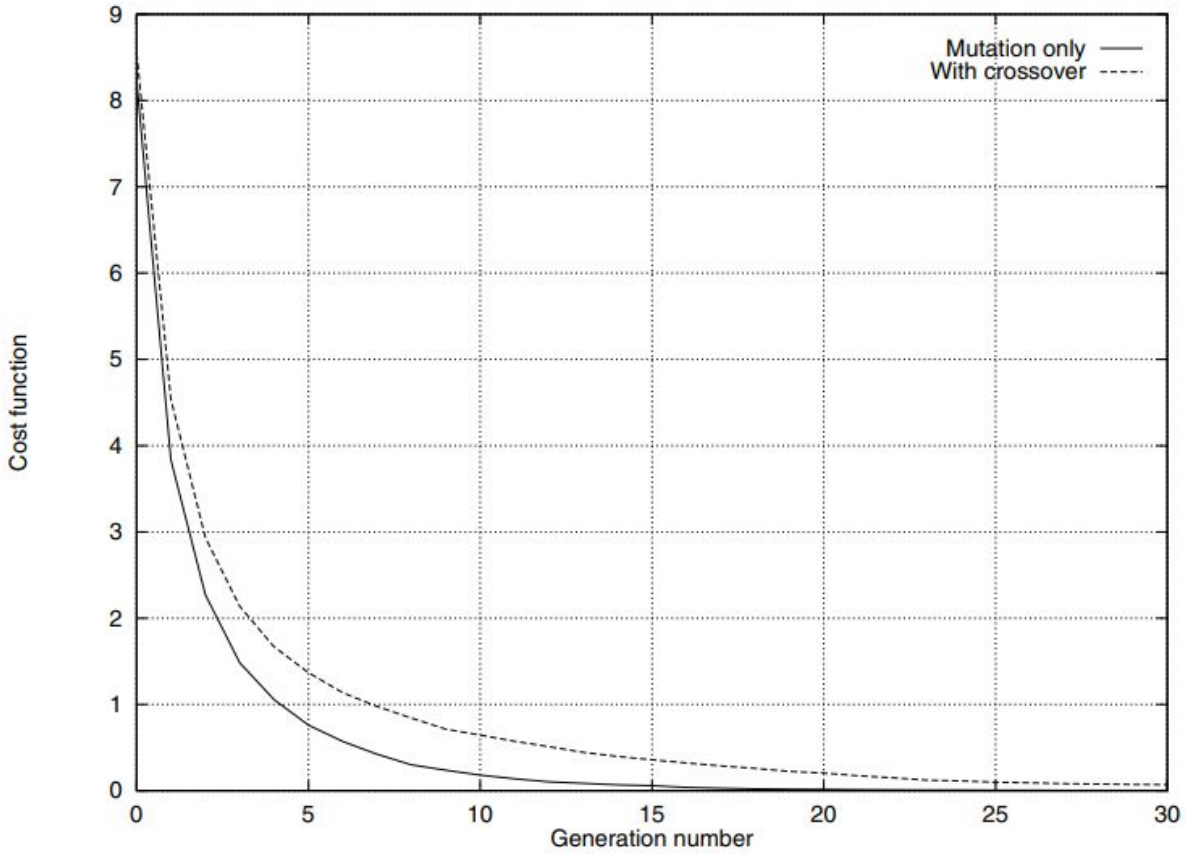
Problem Statement

- Crossover hinders search Cartesian Genetic Programming but not in Linear Genetic Programming, despite their design similarities
- There is very few literature of *why* this happens, only that people are trying to develop better operators
- Trying to develop a solution without understanding the problems is very much putting the cart before the horse!

