BSSE System and Software Engineering

Evaluating Test Data Generation for Untyped Data Structures Using Genetic Algorithms

Ralf Gerlich

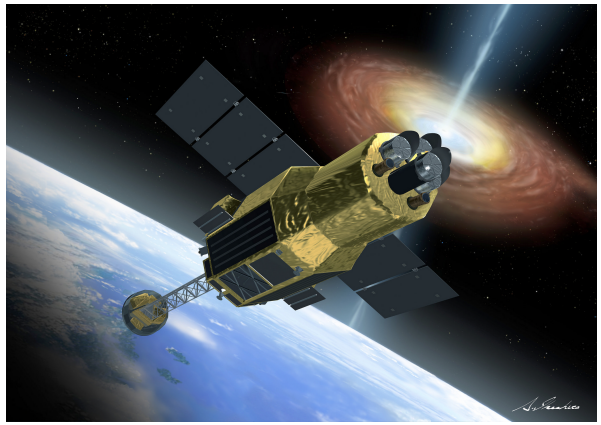
Christian R. Prause

This work is supported by a grant from the German Federal Ministry for Economic Affairs and Energy, based on a decision of the German Bundestag, grant No. 50PS1601.

Motivation

- Spacecraft software assumes critical functions
- Failure may be costly
  - Ariane 5.01: €1 Billion
  - Hitomi: $286 Million
- Software is thoroughly verified, e.g. by test
- High effort ⇒ automation possible?

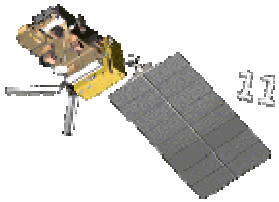
Public domain. Fabio Baccaglioni. https://www.youtube.com/watch?v=Z9EnUQltR9A (2017-04)

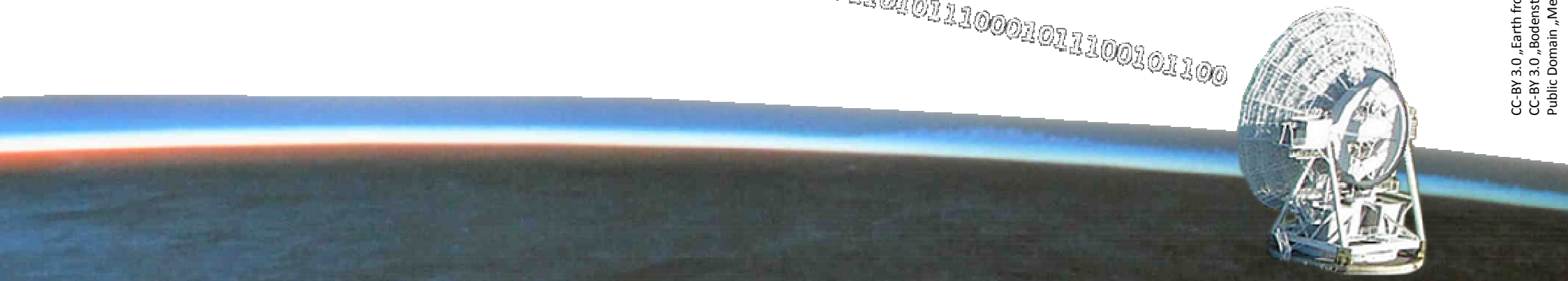


CC BY 2.0: JAXA/Akihiro Ikeshita. https://www.flickr.com/photos/nasablueshift/14070846521 2017-05-02

Test Domain

- Spacecraft processes Telecommands from Ground Station, rejects invalid commands
- Commands arrive as untyped bytestreams
- How to generate test inputs?

