Systematically Applying High-Level Mutations for Fuzz Testing Big Data Applications

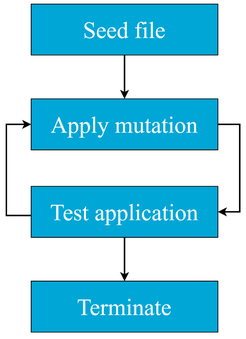
Lars van Koetsveld van Ankeren
L.vankoetsveldvanankeren@student.tudelft.nl
Supervisor - BK Ozkan

TU Delft

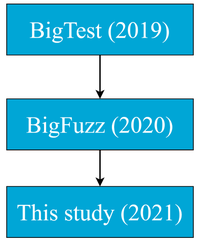
1 Background

- Fuzz testing is effective
- Recently applied to Big Data
- High-level mutations

Fuzzing loop



Big Data Fuzzing Research



2 Research question

'How does systematic exploration of high-level mutations affect the performance of a fuzz testing framework?'

I. How can high level mutations be applied systematically?
II. How does systematic exploration perform when compared to random selection of mutations?
III. Which program properties determine the performance of systematic exploration?

3 Systematic Exploration

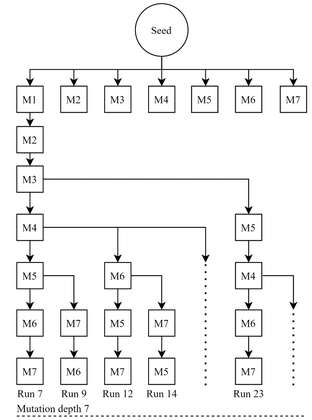
Approach:

- Combine mutations in consecutive runs
- Exclude illogical combinations
- Depth-first & depth bounded search

Mutation types

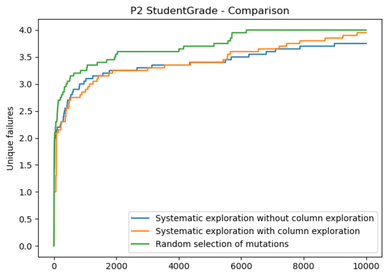
Mutation	Description	Column specific
Data value	Change data to arbitrary integer.	Yes
Data type	Change data type from float to string, integer to float or string to integer.	Yes
Data format	Change delimiter to "~" or if "~" to ",".	No
Data column	Insert arbitrary character in data.	Yes
Null data	Remove column from a row of data.	Yes
Empty data	Mutate data to empty string.	Yes
Add data	Add a column to row of data.	No

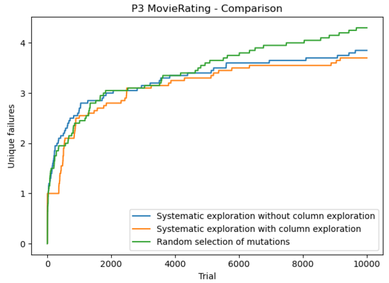
Systematic search

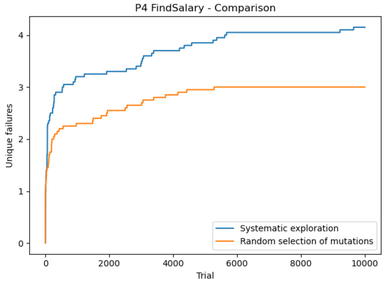


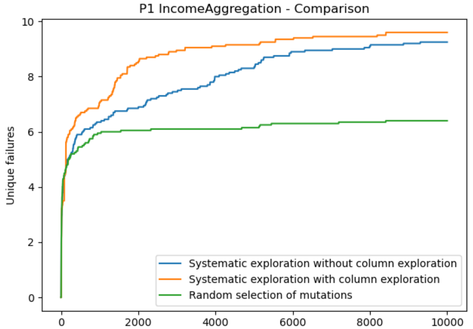
4 Results

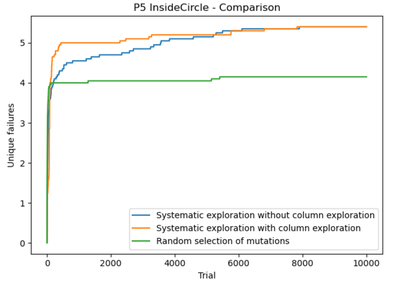
Average number of unique failures found over 20 iterations











5 Conclusions

Findings

- Systematic exploration finds more failures for majority of benchmarks
- Needs more trials for two out of five benchmarks
- Exploring all columns finds more failures
- Input specification determines performance

Future work

- Apply systematic high-level mutation testing to other fields
- Additional fuzz testing big data research