Clegg et al., 2007



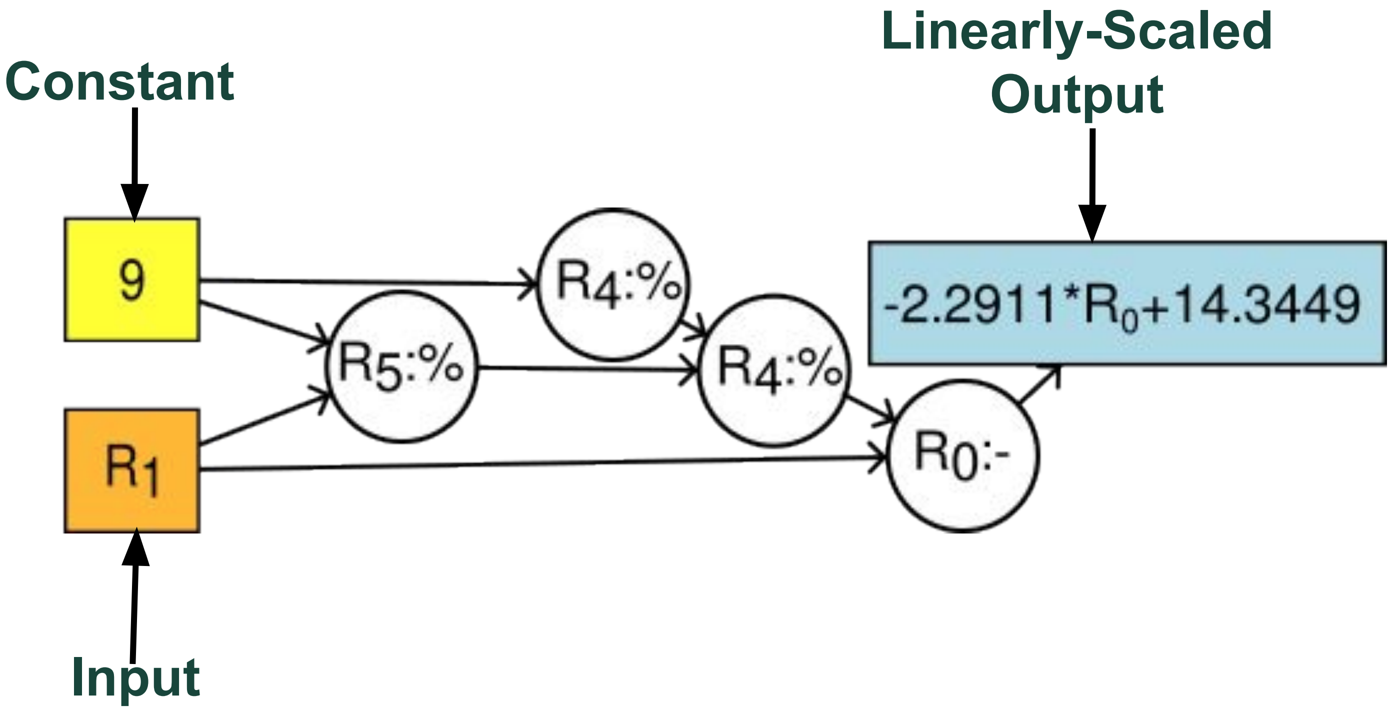
Encoded Representation - LGP

$$p \in P = \begin{bmatrix} d_0 & o_0 & a_{0,1} & a_{0,2} \\ d_1 & o_1 & a_{1,1} & a_{1,2} \\ \dots & \dots & \dots & \dots \\ d_m & o_m & a_{m,1} & a_{m,2} \end{bmatrix}$$

Destination Operator Sources

Xover is done per instruction, not pointwise

$R_5 = 9\%R_1$
$R_4 = R_5\%9$
$R_4 = R_5\%R_4$
$R_0 = R_4\text{-}R_1$



% Indicates the Analytic Quotient
Introns have been removed

Encoded Representation - CGP

$$p = \begin{bmatrix} o_0 & a_{0,1} & a_{0,2} \\ o_1 & a_{1,1} & a_{1,2} \\ \dots & \dots & \dots \\ o_n & a_{n,1} & a_{n,2} \end{bmatrix} \cup [O_0, O_{m-1}]$$

