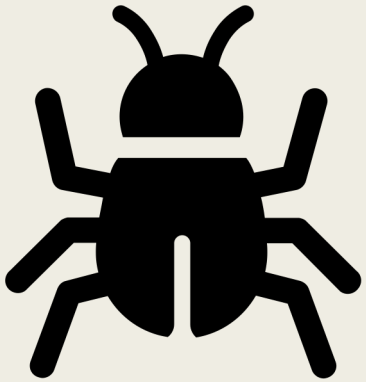


Unit tests are important, but there is more….

Limitations

Testing can only show the presence of faults.
It cannot determine their absence.

Edsger W. Dijkstra



Challenge 1

- To detect a program FAILURE we need to:
  - Reach a FAULT in the code
  - Infect the code (change to incorrect state) - ERROR
  - Propagate the error out of program
  - Reveal (detect) the error – (ORACLE)

RIPR model *Ammann, Offutt (Introduction to Software Testing, 2016)*

Challenge 2

- Covering code during testing, only tests the logic that is there!
  - *Also need to test from the system – does the software meet the specifications*

Tests Can Miss Faults

```python
def classify_triangle(a, b, c):
    # Sort the sides so that a <= b <= c
    if a > b:
        tmp = a
        a = tmp   #fault should be a=b
        b = tmp

    if a > c:
        tmp = a
        a = c
        c = tmp

    if b > c:
        tmp = b
        b = c
        c = tmp

    if a + b <= c:
        return TriangleType.INVALID
    elif a == b and b == c:
        return TriangleType.EQUILATERAL
    elif a == b or b == c:
        return TriangleType.ISOSCELES
    else:
        return TriangleType.SCALENE
```

Tests Can Miss Faults

```python
def classify_triangle(a, b, c):
    # Sort the sides so that a <= b <= c
    if a > b:
        tmp = a
        a = tmp   #fault should be a=b
        b = tmp

    if a > c:
        tmp = a
        a = c
        c = tmp

    if b > c:
        tmp = b
        b = c
        c = tmp

    if a + b <= c:
        return TriangleType.INVALID
    elif a == b and b == c:
        return TriangleType.EQUILATERAL
    elif a == b or b == c:
        return TriangleType.ISOSCELES
    else:
        return TriangleType.SCALENE
```

1. Test Case 3, 4, 5 (scalene)
   - doesn't reach fault

Tests Can Miss Faults

```python
def classify_triangle(a, b, c):
    # Sort the sides so that a <= b <= c
    if a > b:
        tmp = a
        a = tmp   #fault should be a=b
        b = tmp       5,5,1

    if a > c:
        tmp = a
        a = c
        c = tmp

    if b > c:
        tmp = b
        b = c
        c = tmp

    if a + b <= c:
        return TriangleType.INVALID
    elif a == b and b == c:
        return TriangleType.EQUILATERAL
    elif a == b or b == c:
        return TriangleType.ISOSCELES
    else:
        return TriangleType.SCALENE
```

1. Test Case 3, 4, 5 (scalene)
   - doesn't reach fault

2. Test Case 5, 1, 1 (invalid)
   - reaches fault and infects
   - reveals (returns isosceles)

Tests Can Miss Faults

```python
def classify_triangle(a, b, c):
    # Sort the sides so that a <= b <= c
    if a > b:
        tmp = a
        a = tmp   #fault should be a=b
        b = tmp       2,2,-1

    if a > c:
        tmp = a
        a = c
        c = tmp

    if b > c:
        tmp = b
        b = c
        c = tmp

    if a + b <= c:
        return TriangleType.INVALID
    elif a == b and b == c:
        return TriangleType.EQUILATERAL
    elif a == b or b == c:
        return TriangleType.ISOSCELES
    else:
        return TriangleType.SCALENE
```

1. Test Case 3, 4, 5 (scalene)
   - doesn't reach fault

2. Test Case 5, 1, 1 (invalid)
   - reaches fault and infects
   - reveals (returns isosceles)

3. Test Case 2, 1, -1 (invalid)
   - reaches fault and infects
   - Doesn't propagate (2, 2,-1) is still INVALID

kbase-all-hands-on

v1 - KBaseFBA.FBA-13.2

Overview | Reaction fluxes | Exchange fluxes | Genes | Biomass | Pathways | Bar charts

ID	myracohen:narrative_15
Object type	KBaseFBA.FBA-13.2
Owner	myracohen
Version	1
Mod-date	2024-02-26T06:15:44+0
Objective value	0.507834
Model	E-Coli_k12_MG1
Media	Carbon-D-Glucose
Single KO	0
Number reactions	857
Number compounds	29
Gene KO	0
Reaction KO	0
Custom bounds	0

?

kbase-all-hands-on-different-media

v2 - KBaseFBA.FBA-13.2

Overview | Reaction fluxes | Exchange fluxes | Genes | Biomass | Pathways | Bar charts

ID	myracohen:narrative_1501886992034/kbase-all-hands-on-different-media
Object type	KBaseFBA.FBA-13.2
Owner	myracohen
	2
	2024-02-26T06:36:45+0000
	13.4555
Model	E-Coli_k12_MG1
Media	Complete
Single KO	0
Number reactions	857
Number compounds	29
Gene KO	0
Reaction KO	0
Custom bounds	0

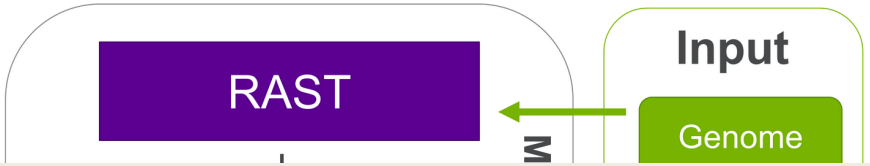
The Future?

, or alternatively, Machine learning (ML) classifiers will select the templates for model reconstruction

Build and gap-fill genome-scale metabolic models with ModelSEED v2 (MS2)

ModelSEED 2 genome-scale metabolic reconstruction pipeline enabling quantitative prediction of ATP production

A genome annotated with RAST is provided as input. Users can select a template for reconstruction, or alternatively, Machine learning (ML) classifiers will select the templates for model reconstruction. ATP production is tested across 54 media formulations representing many diverse energy biosynthesis strategies, with gap-filling performed as necessary to ensure ATP is produced in at least one condition. Next, we expand the core metabolism model to the genome-scale. The genome-scale model's gap-filling is then performed using our default auxotrophic medium or an optional user-specified custom media formulation.



Overview

Types of Testing

Challenges

Models

Coverage

Oracles

Configurability

Models

Provide an abstraction of the software we are testing

Can be for different dimensions of the software (specifications, interface, code)

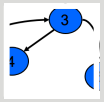
Allow us to reason about how much we have tested

The foundation for automated test generation

Example Models

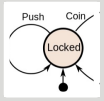
- Graphs
- Tabular
- Relational
- Grammar based
- Logic based

Graph Models

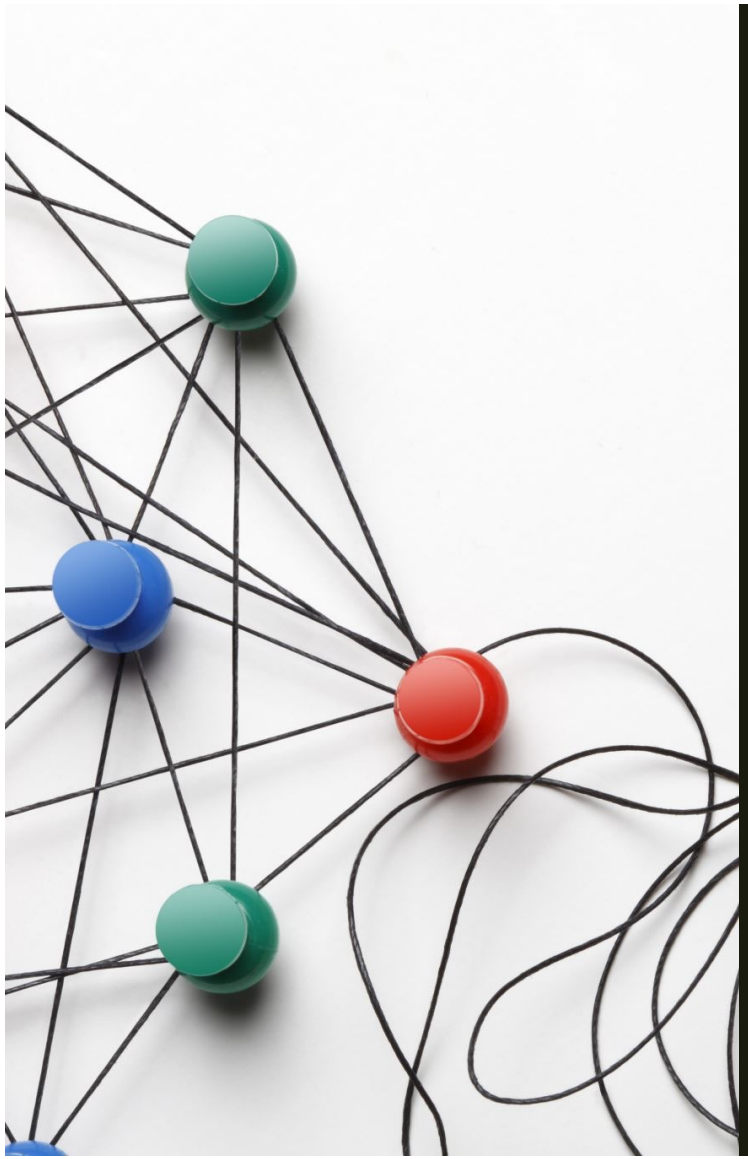


Program control flow graph

User interface



Program state machine

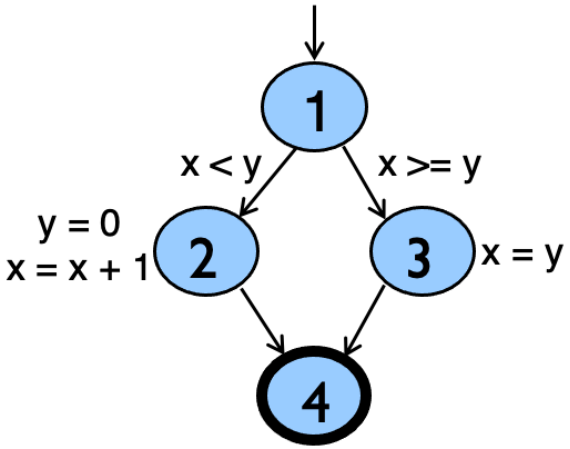


Types of Graph Coverage

- All nodes
- All edges (pairs of nodes)
- All length N paths
- M random length N paths