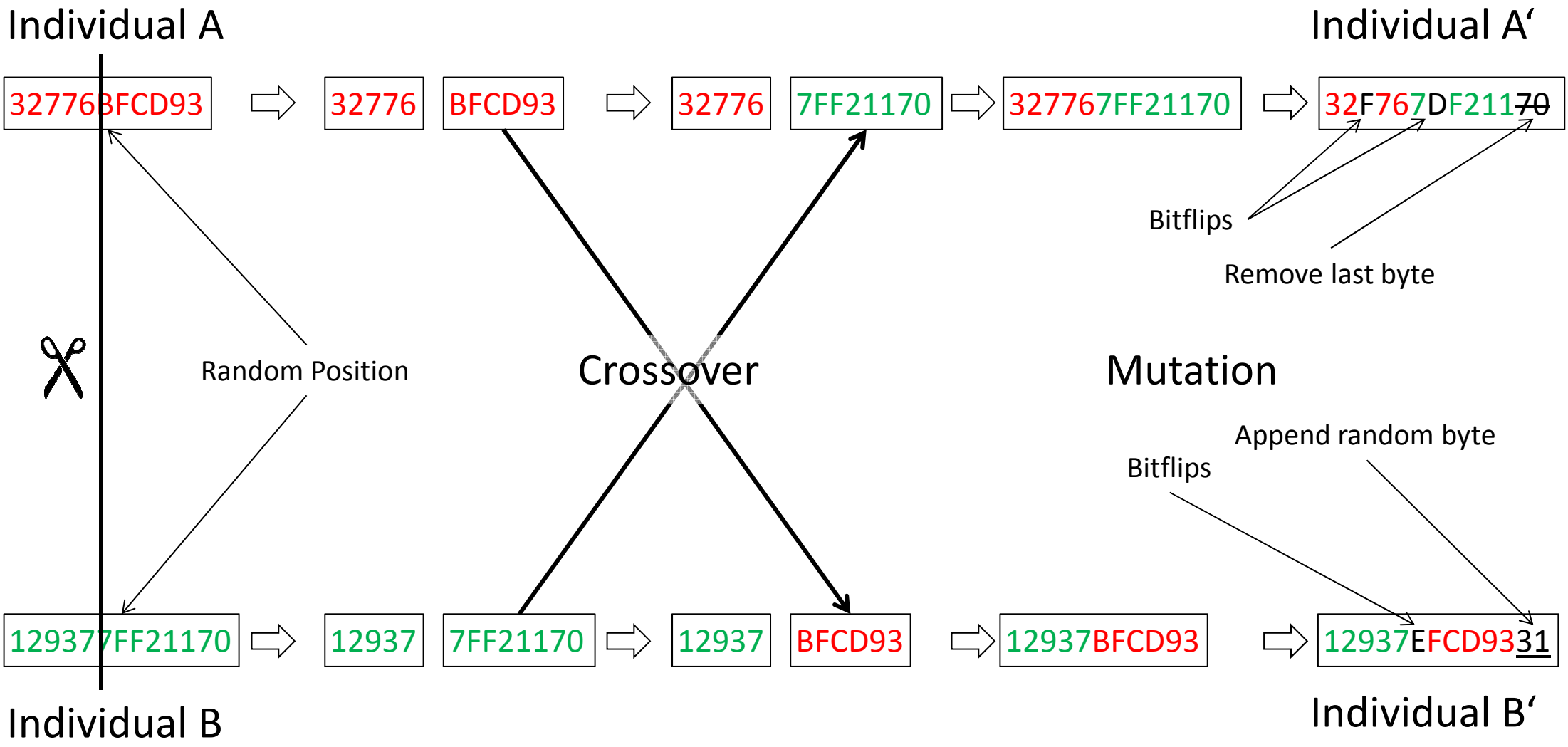




Genetic Algorithms

- Evolution towards optimization goal
- Sequence of Generations
- Individuals are Byte Sequences
- Fitness determines probability of procreation
- Genetic Operators modify individuals
  - Cross-Over
  - Mutation with probabilistic reversal
- Elite gets into next generation
- Influx due to immigration

