This is the accepted version of a paper presented at 2016 IEEE Int. Conf. on Software Quality, Reliability and Security (QRS 2016).

Citation for the original published paper:


N.B. When citing this work, cite the original published paper.

Permanent link to this version:
http://urn.kb.se/resolve?urn=urn:nbn:se:kth:diva-199094
II. RELATED WORK

Existing prioritized combinatorial test generation algorithms [1], [2], [6], [10], [16] have evaluated their test suites with weight coverage and KL divergence but not fault detection effectiveness as described in Section I.

On the other hand, X. Qu et al. [17] evaluate fault detection effectiveness of test suites by an order-focused prioritized pairwise test generation algorithm called a deterministic density algorithm (DDA), which is a greedy algorithm proposed by R. Bryce and C. Colbourn [1]. X. Qu et al. presented priority weight extractions from code coverage and specification and showed that combinatorial test generation by DDA based on their weights can find faults more effectively than exhaustive test cases. They evaluate neither weight coverage nor KL divergence with fault detection effectiveness, and their research purpose is different from ours.

To evaluate the efficiency of combinatorial $t$-way testing, Petke et al. [15] investigate fault detection effectiveness of combinatorial $r$-way test suites ($2 \leq t \leq 6$) that are generated by a simulated annealing algorithm, CASA [19], and a greedy algorithm, ACTS [18]. They also examine the fault detection rate of test prioritization of the $t$-way test suites w.r.t. $r$-way interaction coverage with $2 \leq t \leq 6$, which means the test suites whose test cases are re-ordered in the descent order of $r$-way coverage.

Henard et al. [8] also evaluate the fault detection availability of test prioritization of exhaustive test suites w.r.t. $t$-way coverage with $2 \leq t \leq 4$ in their comparison of white-box prioritization and black-box prioritization. In addition, Henard et al. [7] examine $t$-way coverage and the fault detection rate by test prioritization w.r.t. test case similarity for software product line systems.

While we in this paper explore weight coverage and KL divergence with fault detection effectiveness of prioritized combinatorial testing with weighted SUT, the work [15], [7], [8] consider combinatorial testing with non-weighted SUT and investigate neither weight coverage nor KL divergence.

III. PRIORITIZED COMBINATORIAL TESTING

A. Prioritized pairwise testing

A system under test (SUT) for combinatorial testing is modeled from parameters, their associated values from finite sets, and constraints between parameter values. Table I shows an example SUT model with three parameters ($p_1$, $p_2$, $p_3$) and a constraint between $p_1$ and $p_3$: $p_1$ and $p_2$ have two values, $p_3$ has three values, and value pair ($b$, $g$) is not allowed by the constraint.

A test case for an SUT model assigns to each parameter a value that does not violate constraints in the SUT model. For example, a 3-tuple ($a$, $c$, $e$) is a test case for our example SUT model. We call a sequence of test cases a test suite.

A pairwise test suite for an SUT model is a test sequence to cover all possible value pairs between two parameters in the SUT model at least once. We say that a value pair is possible iff it does not violate SUT constraints. Table II shows an example pairwise test suite for the SUT model in Table I; it covers all possible 15 value pairs between two parameters, ($a$, $c$), ($a$, $d$), \ldots, ($d$, $g$).

Prioritized pairwise testing takes an SUT whose parameter values are assigned a weight representing a relative importance in testing, e.g., error probability, occurrence probability, and risk [10], and constructs a pairwise test suite that considers the weights. Existing algorithms for prioritized pairwise test generation are classified, depending on how weights are reflected in a test suite, into order-focused approaches, frequency-focused approaches, and their integration.

B. Order-focused prioritization and weight coverage

The algorithms in the order-focused approach, e.g., DDA [1] and CTE-XL [10], consider that highly weighted values (value pairs) should appear early in a test suite. Hence, they use weights to let higher-priority values appear earlier in test generation.

To evaluate a test suite $T$, they use a metric called weight coverage, which is defined as

$$WC(T) = \frac{\text{Sum of weights of value pairs covered by } T}{\text{Sum of weights of all possible value pairs}}.$$  

For example, weight coverage for the first two test cases of $T$ in Table II for the SUT in Table I is 0.5 since the sum of weights of all possible 15 values pairs is 4.4 and that of value pairs covered by $T$ is 2.2. In a test suite, order-focused prioritization uses higher-weighted values earlier, which implies obtaining higher weight coverage earlier.