Program Code Coverage

```
if (x < y)
{
   y = 0;
   x = x + 1;
}
else
{
   x = y;
}
```



Control flow graph

Program Code Coverage

Cog coverage: 38.75%

coverage.py v7.2.7, created at 2023-05-29 15:26 -0400

Module	*statements*	*missing*	*excluded*	*branches*	*partial*	*coverage*
cogapp/__init__.py	1	0	0	0	0	100.00%
cogapp/__main__.py	3	3	0	0	0	0.00%
cogapp/cogapp.py	500	224	1	210	30	49.01%
cogapp/makefiles.py	22	18	0	14	0	11.11%
cogapp/test_cogapp.py	845	591	2	24	1	29.57%
cogapp/test_makefiles.py	70	53	0	6	0	22.37%
cogapp/test_whiteutils.py	68	50	0	0	0	26.47%
cogapp/whiteutils.py	43	5	0	34	4	88.31%
Total	**1552**	**944**	**3**	**288**	**35**	**38.75%**

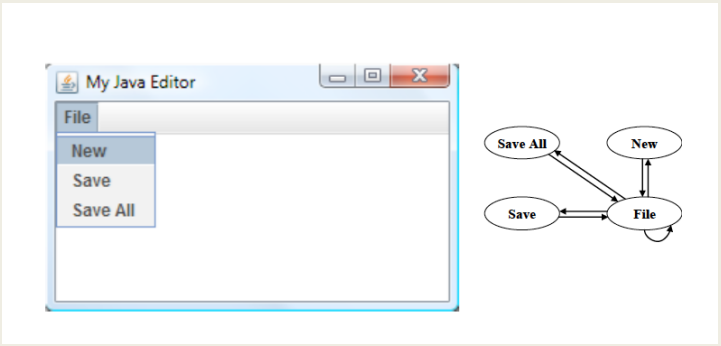
coverage.py v7.2.7, created at 2023-05-29 15:26 -0400

Triangle.java

```
1.  public class Triangle {
2.
3.      public enum TriangleType {
4.          INVALID, SCALENE, EQUILATERAL, ISOSCELES
5.      }
6.
7.      public static TriangleType classifyTriangle(int a, int b, int c) {
8.          if (a > b) {
9.              int tmp = a;
10.             a = b;
11.             b = tmp;
12.         }
13.
14.         if (a > c) {
15.         int tmp = c; // original: int tmp = a;
16.             a = c;
17.             c = tmp;
18.         }
19.
20.         if (b > c) {
21.             int tmp = b;
22.             b = c;
23.             c = tmp;
24.         }
25.
26.         if (a + b <= c) {
27.             return TriangleType.INVALID;
28.         } else if (a == b && b == c) {
29.             return TriangleType.EQUILATERAL;
30.         } else if (a == b || b == c) {
31.             return TriangleType.ISOSCELES;
32.         } else {
33.             return TriangleType.SCALENE;
34.         }
35.
36.     }
37. }
```

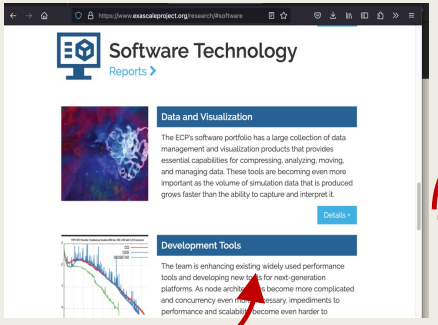
Example tools: jacoco, coverage.py, gcov

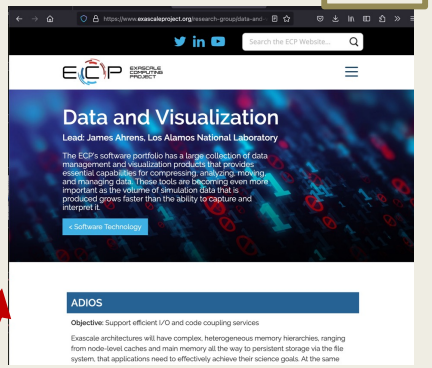
Interface (graph) Coverage

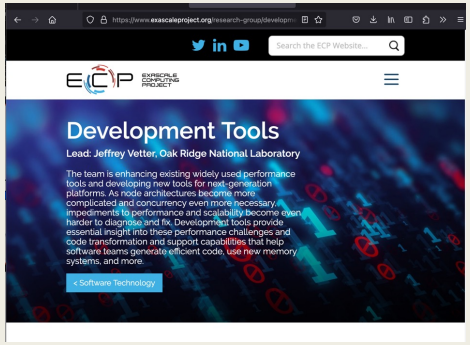


GUI

Web







Other Coverage

Specification coverage

- Cover the system requirements

Interaction coverage

- Measure interactions between components
  - Pairs, n-way coverage

Overview

Types of Testing

Challenges

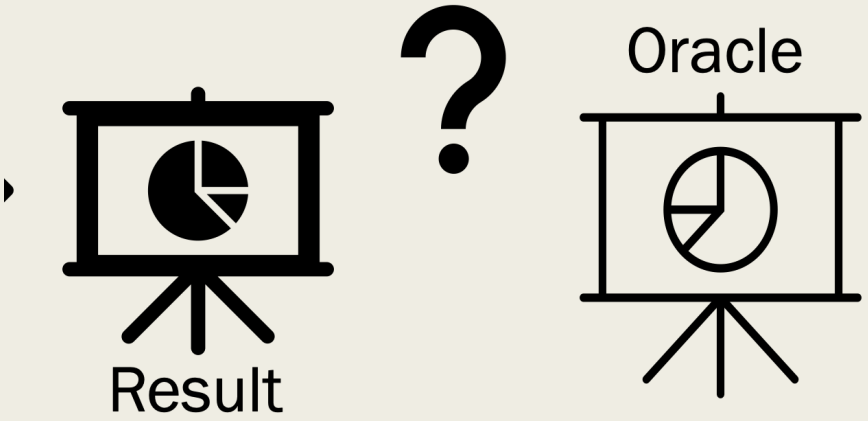
Models

Coverage

Oracles

Configurability

What is the Correct Answer?



Trivial Oracles

- Program crashes
- Core dump
- Segmentation error
- Overflow
- Program hangs

Trivial Oracles

- Good when we don't have a known result
- Weakest oracle since it only shows that the program fails/not that the result is incorrect
- Exact oracles are easy to compute in some programs

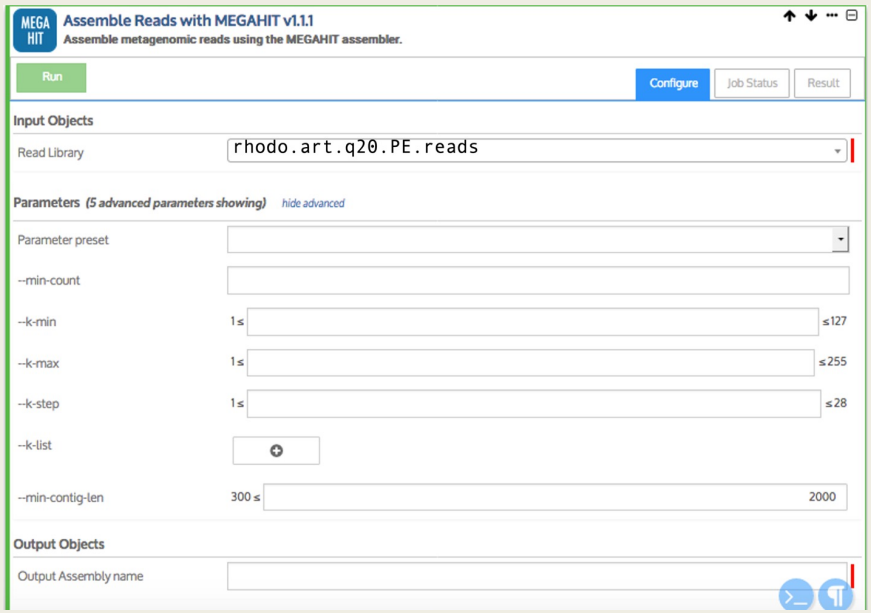
```
def classify_triangle(a, b, c):
    # Sort the sides so that a <= b <= c
    if a > b:
        tmp = a
        a = tmp   #fault should be a=b
        b = tmp

    if a > c:
        tmp = a
        a = c
        c = tmp

    if b > c:
        tmp = b
        b = c
        c = tmp

    if a + b <= c:
        return TriangleType.INVALID
    elif a == b and b == c:
        return TriangleType.EQUILATERAL
    elif a == b or b == c:
        return TriangleType.ISOSCELES
    else:
        return TriangleType.SCALENE
```

Harder Oracles



Making Oracles Hard

- Results may differ by small epsilons (due to rounding)
- Expected result may not be computable without program
- May have time series results
- Takes a long time to manually compute each oracle (even when we can)
- Programs may be stochastic (or non-determinstic)

Examples

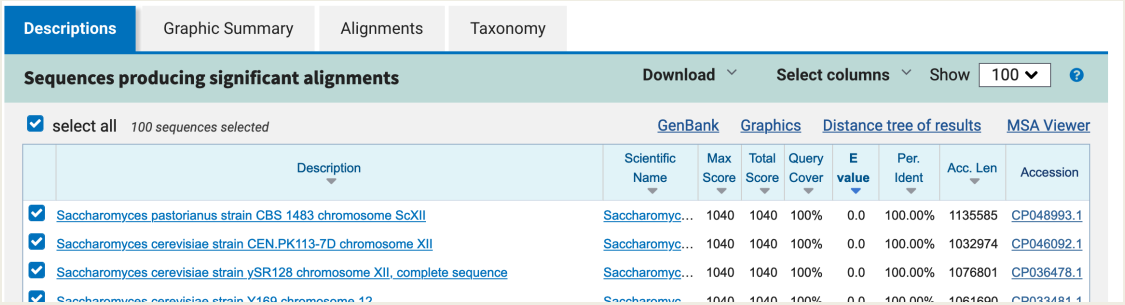
Python docs

Note: The behavior of `round()` for floats can be surprising: for example, `round(2.675, 2)` gives `2.67` instead of the expected `2.68`. This is not a bug: it's a result of the fact that most decimal fractions can't be represented exactly as a float. See Floating Point Arithmetic: Issues and Limitations for more information.