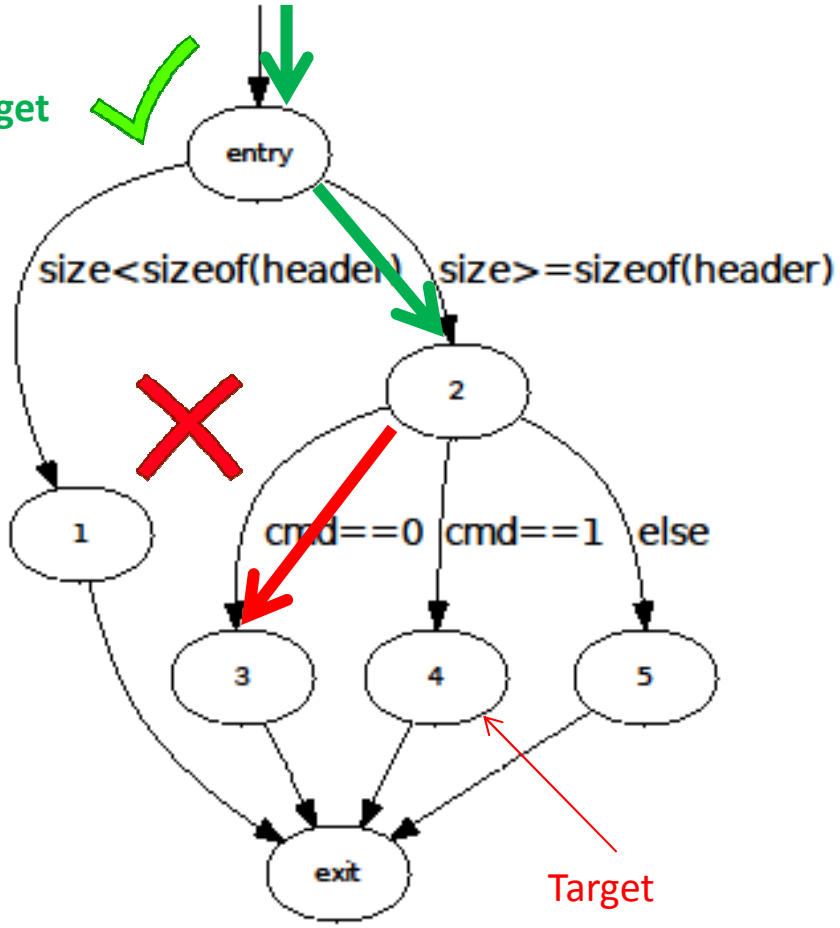
Cross-Over and Mutation



Cost-Function

Can still reach target
Cost = 0

Cannot reach target anymore
Cost > 0

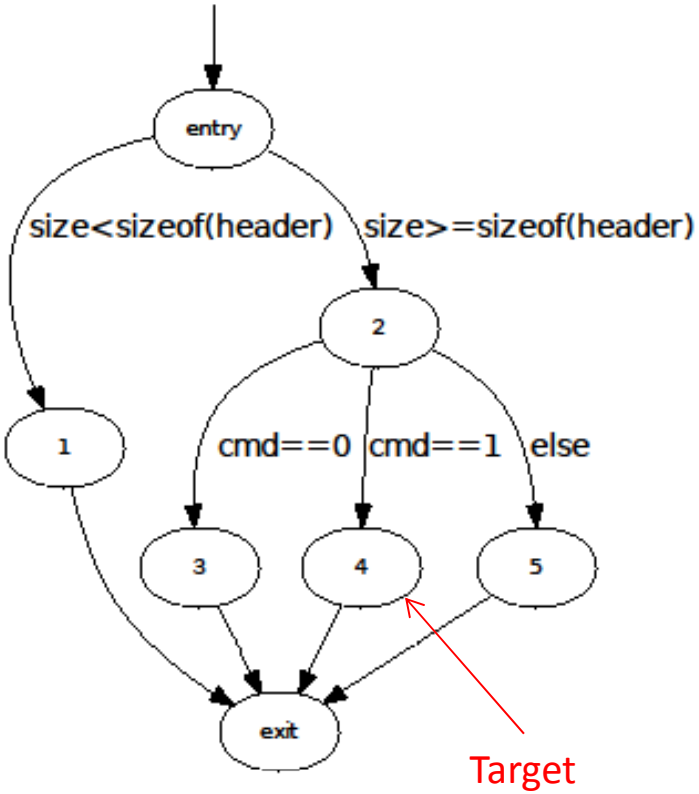


Cost Function describes how far individual is off from reaching target

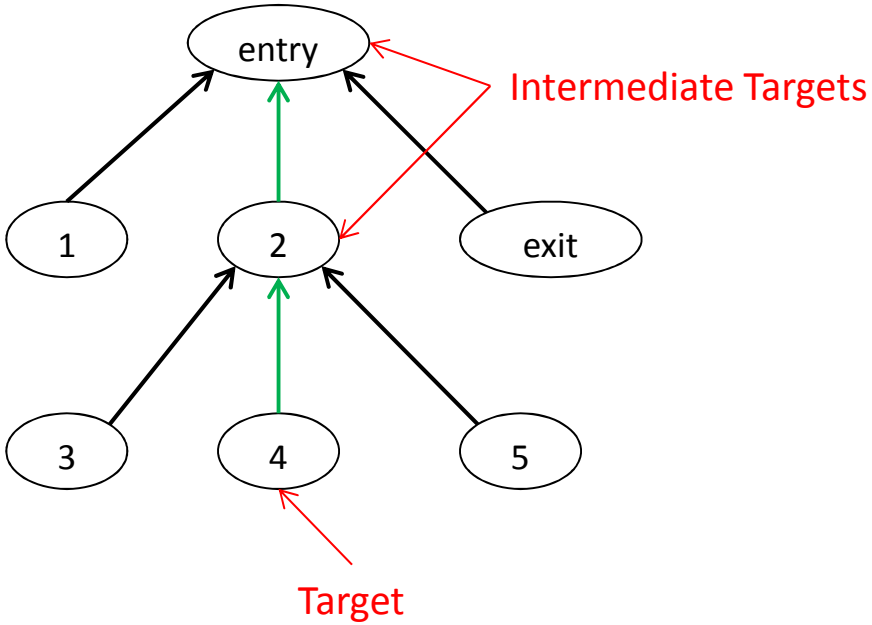
Desired Condition	Cost Function
E op F (op ∈ {=,<,>,≤,≥})	$\begin{cases} 0 & \text{if } E \text{ op } F \\ \lvert F - E \rvert & \text{otherwise} \end{cases}$
E != F	$\begin{cases} 1 & \text{if E op F} \\ 0 & \text{otherwise} \end{cases}$