C. Frequency-focused prioritization and KL divergence

The algorithms in the frequency-focused approach, e.g., PICT [3], the method by Fujimoto et al. [6], and FoCuS [16], consider that highly weighted values should appear frequently in a test suite. Hence, they use weights to utilize higher-priority values more often in test generation.

To evaluate a test suite $T$, they use KL divergence [12], which measures the difference between two probability distributions $P$ and $Q$ by

$$D(T) = \sum_v P(v) \log(P(v)/Q(v)),$$

where $P(v)$ and $Q(v)$ respectively denote the current frequency of each parameter value $v$ in $T$ and the ideal occurrence frequency for $v$. The frequency-focused prioritization assumes that the number of occurrences of $v$ is proportional to its weight.

For our example SUT in Table I, the ideal distribution $Q(v)$ is $2/3$, $1/3$, \ldots, $2/3$ for each value, $a$, $b$, \ldots, $g$. On the other hand, the current distribution $P(v)$ of test suite $T$ in Table II is $2/3$, $1/3$, \ldots, $1/3$, and the KL divergence $D(T)$ is 0.2310. By definition of KL divergence, $D(T)$ equals zero in the ideal situation, i.e., when $P = Q$, and it grows when the difference between $P$ and $Q$ is larger.
TABLE III
PROJECT DATA, NUMBER OF SEED FAULTS AND NUMBER OF DETECTED FAULTS.

<table>
<thead>
<tr>
<th>No.</th>
<th>project</th>
<th>ver.</th>
<th>LoC</th>
<th>seeded</th>
<th>detected by</th>
<th>detected by</th>
<th>pairwise tests by nine algo.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>all tests</td>
<td>prioritized</td>
<td>cs</td>
</tr>
<tr>
<td>1</td>
<td>flex</td>
<td>v1</td>
<td>12,160</td>
<td>19</td>
<td>16</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>2</td>
<td>v2</td>
<td>12,737</td>
<td>20</td>
<td>13</td>
<td>13</td>
<td>13</td>
<td>13</td>
</tr>
<tr>
<td>3</td>
<td>v3</td>
<td>12,781</td>
<td>17</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td>5</td>
<td>v5</td>
<td>12,893</td>
<td>9</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>grep</td>
<td>v1</td>
<td>12,507</td>
<td>18</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>7</td>
<td>v2</td>
<td>13,179</td>
<td>8</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>8</td>
<td>v3</td>
<td>13,291</td>
<td>18</td>
<td>8</td>
<td>7</td>
<td>8</td>
<td>7</td>
</tr>
<tr>
<td>9</td>
<td>v4</td>
<td>13,359</td>
<td>12</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>10</td>
<td>make</td>
<td>v1</td>
<td>18,460</td>
<td>19</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>11</td>
<td>v2</td>
<td>19,149</td>
<td>6</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>12</td>
<td>v3</td>
<td>20,340</td>
<td>5</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

TABLE IV
NUMBER OF ALL POSSIBLE TESTS, AND SIZES OF PAIRWISE TEST SUITES USED IN THE EXPERIMENT.

<table>
<thead>
<tr>
<th>No.</th>
<th>project</th>
<th>ver.</th>
<th># of all tests</th>
<th># of prioritized pairwise tests by nine algo.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>cs</td>
<td>co</td>
</tr>
<tr>
<td>1</td>
<td>flex</td>
<td>v1</td>
<td>525</td>
<td>52</td>
</tr>
<tr>
<td>2</td>
<td>v2</td>
<td>525</td>
<td>52</td>
<td>51</td>
</tr>
<tr>
<td>3</td>
<td>v3</td>
<td>525</td>
<td>52</td>
<td>53</td>
</tr>
<tr>
<td>4</td>
<td>v4</td>
<td>525</td>
<td>52</td>
<td>52</td>
</tr>
<tr>
<td>5</td>
<td>v5</td>
<td>525</td>
<td>52</td>
<td>52</td>
</tr>
<tr>
<td>6</td>
<td>grep</td>
<td>v1</td>
<td>470</td>
<td>75</td>
</tr>
<tr>
<td>7</td>
<td>v2</td>
<td>470</td>
<td>75</td>
<td>75</td>
</tr>
<tr>
<td>8</td>
<td>v3</td>
<td>470</td>
<td>75</td>
<td>78</td>
</tr>
<tr>
<td>9</td>
<td>v4</td>
<td>470</td>
<td>75</td>
<td>74</td>
</tr>
<tr>
<td>10</td>
<td>make</td>
<td>v1</td>
<td>793</td>
<td>33</td>
</tr>
<tr>
<td>11</td>
<td>v2</td>
<td>793</td>
<td>33</td>
<td>33</td>
</tr>
<tr>
<td>12</td>
<td>v3</td>
<td>793</td>
<td>33</td>
<td>33</td>
</tr>
</tbody>
</table>