Same growth values?

Expected:	0.3569512**4**
Observed:	0.3569512**2**

Correct hits?



Some Techniques

Differential testing

Metamorphic testing

Differential Testing

Differential testing

Run same tests using different programs that have the same functionality

- run tests with BLAST
- run tests with HPC-BLAST

Challenge is determining equivalency

Metamorphic Testing

Metamorphic testing

Define relations on sets of tests:

e.g. (subtraction)

(1) A-B = C

Create A' (greater than A)

(2) A' -B = C' means C' is greater than C

Metamorphic Testing

Use Domain Knowledge
e.g. Ocean temperature modeling
A. Compute predicted temperature
B. Modify to increase expected temperature

Confirm relation holds

Metamorphic testing

kbase-all-hands-on
v1 - KBaseFBA.FBA-13.2

Overview | Reaction fluxes | Exchange fluxes | Genes | Biomass | Pathways | Bar charts

ID	myracohen:narrative_15
Object type	KBaseFBA.FBA-13.2
Owner	myracohen
Version	1
Mod-date	2024-02-26T06:15:44+0
Objective value	0.507834
Model	E-Coli_k12_MG1
Media	Carbon-D-Glucose
Single KO	0
Number reactions	857
Number compounds	29
Gene KO	0
Reaction KO	0
Custom bounds	0

kbase-all-hands-on-different-media
v2 - KBaseFBA.FBA-13.2

Overview | Reaction fluxes | Exchange fluxes | Genes | Biomass | Pathways | Bar charts

ID	myracohen:narrative_1501886992034/kbase-all-hands-on-different-media
Object type	KBaseFBA.FBA-13.2
Owner	myracohen
Version	2
Mod-date	2024-02-26T06:36:45+0000
Objective value	13.4555
Model	E-Coli_k12_MG1
Media	Complete
Single KO	0
Number reactions	857
Number compounds	29
Gene KO	0
Reaction KO	0
Custom bounds	0

Carbon-D-Glucose <= Complete

Overview

Types of Testing

Challenges

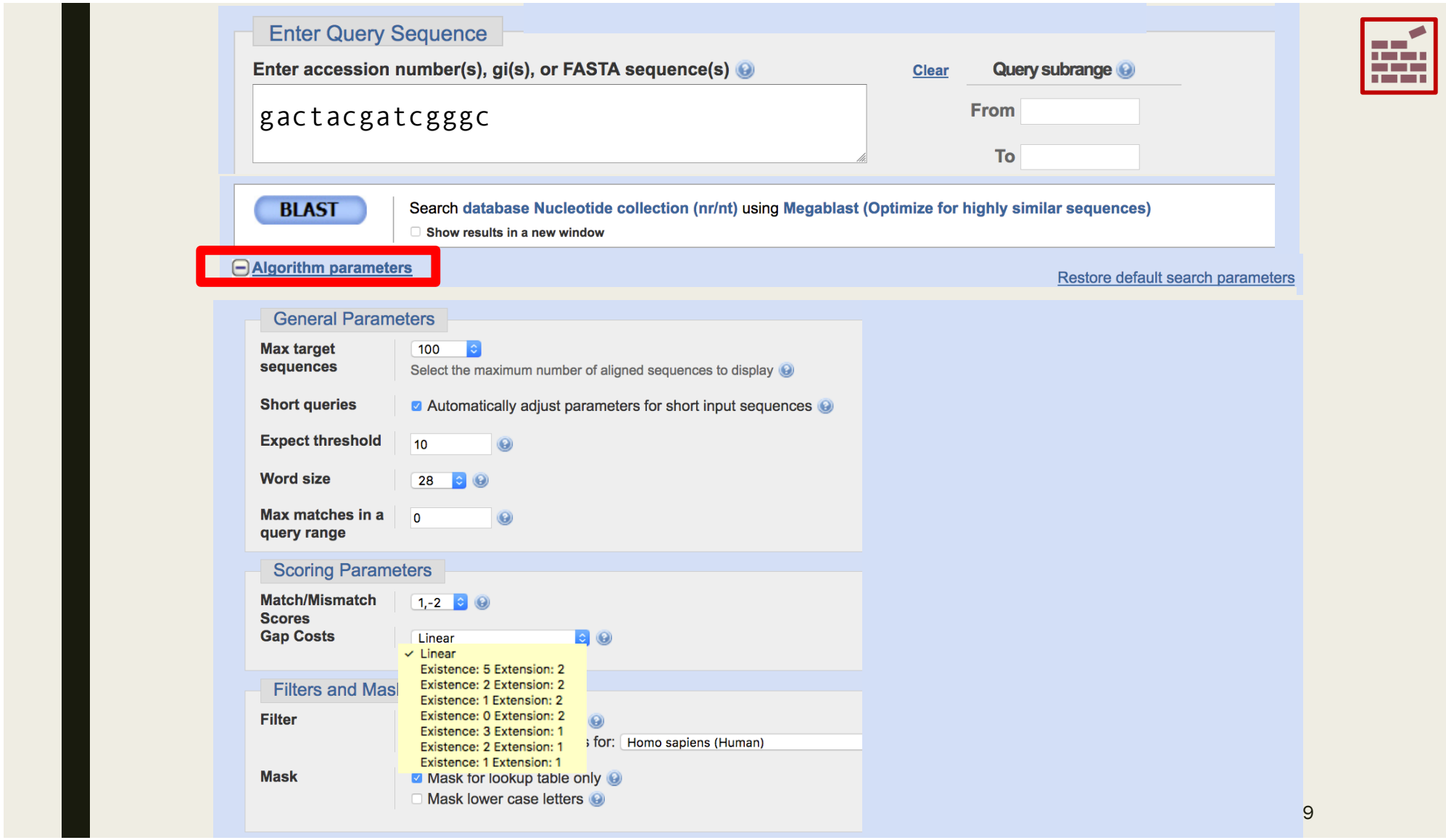
Models

Coverage

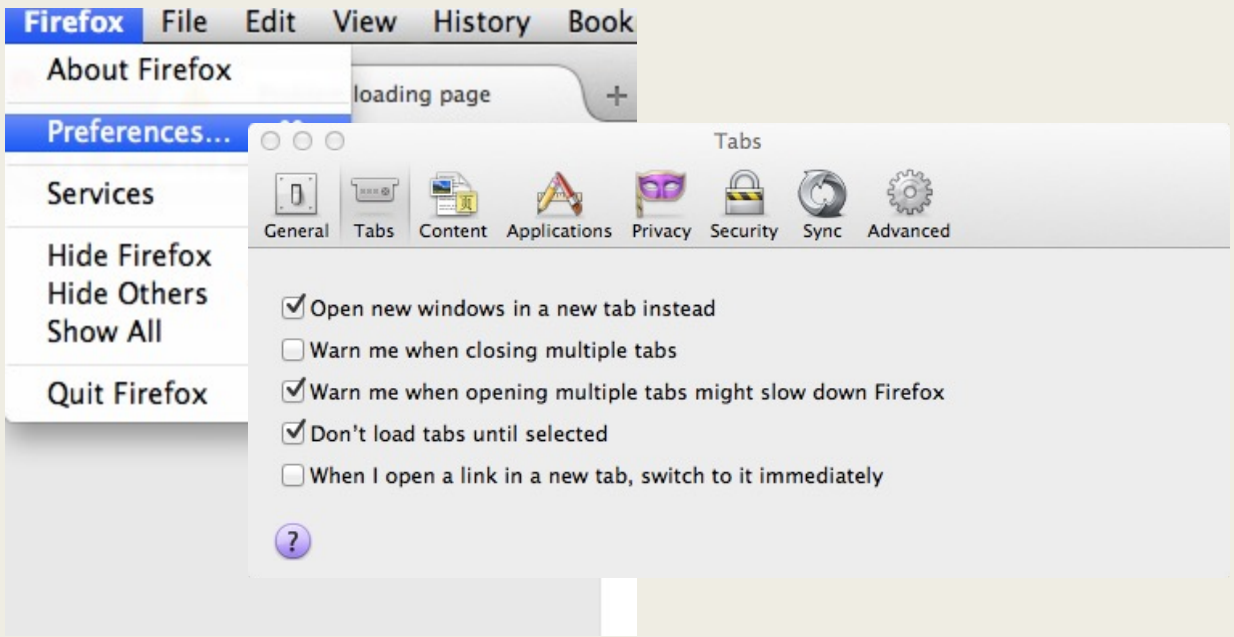
Oracles

Configurability

Configurability


Enter Query Sequence
Enter accession number(s), gi(s), or FASTA sequence(s)
Clear
Query subrange
gactacgatcgggc
From
To
BLAST
Search database Nucleotide collection (nr/nt) using Megablast (Optimize for highly similar sequences)
Show results in a new window
Algorithm parameters
Restore default search parameters
General Parameters
Max target sequences
100
Select the maximum number of aligned sequences to display
Short queries
Automatically adjust parameters for short input sequences
Expect threshold
10
Word size
28
Max matches in a query range
0
Scoring Parameters
Match/Mismatch Scores
1,-2
Gap Costs
Linear
Linear
Existence: 5 Extension: 2
Existence: 2 Extension: 2
Existence: 1 Extension: 2
Existence: 0 Extension: 2
Existence: 3 Extension: 1
Existence: 2 Extension: 1
Existence: 1 Extension: 1
Filters and Mas
Filter
for:
Homo sapiens (Human)
Mask
Mask for lookup table only
Mask lower case letters