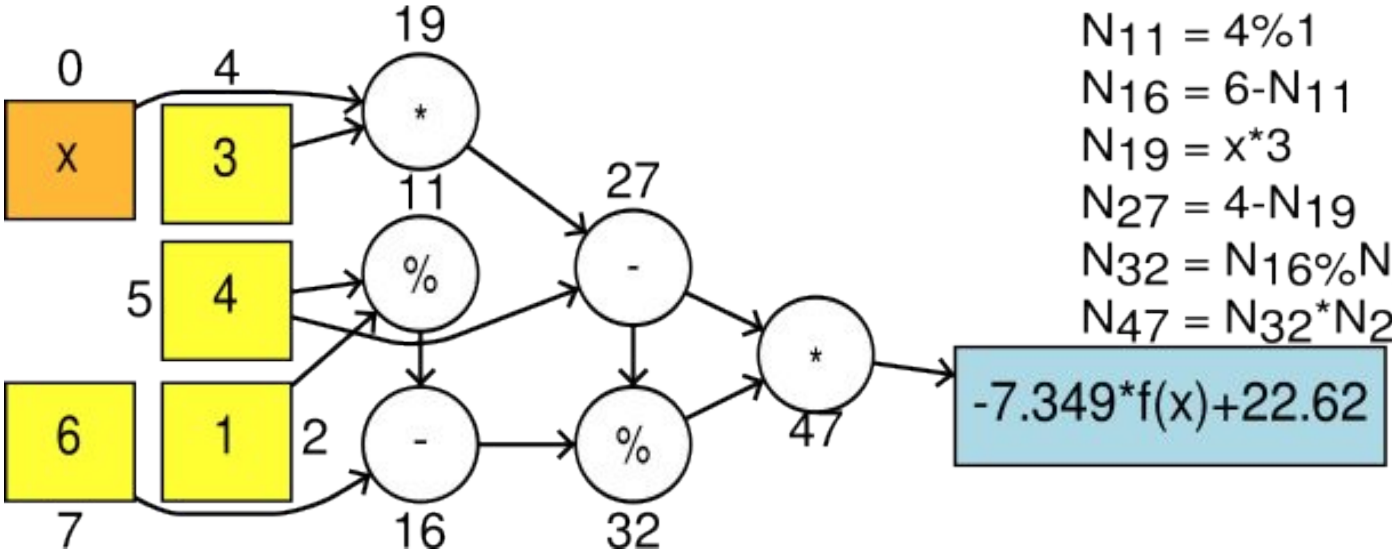
Operator Source Nodes Output

Easier to operate on

Crossover Only:

$$p \in P = o_0 a_{0,1} a_{0,2} o_1 a_{1,1} a_{1,2} \dots o_n a_{n,1} a_{n,2} O_0 \dots O_m$$

Conventional Representation



$N_{11} = 4\%1$
$N_{16} = 6\text{-}N_{11}$
$N_{19} = x*3$
$N_{27} = 4\text{-}N_{19}$
$N_{32} = N_{16}\%N_{27}$
$N_{47} = N_{32}*N_{27}$

$$f(x) = \frac{(4-3x)\left(6-\frac{4}{\sqrt{2}}\right)}{\sqrt{(1+(4-3x)^2}}$$

Experiments

- To demonstrate the crossover problem, we tested eight different CGP and LGP crossover methods
- 10,000 generations in each run
- Tournament size of 4
- Individual max size of 64

Notation	**Xover**	**Mutation**
CGP(1+4)	None	$4(\mu = 100\%)$
CGP(16+64)	None	$4(\mu = 100\%)$
CGP-1x(40+40)	One-Point (50%)	$\mu = 2.50\%$
CGP-2x(40+40)	Two-Point (50%)	$\mu = 2.50\%$
CGP-SGx(40+40)	Subgraph (50%)	$\mu = 2.50\%$
LGP-Ux(40+40)	Uniform (50%)	$\mu = 2.50\%$
LGP-1x(40+40)	One-Point (50%)	$\mu = 2.50\%$
LGP-2x(40+40)	Two-Point (50%)	$\mu = 2.50\%$

Problems

- Example Problems chosen to match Kalkreuth (2020)
- Taken from Koza and Nguyen problem sets
- We used 50 runs per problem

Problem	Function	Domain
Koza-1	$x^4 + x^3 + x^2 + x$	[-1, 1]
Koza-2	$x^5 - 2x^3 + x$	[-1, 1]
Koza-3	$x^6 - 2x^4 + x^2$	[-1, 1]
Nguyen-4	$x^6 + x^5 + x^4 + x^3 + x^2 + x$	[-1, 1]
Nguyen-5	$\sin(x^2)\cos(x) - 1$	[-1, 1]
Nguyen-6	$\sin(x) + \sin(x + x^2)$	[-1, 1]
Nguyen-7	$\ln(x + 1) + \ln(x^2 + 1)$	[0, 2]