D. Pricot

The algorithm in [2], which we call pricot, integrates the order-focused prioritization (shortly co) and the frequency-focused prioritization (shortly cf) with a size-focused prioritization (shortly cs) which considers that the size of a test suite should be small. To realize a small test suite where high-priority test cases appear early and frequently in a good balance, pricot takes a prioritization order of cs, co, and cf (e.g., cs > co > cf, denoted by cs.co.cf) as an input and generates a pairwise test suite that considers the weights in the given order.

To evaluate test suites, pricot uses both weight coverage and KL divergence [2]. Table II shows a pairwise test suite that is generated by pricot with co.cf, together with cumulative weight coverage and KL divergence of its test cases. For our case study to investigate the relation of fault detection effectiveness with weight coverage and KL divergence, we use pairwise test suites generated by pricot with various prioritization orders.

IV. EXPERIMENTS

A. Research Questions

We set up the following two research questions to investigate the effectiveness of existing evaluation metrics of prioritized combinatorial testing.

RQ1. Do order-focused prioritized combinatorial test suites with higher weight coverage achieve better fault detection effectiveness?

RQ2. Do frequency-focused prioritized combinatorial test suites with better (lower) KL divergence achieve better fault detection effectiveness?

B. Experimental Setting

1) Subjects: For empirical experiments, we use three open source projects of C programs, flex, grep, and make, from the Software artifact Infrastructure Repository (SIR) [20]. Each project includes

- multiple versions of programs with seeded faults,
- a test plan in Test Specification Language (TSL) [14],
- all test cases satisfying the test plan, and
- a bug report for each version of the project that describes which test case detects a fault.

Table III shows the lines of code (LoC) including comments, the number of seeded faults, the number of detected faults by all test cases. Table IV shows the number of all test cases for each version of the projects we use. The faults in the repository were hand-seeded by multiple developers to reflect real types of faults based on their experience [4]. We choose the versions whose number of detected faults is not zero from the repository.
2) SUT models: For each project, we construct an SUT model whose parameters, values, and constraints are fully extracted from the TSL specification. For example, Fig. 1 show a part of the test plan in TSL for project flex included in SIR. From the TSL specification, we construct the SUT model for flex whose parameters include Bypass use(= p), and Fast scanner(= v), values for p includes Bypass_on(= v), values for p includes FullScan(= v), and constraints include (p ≠ v). Table V shows the size of the SUT model for each project. In the table, the size of a model is expressed as \( k_1 \cdot k_2 \cdots k_n \) which indicates that the number of parameters is k and for each i there are \( k_i \) parameters that have g values. The size of constraints is expressed as \( h_1 \cdot h_2 \cdots h_n \) which indicates that the constraint is described in conjunctive normal form (CNF) with l variables whose Boolean value represents an assignment of a value to a parameter and for each j there are \( h_j \) clauses that have \( l_j \) literals.

3) Weights: For each version of the project, we extract the weight of each parameter value \( v \), denoted by \( w(v) \), from the bug report. We define \( w(v) \) as the conditional probability that a test case \( t \) detects a fault given that \( v \) is assigned to the test case \( t \). \( w(v) \) is then calculated using the Bayesian inference as follows [9]:

\[
w(v) = \frac{P(t \text{ detects a fault} \mid v \text{ is assigned to } t)}{P(v \text{ is assigned to } t \mid t \text{ detects a fault})} \tag{1}
\]

where

\[
w(v) = \frac{P(v \text{ is assigned to } t)}{P(t \text{ detects a fault})} \tag{2}
\]

We compute the above equation (2) and determine the weight for each parameter value \( v \) using the information in the bug report of SIR that describes whether each test case \( t \) detects a fault or not.