Configurability

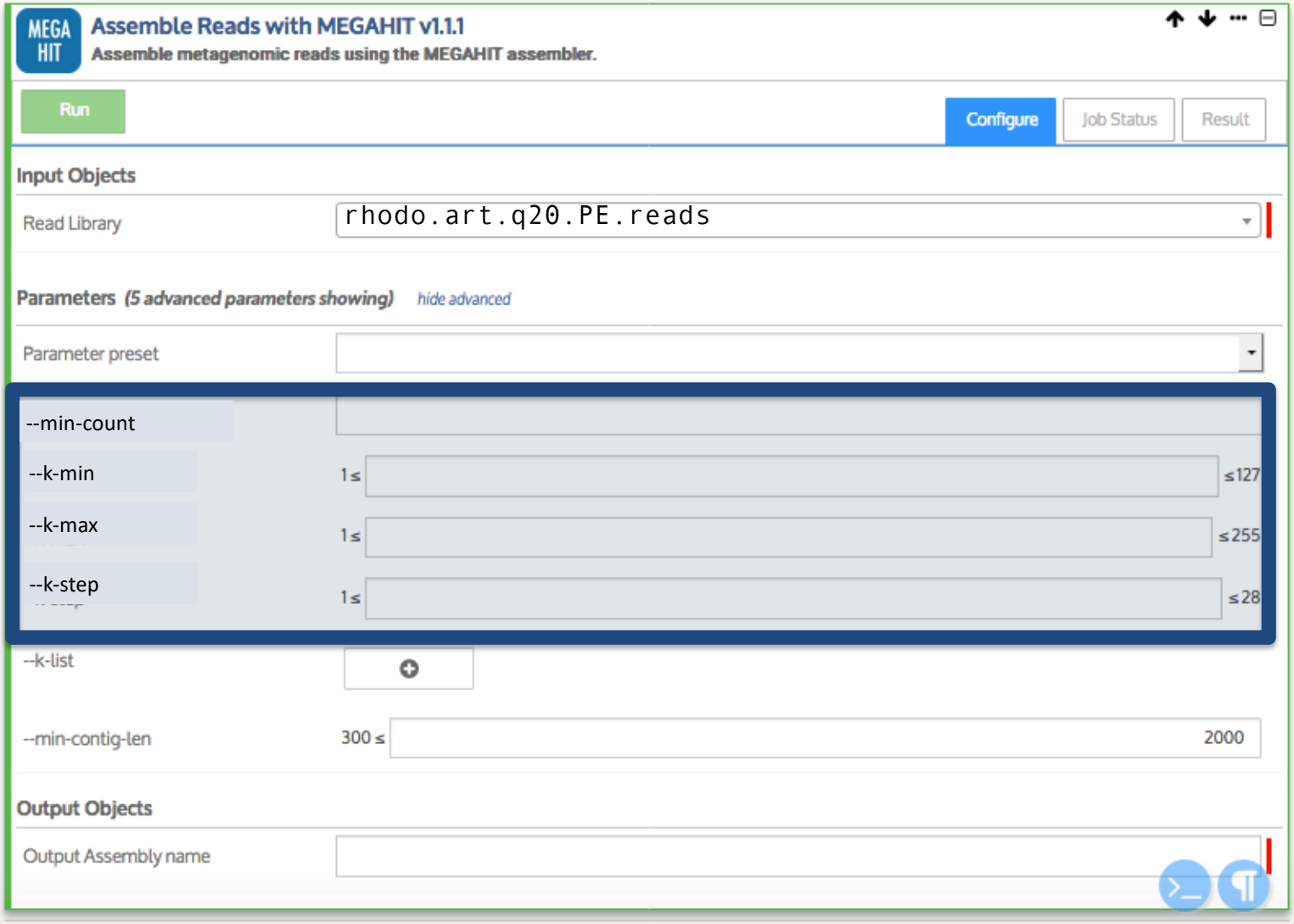


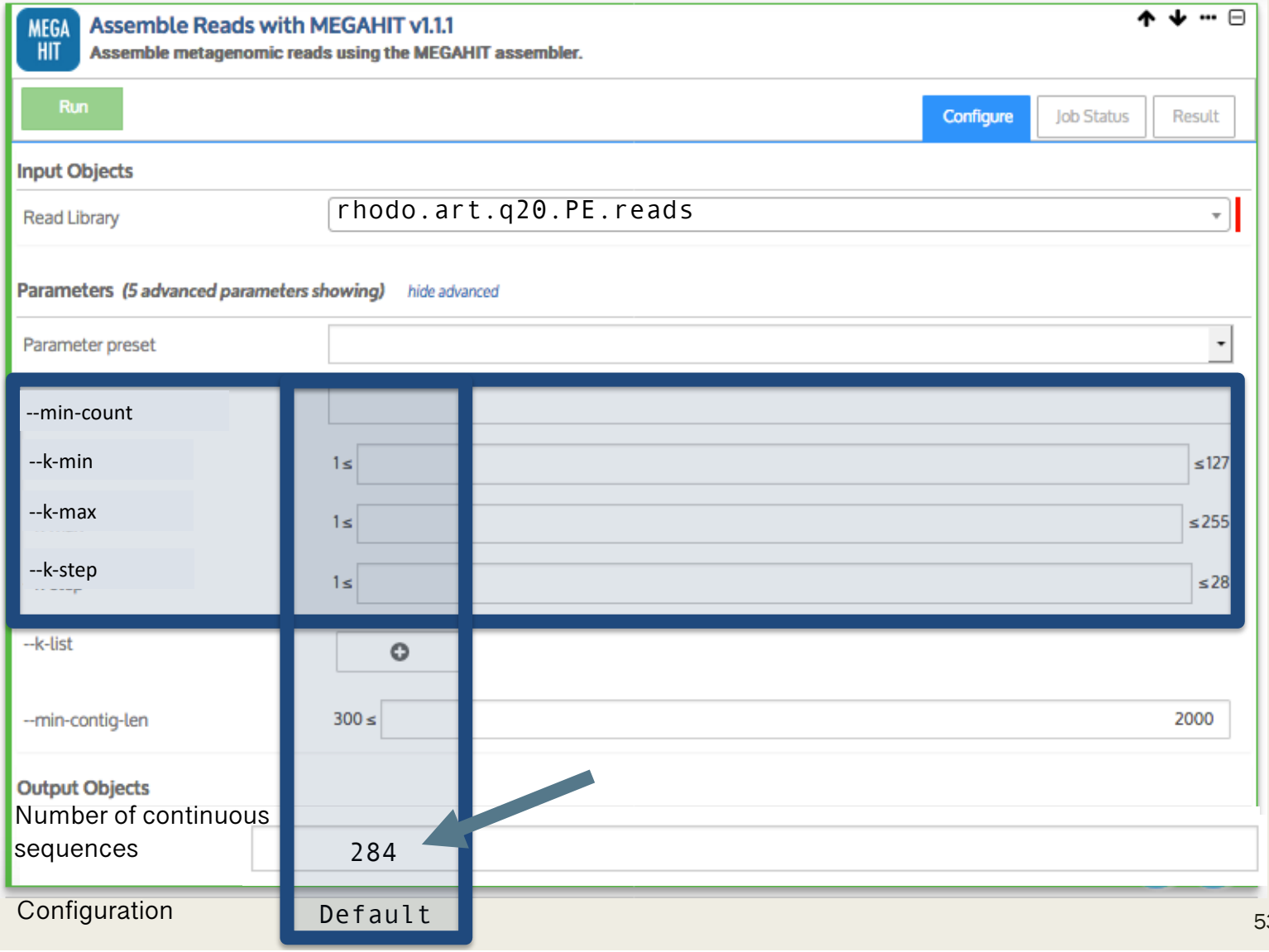
MEGAHIT (DNA Assembler)

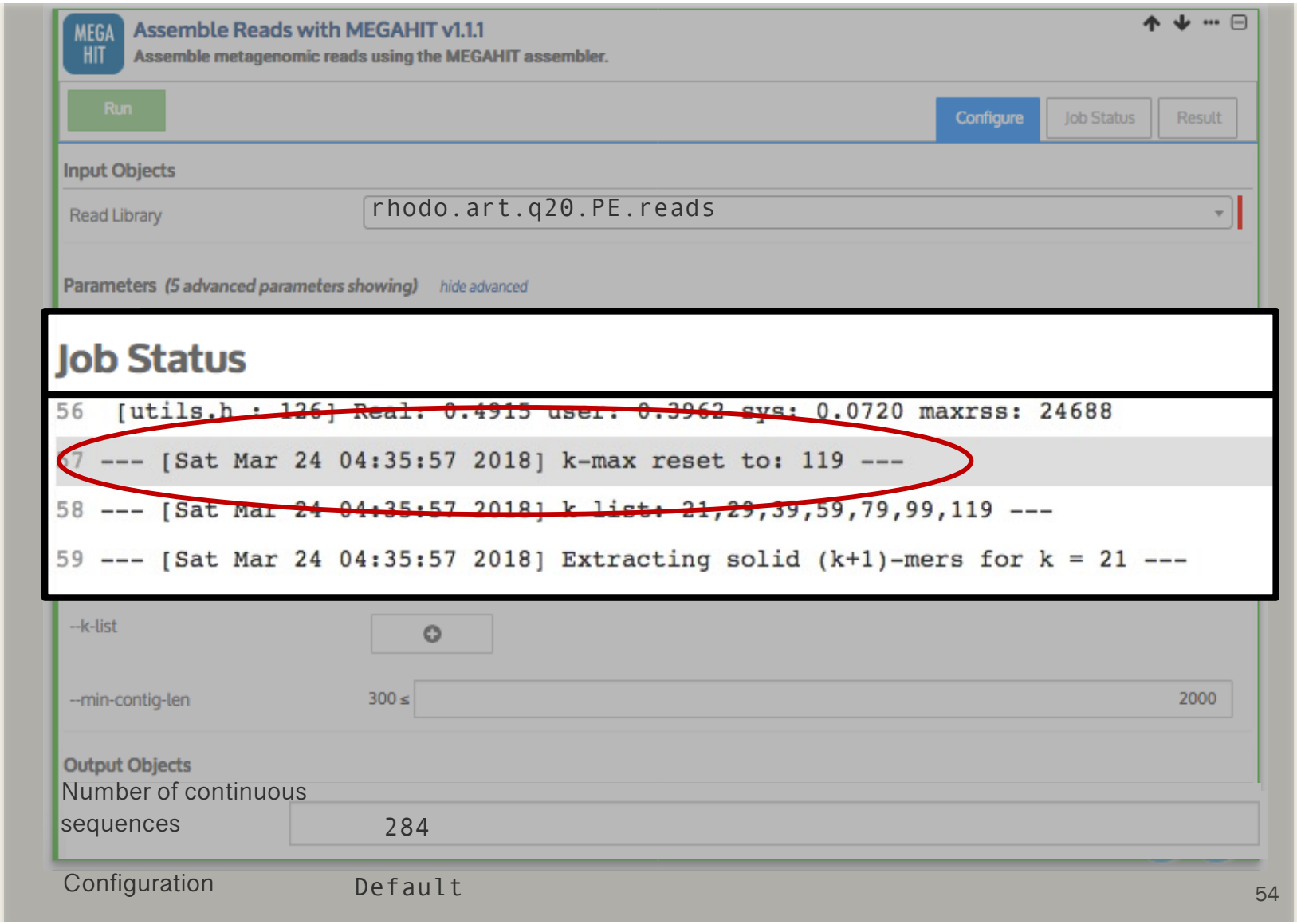
- DNA sequenced into small segments (reads)
- Assembly combines reads into longer *continuous sequences*
- Result is a certain number of *continuous sequences*