Sequential Approach: Intermediate Target

Control-Flow-Graph



Dominator Graph



- Single Step Approach: Use final target only
- Sequential Approach: Use intermediate targets

Preliminary Evaluation

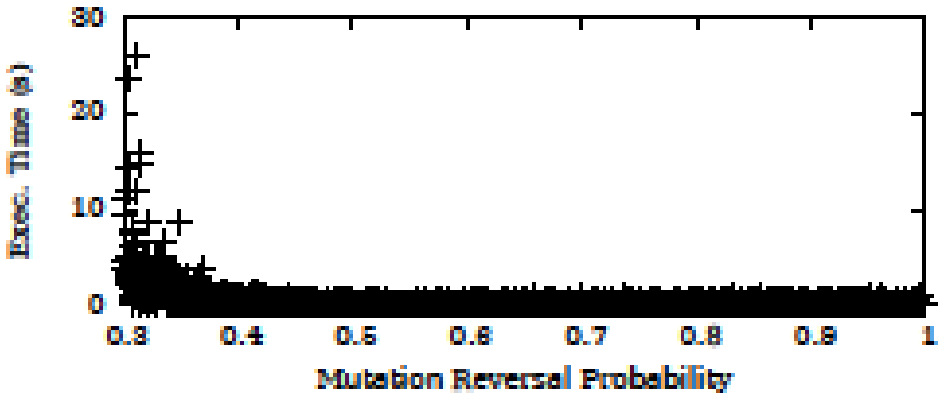
- Analyse Feasibility
- Compare runtime of algorithm variants
- Determine impact of parameters on runtime
  - Population Size
  - Elite Proportion
  - Immigrant Proportion
  - Mutation Probabilities
  - Mutation Reversal Probability
- So far only on simple example

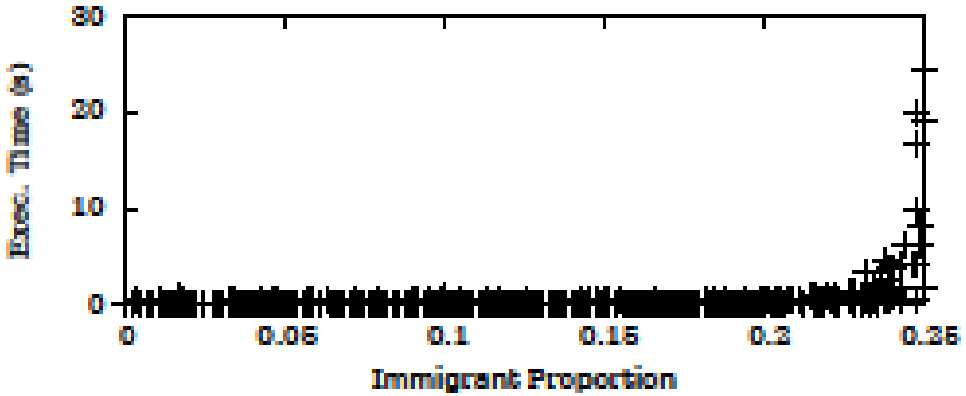
Runtime Measurements: Algorithm Variants

Variant	Min (s)	Mean (s)	Max (s)
Sequential	0.161	2.595	15.931
Single-Step	0.268	15.553	146.180

- Sequential is way faster than Single-Step
  - Mean runtime: Factor 6
  - Maximum runtime: Factor 9

Runtime Measurements: Parameters (1/2)





- Mutation Reversal has positive effect on runtime
- Does not seem to extend past p≳0.4

- Immigrant Proportion has no noticeable effect for p≲0.2
- Negative effect for p≳0.2

Runtime Measurements: Parameters (2/2)

Byte Extension Probability	0.00	0.25	0.50	0.75	1.00
Mean Execution Time (s)	0.46	0.13	0.12	0.12	0.13

- Optimum Value in the range $0.5 \lessapprox p \lessapprox 0.75$
- Decrease of Factor 3 from p=0 to p=0.25
- No further noticeable change for $p \rightarrow 1$

Conclusions so far

- Sequential Approach is superior
- Significant positive impact of mutation reversal
- No positive effect from immigration, negative effect for more than 20% of population
- This genetic algorithm seems promising for generating telecommand test data

Future Work

- Integration with random test tool (DCRTT) is underway…
- Application to industry-grade code
- Comparison to random testing performance
- More detailed analysis of parameter impacts
  - Multivariate
  - Measurements on more realistic code

Thank you for your attention!

Important Question:

Is there a general „good enough“ set of parameters for all applications?