4) Test suites: We use prioritized pairwise test suites generated by pricot [2] for the constructed SUT models with constraints and weights. For each model, we use nine variants of test suites generated with the following prioritization orders: 1) cs, 2) co, 3) cf, 4) cs.co, 5) co.cs, 6) cs.cf, 7) co.cf, 8) cs.co.cf, and 9) co.cs.cf. In Tables III and IV, we show the size of each test suite and the number of faults detected by the test suite. We highlight the case where more faults are detected in Table III, and highlight the case where the size of the test suite is minimum in Table IV. For all subjects except grep v3, all the pairwise test suites detect all faults detected by all test cases, while sizes of the pairwise test suites are less than 18% of those of exhaustive test suites.

C. Evaluation metrics

To evaluate the fault detection effectiveness of a test suite \( T \), we use the metric called \( NAPFD \) (Normalized Average Percentage of Faults Detected) [17], which is defined by

\[
NAPFD(T) = p - \frac{F_1 + F_2 + \ldots + F_m}{m \times n} + \frac{p}{2n}
\]

where \( m \) denotes the number of faults detected by the all test cases, \( n \) denotes the number of test cases of \( T \), \( F_i \) \((1 \leq i \leq m)\) denotes the number of the test cases where fault \( i \) is detected, and \( p \) denotes the number of faults detected by \( T \) divided by \( m \). For example, assume that there are two faults and the first test case and the third test case of \( T \) in Table II detect each of the two faults. (We call this assumption \( X \) in the following.) \( NAPFD \) of \( T \) is 0.75(= 1 - 4/12 + 1/12).

\( NAPFD \) is a normalized \( APFD \) [5], which is a common metric to evaluate fault detection effectiveness of test prioritization in regression testing, for evaluating test suites with different sizes and thus different numbers of faults detected (See [17] for further details). \( NAPFD \) measures the area under the curve when the percent of detected faults is on the \( y \)-axis and the percent of test cases is on the \( x \)-axis; higher \( NAPFD \) implies faster and more effective fault detection.

To evaluate weight coverage and KL divergence for prioritized test suites with different sizes, we use normalized values of weight coverage \( WC \) and KL divergence \( D \) following \( NAPFD \), which we call Normalized Weight Coverage (NWC) and Normalized KL divergence (NKLD) respectively. We define \( NWC \) and \( NKLD \) of a test suite \( T \) as follows:

\[
NWC(T) = \frac{p_W}{n} (\sum_{1 \leq i \leq n} WC(T_i) - WC(T) - \frac{1}{2})
\]

\[
NKLD(T) = \frac{p_D}{n} (\sum_{1 \leq i \leq n} D(T_i) - D_{max}(T) - \frac{1}{2})
\]

where

- \( n \) denotes the number of test cases in \( T \),
- \( T_i \) denotes the test suite having the first \( i \) test cases in \( T \),
- \( p_W \) denotes \( WC(T) \) divided by the maximum value of weight coverage, i.e., 1.
- \( D_{max}(T) \) denotes the maximum value of \( D(T_i) \) for \( 1 \leq i \leq n \),
- \( p_D \) denotes \( D_{max}(T) \) divided by \( D_{max} \), where \( D_{max} \) denotes the maximum value of \( D_{max}(T') \) for each \( T' \) of all test suites for evaluation.

For the test suite \( T \) in Table II, \( NWC \) is 0.5871(= 4.0227/6 – 1/12). \( NKLD \) is 0.3543(= 3.9496/16 × 1.5041) – 1/12 where \( D_{max} = 1.5041 \).
the best
NAPFD
the best
For
flex
the best
NKLD
x
co
pricot
generated by
frequency-focused
prioritization.

for three subjects
(order-focused prioritization), and
correction
for no subject.