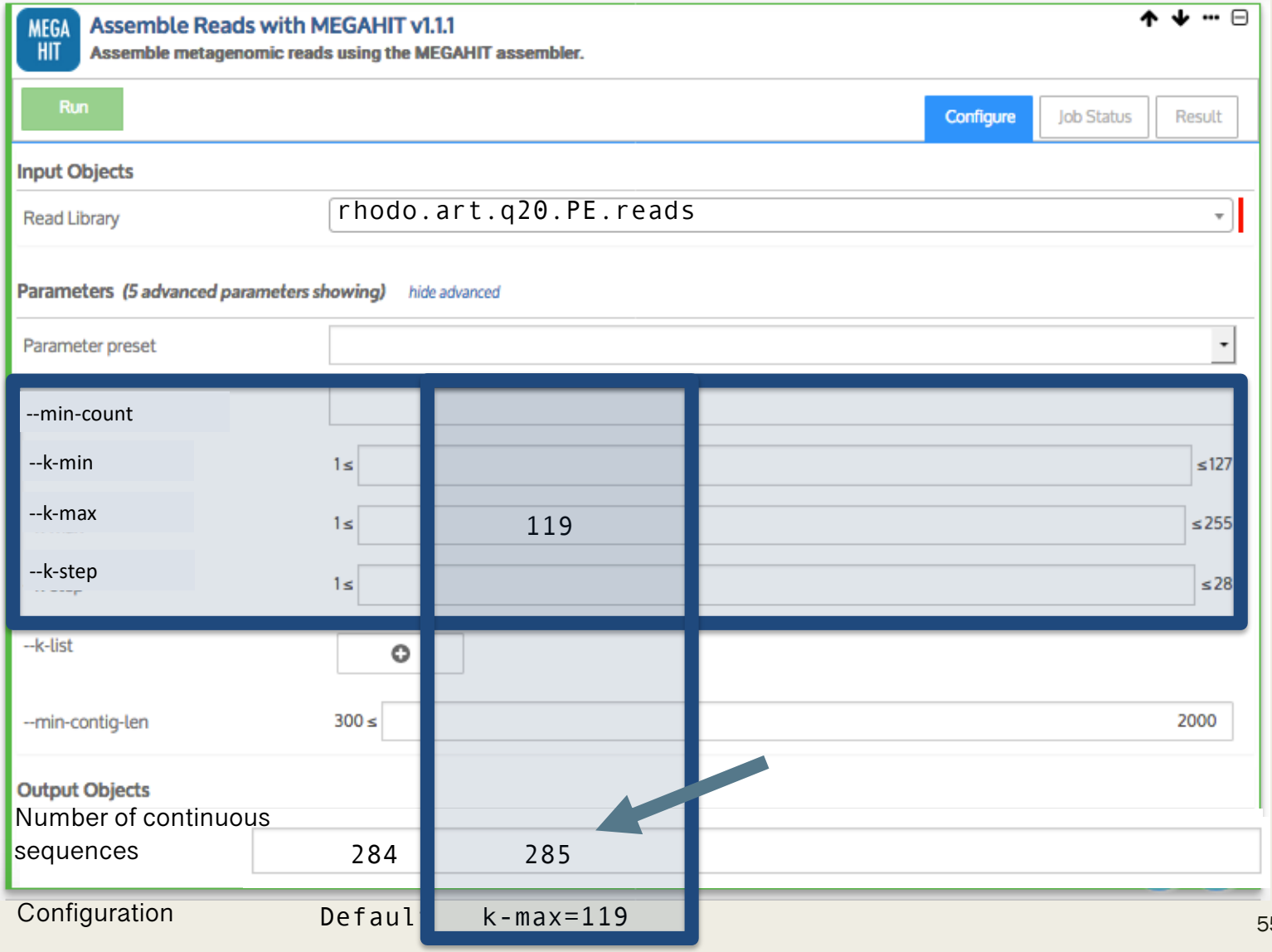
Credit to Mikaela Cashman

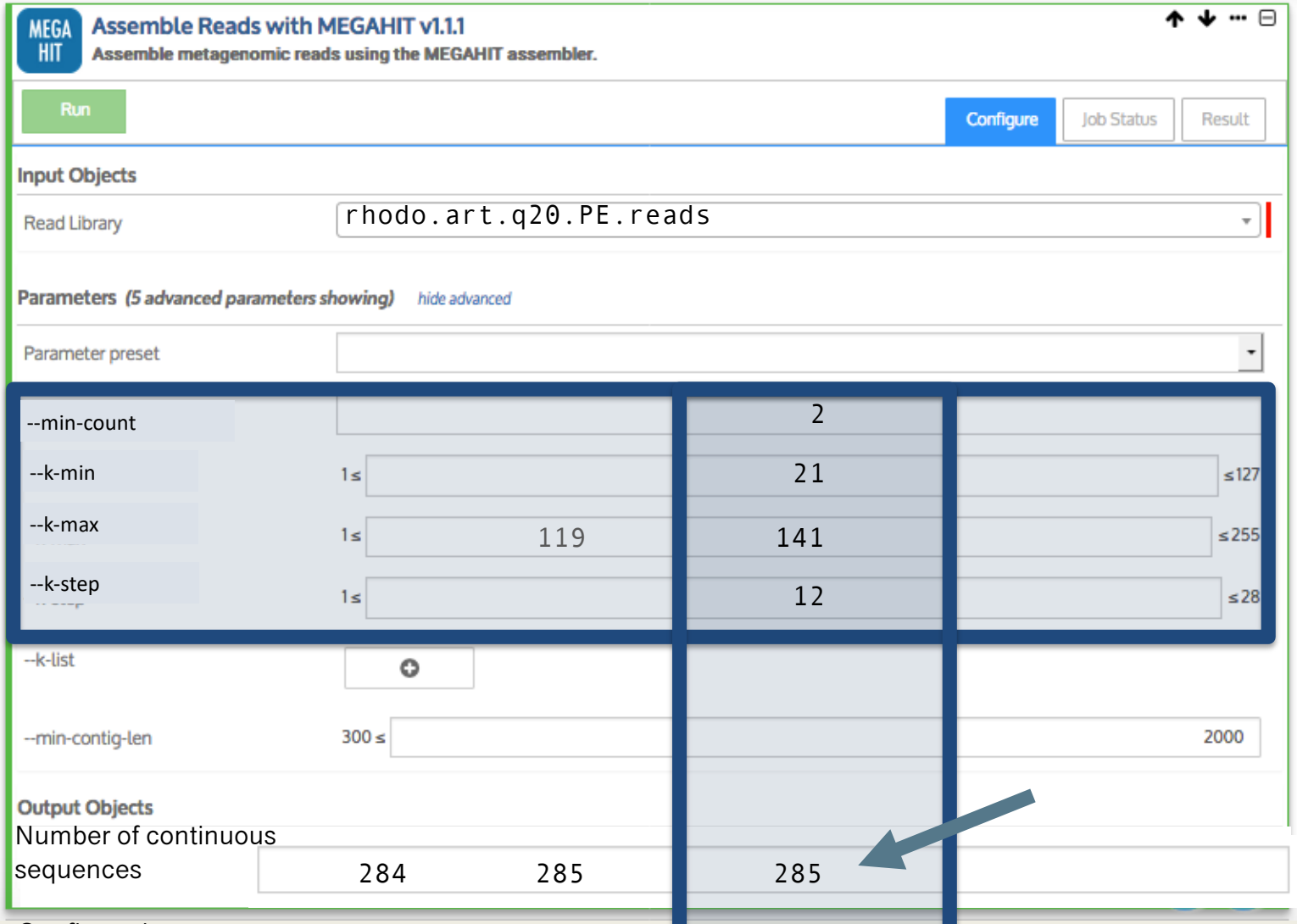
M. Cashman, M. B. Cohen, P. Ranjan, R. W. Cottingham, Navigating the Maze: the Impact of Configurability in Bioinformatics Software, *ASE, 2018*