fig.
4 shows
scatter plots
with regression lines
and coefficients
R
for
the correlation
between
NKLD
and
NWC
and
NAPFD,
and
that
between
NKLD
and
NAPFD,
using
the
test
suites.
From
the
result,
NAPFD
is
correlated
with
NWC
(R = 0.389)
although
no
correlation
is
found
between
NAPFD
and
NKLD
(R = −0.101).
We
also
investigated
NWC, NKLD, and
NAPFD
of
the
minimum
test
suite
T
i
having
the
first
i
test
cases
of
each
test
suite
T
that
detect
all
faults
detected
by
T.
(For
example,
assuming
X
in
Section
IV,
the
minimum
test
suite
of
T
is
the
one
having
the
first
three
test
cases.)
fig.
5
shows
the
correlation
using
the
minimum
test
suites.
The
result
shows
that
NAPFD
is
more
significantly
related
with
NWC
(R = 0.556)
but
is
still
not
related
with
NKLD
(R = 0.146).

The
experimental
results
answer
to
the
research
questions,
RQ1
and
RQ2,
as
follows:
Combinatorial
test
generation
that
achieves
higher
weight
coverage
can
provide
better
(faster)
fault
detection
but
that
with
better
KL
divergence
might
not.

Basically,
frequency-focused
priority
aims
to
provide
more
effective
fault
detection
while
order-focused
prioritization
aims
to
provide
earlier
fault
detection.
Therefore,
to
investigate
the
fault
detection
effectiveness
of
frequency-focused
combinatorial
test
generation,
examining
the
correlation
of
KL
divergence
to
the
number
of
faults
detected
is
also
our
interest.
Unfortunately,
the
numbers
of
faults
detected
by
test
suites
used
in
our
experiments
are
almost
the
same,
and
thus
further
case
studies
on
more
software
projects
will
be
included
in
future
work.

V. CONCLUSION AND FUTURE WORK

This
paper
investigates
the
fault
detection
effectiveness
with
weight
coverage
and
KL
divergence
of
prioritized
combinatorial
test
generation.
In
our
empirical
evaluation
using
a
collection
of
open
source
utilities,
order-focused
combinatorial
test
generation
with
higher
weight
coverage
achieves
the
best
(fastest)
fault
detection
while
the
frequency-focused
combinatorial
test
generation
with
better
KL
divergence
fares
worse.
The
correlation
between
KL
divergence
and
the
effectiveness
w.r.t.
detecting
more
faults
will
be
investigated
in
future
work.
In
addition,
further
case
studies
on
software
projects
with
real
faults
is
an
important
future
work.
We
are
also
investigating
automated
methods
of
extracting
priority
weights
for
prioritized
combinatorial
testing
to
achieve
better
fault
detection
effectiveness.

ACKNOWLEDGMENTS

The
authors
would
like
to
thank
anonymous
referees
for
their
helpful
comments
to
improve
this
paper.
This
work
was
partly
supported
by
JSPS
KAKENHI
Grant
Number
16K12415.
Fig. 2. Number of faults detected, weight coverage, and KL divergence for sample subjects.
TABLE VII

<table>
<thead>
<tr>
<th>NAPFD</th>
<th>NWC</th>
<th>NKLD</th>
<th>NAPFD</th>
<th>NWC</th>
<th>NKLD</th>
<th>NAPFD</th>
<th>NWC</th>
<th>NKLD</th>
</tr>
</thead>
<tbody>
<tr>
<td>cs</td>
<td>0.9111</td>
<td>0.7204</td>
<td>0.4107</td>
<td>cs</td>
<td>0.8277</td>
<td>0.7140</td>
<td>0.4137</td>
<td>cs</td>
</tr>
<tr>
<td>co</td>
<td>0.9072</td>
<td>0.6349</td>
<td>0.5629</td>
<td>co</td>
<td>0.7907</td>
<td>0.6261</td>
<td>0.5676</td>
<td>co</td>
</tr>
<tr>
<td>cf</td>
<td>0.9571</td>
<td>0.7038</td>
<td>0.4105</td>
<td>cf</td>
<td>0.8107</td>
<td>0.6976</td>
<td>0.4137</td>
<td>cf</td>
</tr>
<tr>
<td>cs.co</td>
<td>0.9111</td>
<td>0.7234</td>
<td>0.4076</td>
<td>cs.co</td>
<td>0.8277</td>
<td>0.7169</td>
<td>0.4107</td>
<td>cs.co</td>
</tr>
<tr>
<td>co.cs</td>
<td>0.9772</td>
<td>0.6745</td>
<td>0.5096</td>
<td>cs.cs</td>
<td>0.8025</td>
<td>0.6666</td>
<td>0.5132</td>
<td>co.cs</td>
</tr>
<tr>
<td>cs.cf</td>
<td>0.9632</td>
<td>0.7204</td>
<td>0.4103</td>
<td>cs.cf</td>
<td>0.8137</td>
<td>0.7140</td>
<td>0.4133</td>
<td>cs.cf</td>
</tr>
<tr>
<td>co.cf</td>
<td>0.9767</td>
<td>0.6381</td>
<td>0.5493</td>
<td>cs.cf</td>
<td>0.8544</td>
<td>0.6294</td>
<td>0.5541</td>
<td>co.cf</td>
</tr>
<tr>
<td>cs.co.cf</td>
<td>0.9632</td>
<td>0.7234</td>
<td>0.4072</td>
<td>cs.co.cf</td>
<td>0.8137</td>
<td>0.7169</td>
<td>0.4103</td>
<td>cs.co.cf</td>
</tr>
<tr>
<td>co.cs.cf</td>
<td>0.9772</td>
<td>0.6738</td>
<td>0.5046</td>
<td>co.cs.cf</td>
<td>0.8602</td>
<td>0.6659</td>
<td>0.5083</td>
<td>co.cs.cf</td>
</tr>
</tbody>
</table>

TABLE VIII

<table>
<thead>
<tr>
<th>NAPFD</th>
<th>NWC</th>
<th>NKLD</th>
<th>NAPFD</th>
<th>NWC</th>
<th>NKLD</th>
<th>NAPFD</th>
<th>NWC</th>
<th>NKLD</th>
</tr>
</thead>
<tbody>
<tr>
<td>cs</td>
<td>0.8488</td>
<td>0.7352</td>
<td>0.4125</td>
<td>cs</td>
<td>0.9904</td>
<td>0.7165</td>
<td>0.4110</td>
<td>cs</td>
</tr>
<tr>
<td>co</td>
<td>0.9205</td>
<td>0.6547</td>
<td>0.5538</td>
<td>co</td>
<td>0.9904</td>
<td>0.6261</td>
<td>0.5676</td>
<td>co</td>
</tr>
<tr>
<td>cf</td>
<td>0.9207</td>
<td>0.7139</td>
<td>0.4117</td>
<td>cf</td>
<td>0.9902</td>
<td>0.7001</td>
<td>0.4109</td>
<td>cf</td>
</tr>
<tr>
<td>cs.co</td>
<td>0.8387</td>
<td>0.7353</td>
<td>0.4291</td>
<td>cs.co</td>
<td>0.9904</td>
<td>0.7178</td>
<td>0.4087</td>
<td>cs.co</td>
</tr>
<tr>
<td>co.cs</td>
<td>0.8565</td>
<td>0.6826</td>
<td>0.5143</td>
<td>co.cs</td>
<td>0.9904</td>
<td>0.6391</td>
<td>0.5447</td>
<td>co.cs</td>
</tr>
<tr>
<td>cs.cf</td>
<td>0.9207</td>
<td>0.7352</td>
<td>0.4120</td>
<td>cs.cf</td>
<td>0.9902</td>
<td>0.7165</td>
<td>0.4106</td>
<td>cs.cf</td>
</tr>
<tr>
<td>co.cf</td>
<td>0.9205</td>
<td>0.6547</td>
<td>0.5494</td>
<td>co.cf</td>
<td>0.9902</td>
<td>0.6294</td>
<td>0.5541</td>
<td>co.cf</td>
</tr>
<tr>
<td>cs.co.cf</td>
<td>0.8244</td>
<td>0.7353</td>
<td>0.4290</td>
<td>cs.co.cf</td>
<td>0.9902</td>
<td>0.7178</td>
<td>0.4083</td>
<td>cs.co.cf</td>
</tr>
<tr>
<td>co.cs.cf</td>
<td>0.8565</td>
<td>0.6626</td>
<td>0.5081</td>
<td>co.cs.cf</td>
<td>0.9904</td>
<td>0.6393</td>
<td>0.5392</td>
<td>co.cs.cf</td>
</tr>
</tbody>
</table>

REFERENCES