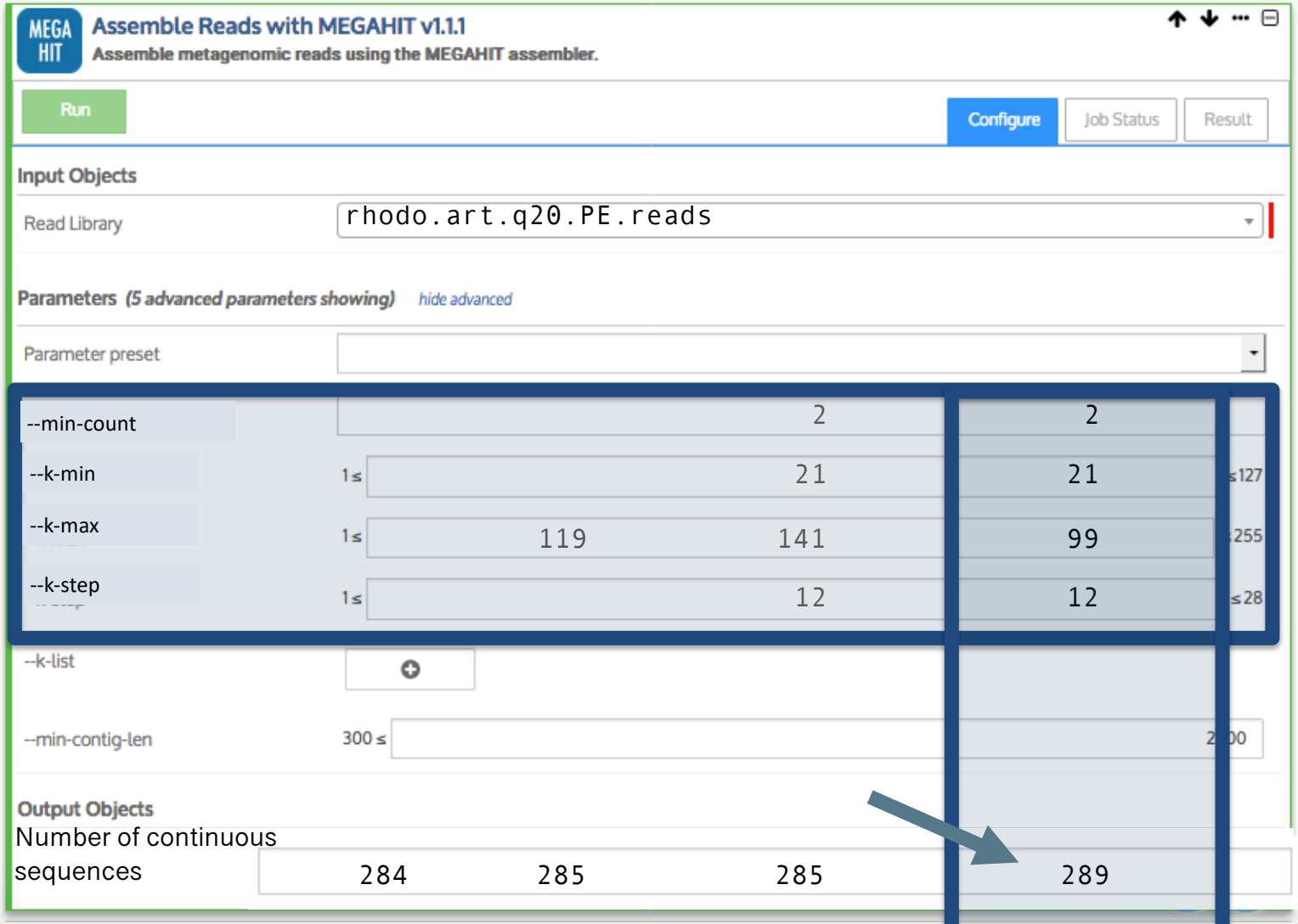

MEGA HIT
Assemble Reads with MEGAHIT v1.1.1
Assemble metagenomic reads using the MEGAHIT assembler.
Run
Configure
Job Status
Result
Input Objects
Read Library
rhodo.art.q20.PE.reads
Parameters (5 advanced parameters showing)
hide advanced
Parameter preset
--min-count
--k-min
1 ≤
≤127
--k-max
1 ≤
≤255
--k-step
1 ≤
≤28
--k-list
--min-contig-len
300 ≤
2000
Output Objects
Output Assembly name

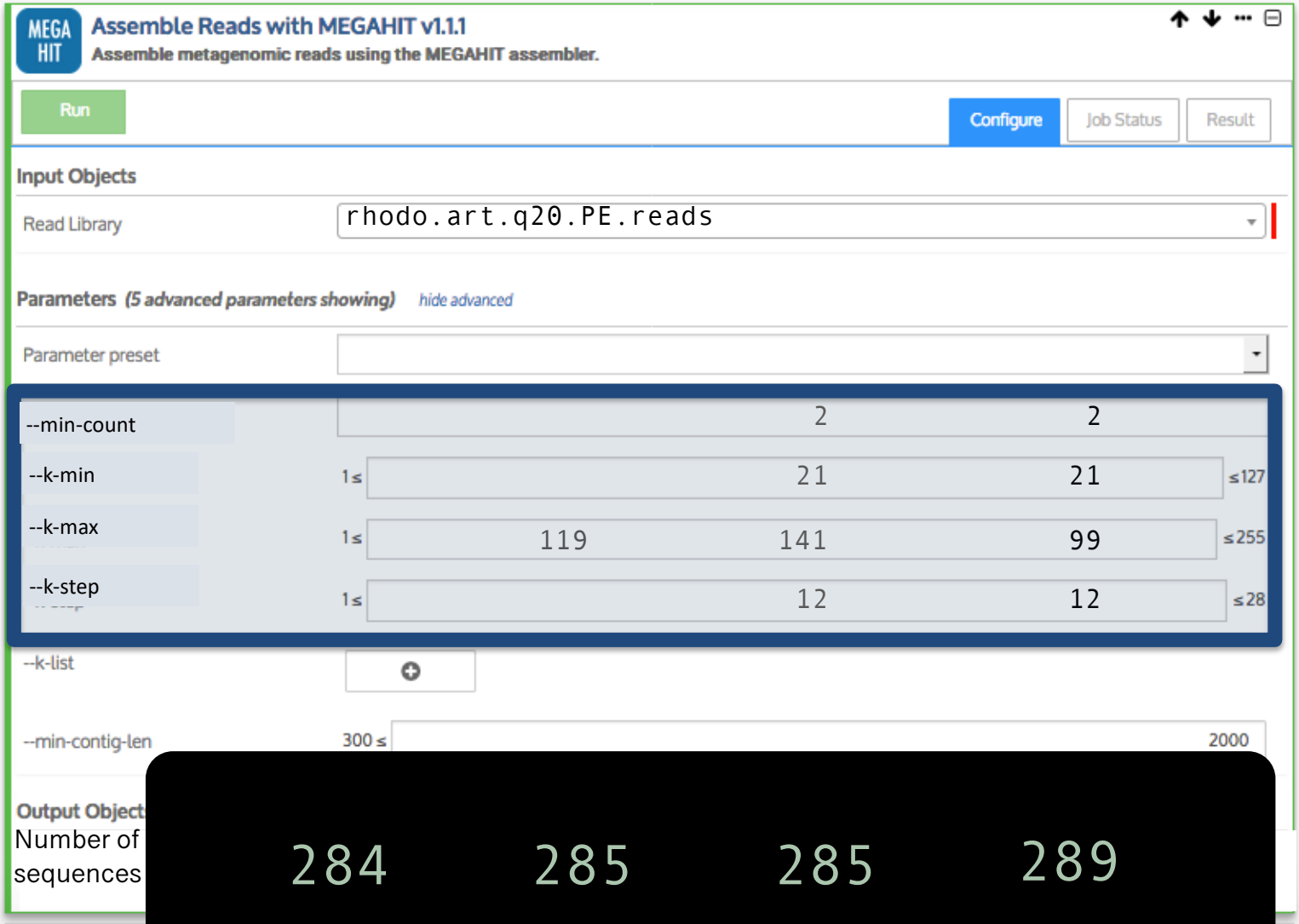

Assemble Reads with MEGAHIT v1.1.1
Assemble metagenomic reads using the MEGAHIT assembler.
Run
Configure
Job Status
Result
Input Objects
Read Library
rhodo.art.q20.PE.reads
Parameters (5 advanced parameters showing)
hide advanced
Parameter preset
--min-count
--k-min
1 ≤
≤127
--k-max
1 ≤
≤255
--k-step
1 ≤
≤28
--k-list
--min-contig-len
300 ≤
2000
Output Objects
Number of continuous sequences
284
Configuration
Default


MEGA HIT
Assemble Reads with MEGAHIT v1.1.1
Assemble metagenomic reads using the MEGAHIT assembler.
Run
Configure
Job Status
Result
Input Objects
Read Library
rhodo.art.q20.PE.reads
Parameters (5 advanced parameters showing)
hide advanced
Job Status
56 [utils.h : 126] Real: 0.4915 user: 0.3962 sys: 0.0720 maxrss: 24688
57 --- [Sat Mar 24 04:35:57 2018] k-max reset to: 119 ---
58 --- [Sat Mar 24 04:35:57 2018] k-list: 21,29,39,59,79,99,119 ---
59 --- [Sat Mar 24 04:35:57 2018] Extracting solid (k+1)-mers for k = 21 ---
--k-list
--min-contig-len
300 ≤
2000
Output Objects
Number of continuous sequences
284
Configuration
Default


MEGA HIT
Assemble Reads with MEGAHIT v1.1.1
Assemble metagenomic reads using the MEGAHIT assembler.
Run
Configure
Job Status
Result
Input Objects
Read Library
rhodo.art.q20.PE.reads
Parameters (5 advanced parameters showing)
hide advanced
Parameter preset
--min-count
--k-min
1 ≤
≤127
--k-max
1 ≤
119
≤255
--k-step
1 ≤
≤28
--k-list
--min-contig-len
300 ≤
2000
Output Objects
Number of continuous sequences
284
285
Configuration
Defaul
k-max=119


MEGA HIT
Assemble Reads with MEGAHIT v1.1.1
Assemble metagenomic reads using the MEGAHIT assembler.
Run
Configure
Job Status
Result
Input Objects
Read Library
rhodo.art.q20.PE.reads
Parameters (5 advanced parameters showing)
hide advanced
Parameter preset
--min-count
2
--k-min
1≤
21
≤127
--k-max
1≤
119
141
≤255
--k-step
1≤
12
≤28
--k-list
--min-contig-len
300 ≤
2000
Output Objects
Number of continuous sequences
284
285
285
Configuration
Default
k-max=119
Default-Manual


MEGA HIT
Assemble Reads with MEGAHIT v1.1.1
Assemble metagenomic reads using the MEGAHIT assembler.
Run
Configure
Job Status
Result
Input Objects
Read Library
rhodo.art.q20.PE.reads
Parameters (5 advanced parameters showing)
hide advanced
Parameter preset
--min-count
--k-min
--k-max
--k-step
--k-list
--min-contig-len
Output Objects
Number of continuous sequences
284
285
285
289
Configuration
Default
k-max=119
Default-Manual
k-max=99


MEGA HIT
Assemble Reads with MEGAHIT v1.1.1
Assemble metagenomic reads using the MEGAHIT assembler.
Run
Configure
Job Status
Result
Input Objects
Read Library
rhodo.art.q20.PE.reads
Parameters (5 advanced parameters showing)
hide advanced
Parameter preset
--min-count
2
2
--k-min
1≤
21
21
≤127
--k-max
1≤
119
141
99
≤255
--k-step
1≤
12
12
≤28
--k-list
--min-contig-len
300 ≤
2000
Output Object
Number of sequences
284
285
285
289
Configurati

Challenge for Testing

Encoding	Format	Cache Level	Closed-Captioning	Network Access
MPEG	Audio	Low	Yes	Yes
RAW	Video	Medium	No	No
WAV	Stream	High		

Media Player

Encoding	Format	Cache Level	Closed-Captioning	Network Access
MPEG	Audio	Low	Yes	Yes
RAW	Video	Medium	No	No
WAV	Stream	High		

Media Player

Encoding	Format	Cache Level	Closed-Captioning	Network Access
MPEG	Audio	Low	Yes	Yes
RAW	Video	Medium	No	No
WAV	Stream	High		

Media Player

Encoding	Format	Cache Level	Closed-Captioning	Network Access
MPEG	Audio	Low	Yes	Yes
RAW	Video	Medium	No	No
WAV	Stream	High		

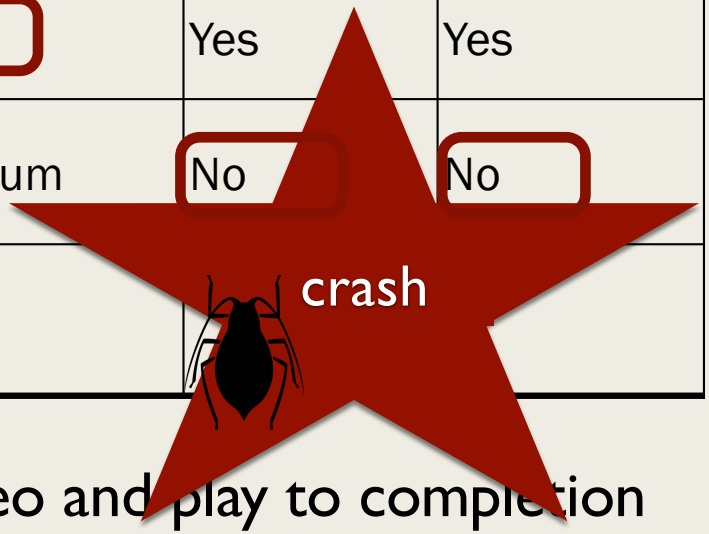
Testing the Player

Encoding	Format	Cache Level	Closed-Captioning	Network Access
MPEG	Audio	Low	Yes	Yes
RAW	Video	Medium	No	No
WAV	Stream	High		

Test Case: Open video and play to completion

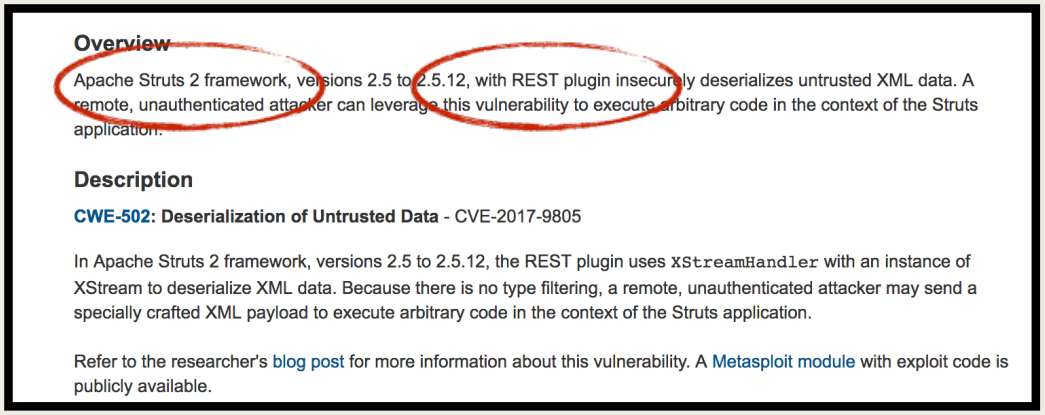
Interaction Fault

Encoding	Format	Cache Level	Closed-Captioning	Network Access
MPEG	Audio	Low	Yes	Yes
RAW	Video	Medium	No	No
WAV	Stream	High		



Test Case: Open video and play to completion

Configuration-Dependent Security Bugs (CERT)



Overview

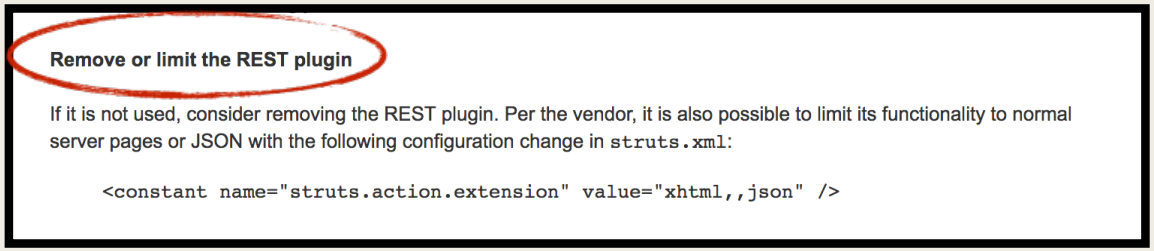
Apache Struts 2 framework, versions 2.5 to 2.5.12, with REST plugin insecurely deserializes untrusted XML data. A remote, unauthenticated attacker can leverage this vulnerability to execute arbitrary code in the context of the Struts application.

Description

CWE-502: Deserialization of Untrusted Data - CVE-2017-9805

In Apache Struts 2 framework, versions 2.5 to 2.5.12, the REST plugin uses `XStreamHandler` with an instance of XStream to deserialize XML data. Because there is no type filtering, a remote, unauthenticated attacker may send a specially crafted XML payload to execute arbitrary code in the context of the Struts application.

Refer to the researcher's blog post for more information about this vulnerability. A Metasploit module with exploit code is publicly available.



Remove or limit the REST plugin

If it is not used, consider removing the REST plugin. Per the vendor, it is also possible to limit its functionality to normal server pages or JSON with the following configuration change in `struts.xml`:

```
<constant name="struts.action.extension" value="xhtml,,json" />
```

Also Impacts Program Performance

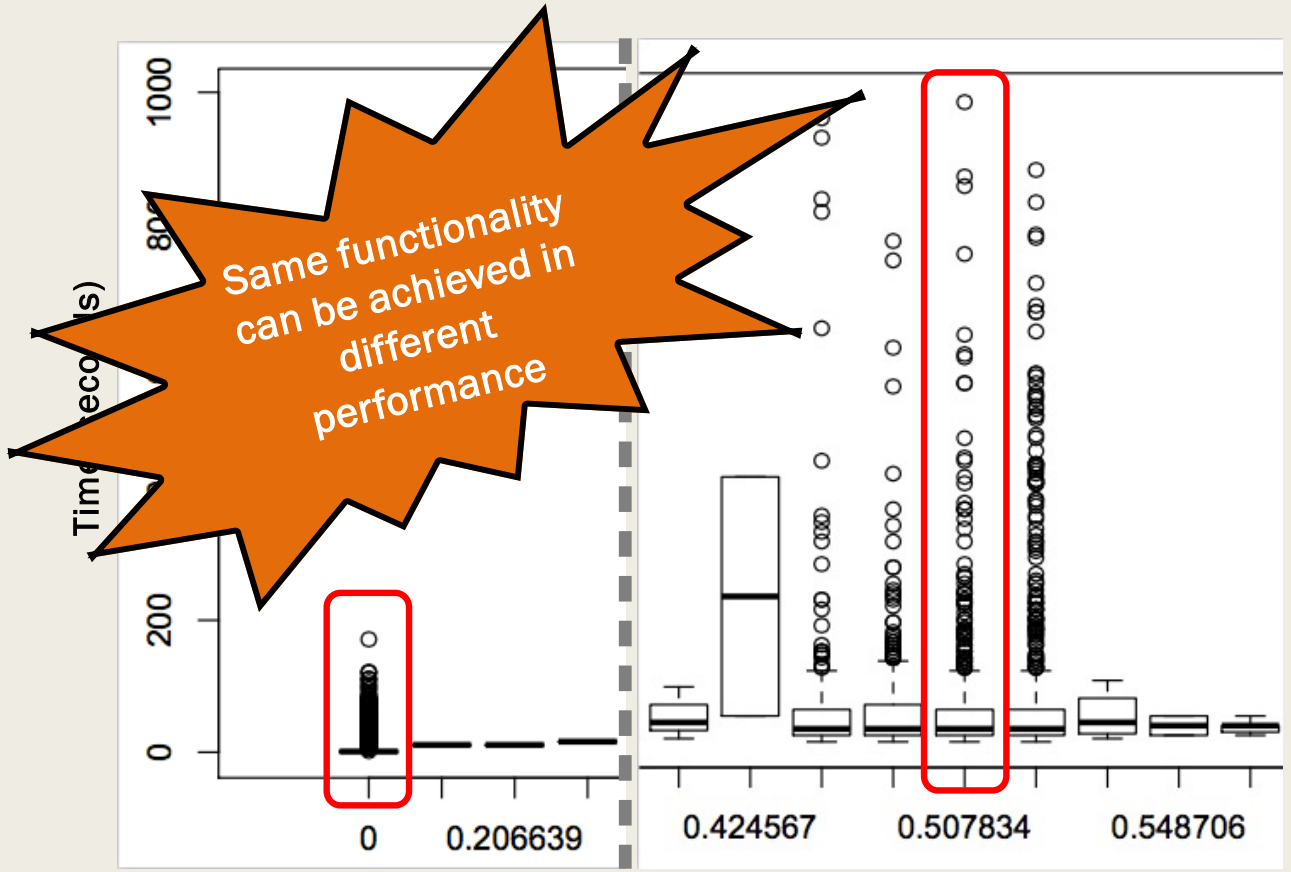


Figure credited to M. Cashman

Configuration Dependence

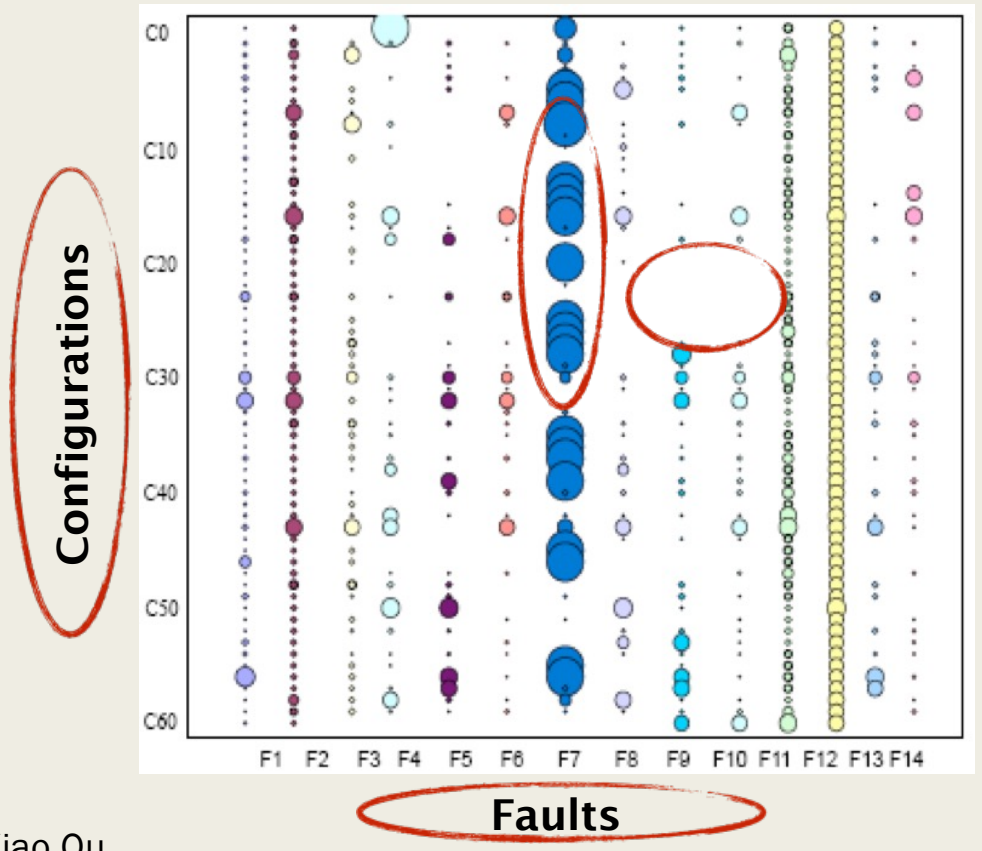


Figure credited to Xiao Qu

Real Configuration Spaces

Encoding	Format	Caching Level	Closed-Captioning	Network Access
MPEG	Audio	Low	Yes	Yes
RAW	Video	Medium	No	No
WAV	Stream	High		

Real Configuration Spaces

3 x 3 x 3 x 2 x 2 = 108 configurations

10 features with 5 options = 9,765,625 configs

4 hours to run test suite —> 4,459 years to run

gcc optimizer (199 options) — 10^{61}

Linux > 10,000 features

Combinatorial Interaction Testing (CIT)

- Sample the space so that all *t*-way combinations of values occur AT LEAST once
- *t* is defined as strength of testing

Combinatorial Interaction Testing (CIT)

	Encoding	Format	Caching Level	Closed-Captioning	Network Access
1	MPEG	Stream	Medium	Yes	Yes
2	RAW	Video	High	No	No
3	MPEG	Video	Low	No	Yes
4	WAV	Stream	High	No	Yes
5	RAW	Stream	Low	Yes	No
6	MPEG	Audio	High	Yes	No
7	WAV	Video	Medium	Yes	No
8	RAW	Audio	Medium	No	Yes
9	WAV	Audio	Low	Yes	Yes

Combinatorial Interaction Testing (CIT)

	Encoding	Format	Caching Level	Closed-Captioning	Network Access
1	MPEG	Stream	Medium	Yes	Yes
2	RAW	Video	High	No	No
3	MPEG	Video	Low	No	Yes
4	WAV	Stream	High	No	Yes
5	RAW	Stream	Low	Yes	No
6	MPEG	Audio	High	Yes	No
7	WAV	Video	Medium	Yes	No
8	RAW	Audio	Medium	No	Yes
9	WAV	Audio	Low	Yes	Yes

Combinatorial Interaction Testing (CIT)

	Encoding	Format	Caching Level	Closed-Captioning	Network Access
1	MPEG	Stream	Medium	Yes	Yes
2	RAW	Video	High	No	No
3	MPEG	Video	Low	No	Yes
4	WAV	Stream	High	No	Yes
5	RAW	Stream	Low	Yes	No
6	MPEG	Audio	High	Yes	No
7	WAV	Video	Medium	Yes	No
8	RAW	Audio	Medium	No	Yes
9	WAV	Audio	Low	Yes	Yes

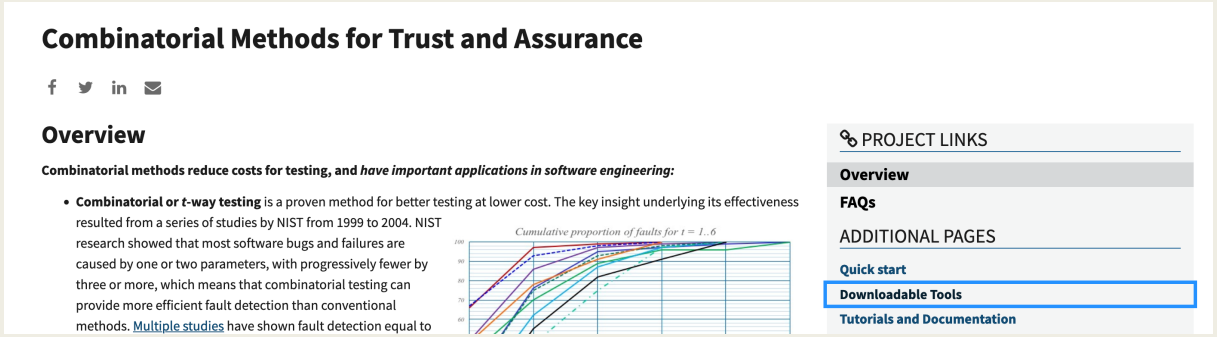
Combinatorial Interaction Testing (CIT)

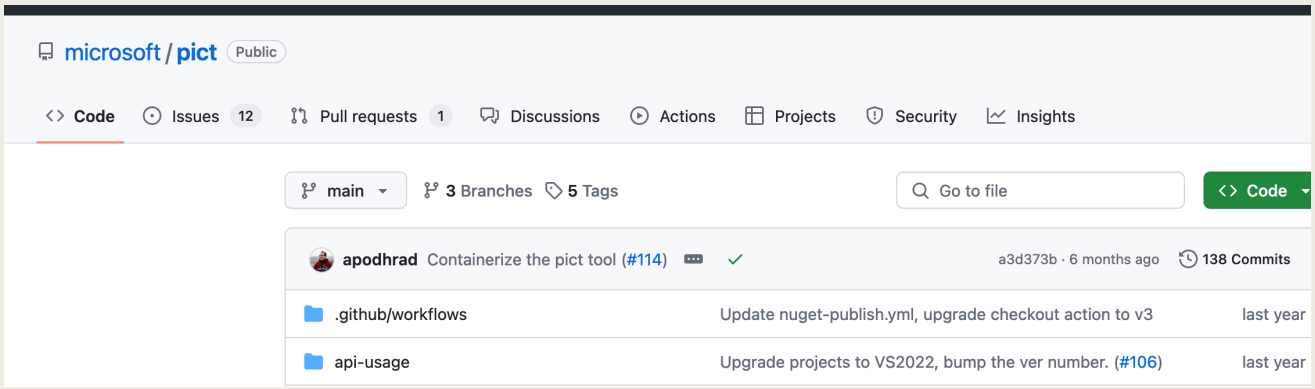
	Encoding	Format	Caching Level	Closed-Captioning	Network Access
1	MPEG	Stream	Medium	Yes	Yes
2	RAW	Video	High	No	No
3	MPEG	Video	Low	No	Yes
4	WAV	Stream	High	No	Yes
5	RAW	Stream	Low	Yes	No
6	MPEG	Audio	High	Yes	No
7	WAV	Video	Medium	Yes	No
8	RAW	Audio	Medium	No	Yes
9	WAV	Audio	Low	Yes	Yes

Some Combinatorial Testing Tools

ACTS

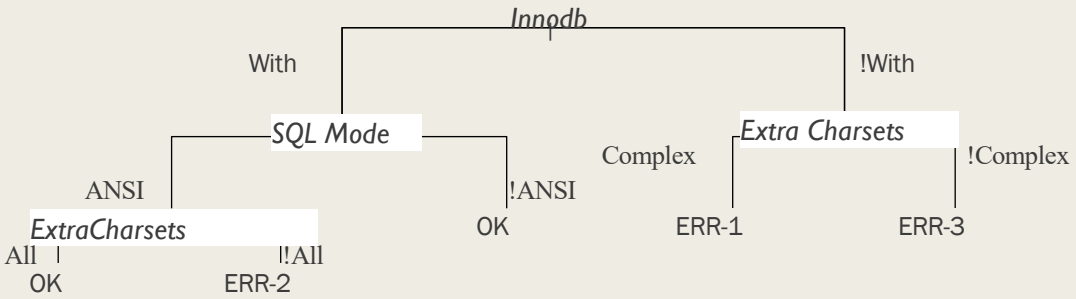
National Institutes of Standards





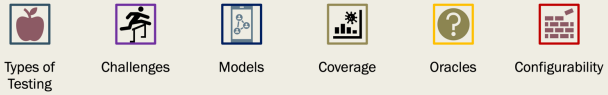
Characterizing Test Failures

- Use machine learning techniques (classification trees) to model option setting patterns that explain test failures



Summary

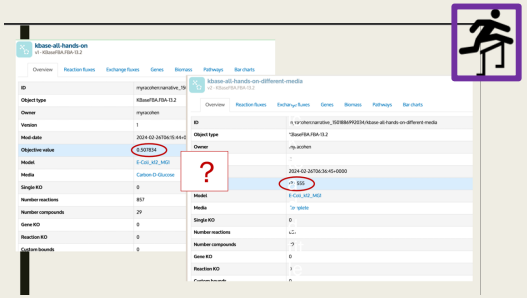
Overview



Some Techniques

Differential testing

Metamorphic testing



Configurability

This work was supported in part by the Better Scientific Software Fellowship Program, funded by the Exascale Computing Project (17-SC-20-SC), a collaborative effort of the U.S. Department of Energy (DOE) Office of Science and the National Nuclear Security Administration; and by the National Science Foundation (NSF) under Grant Nos. 2154495, 2234908, 1909688

Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the DOE or NSF.