Fitness

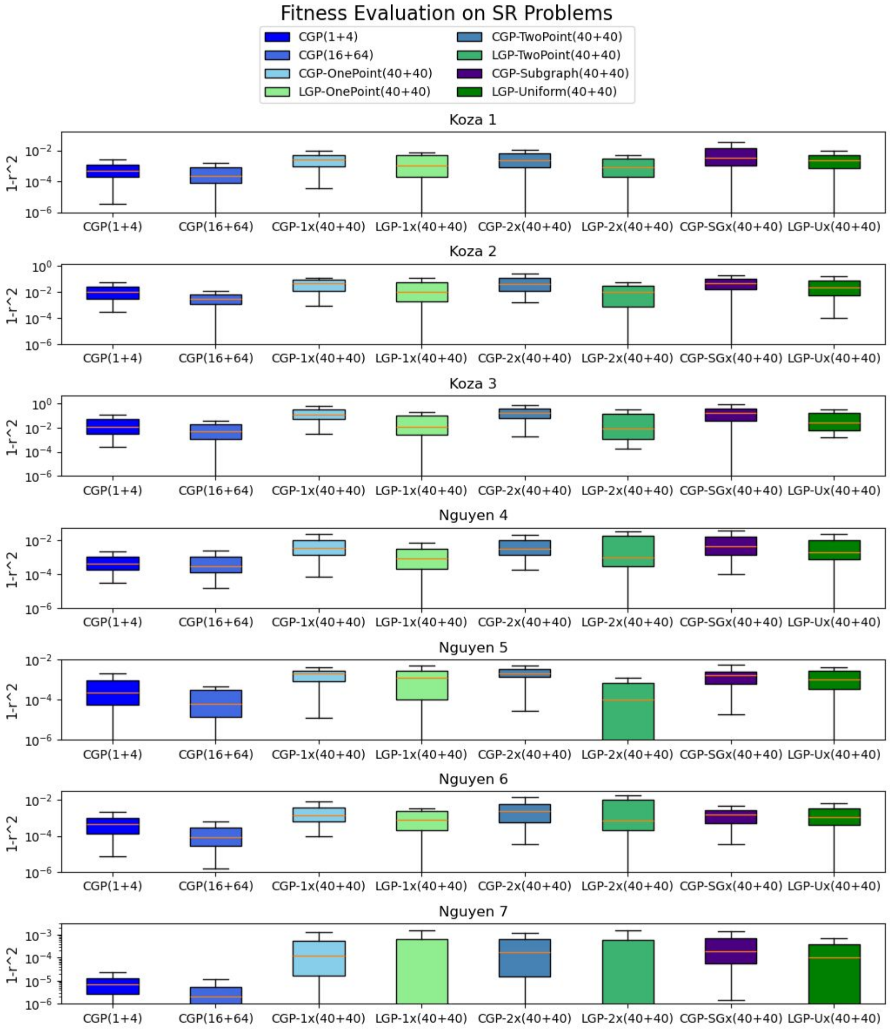
- Instead of RMSE, we use the correlation fitness function described in Haut et al. 2023

$$f_i = 1 - r^2$$

  where r is the pearson correlation

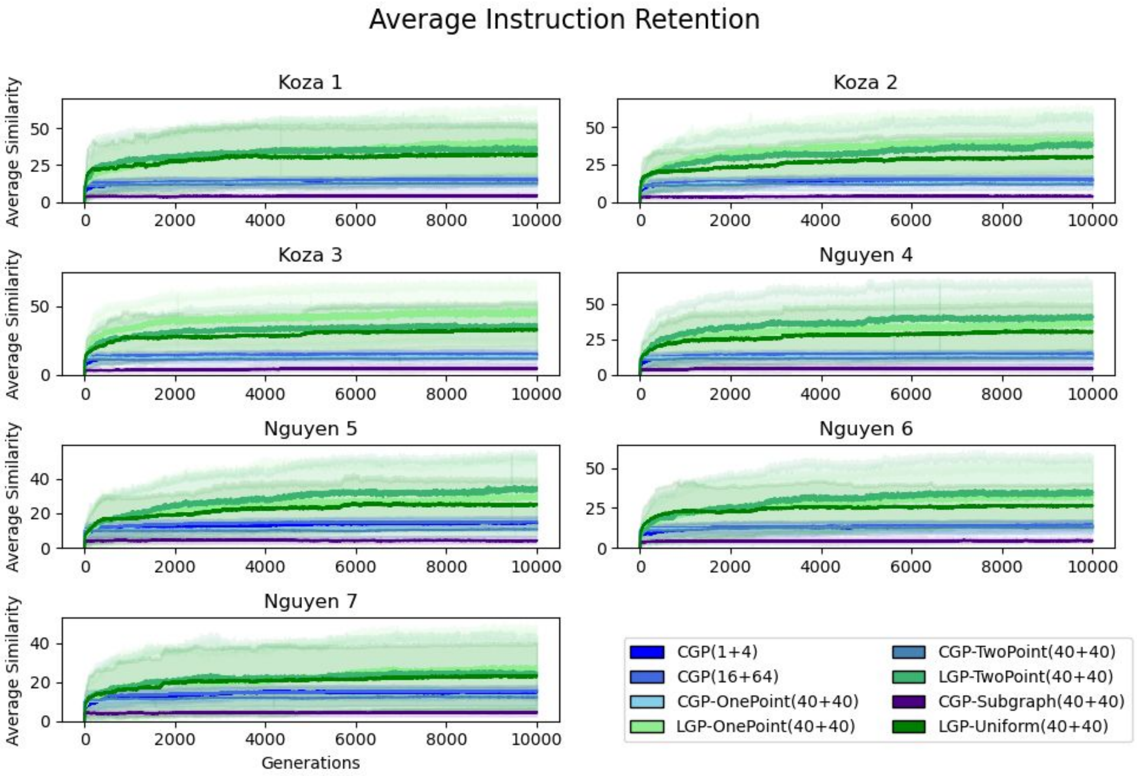
- This metric better accounts for global error

- We see that CGP with crossover is significantly outperformed by their LGP counterparts
- CGP(1+4) and CGP(16+64) perform much better than CGP with crossover
- Thus, we've reproduced the crossover problem



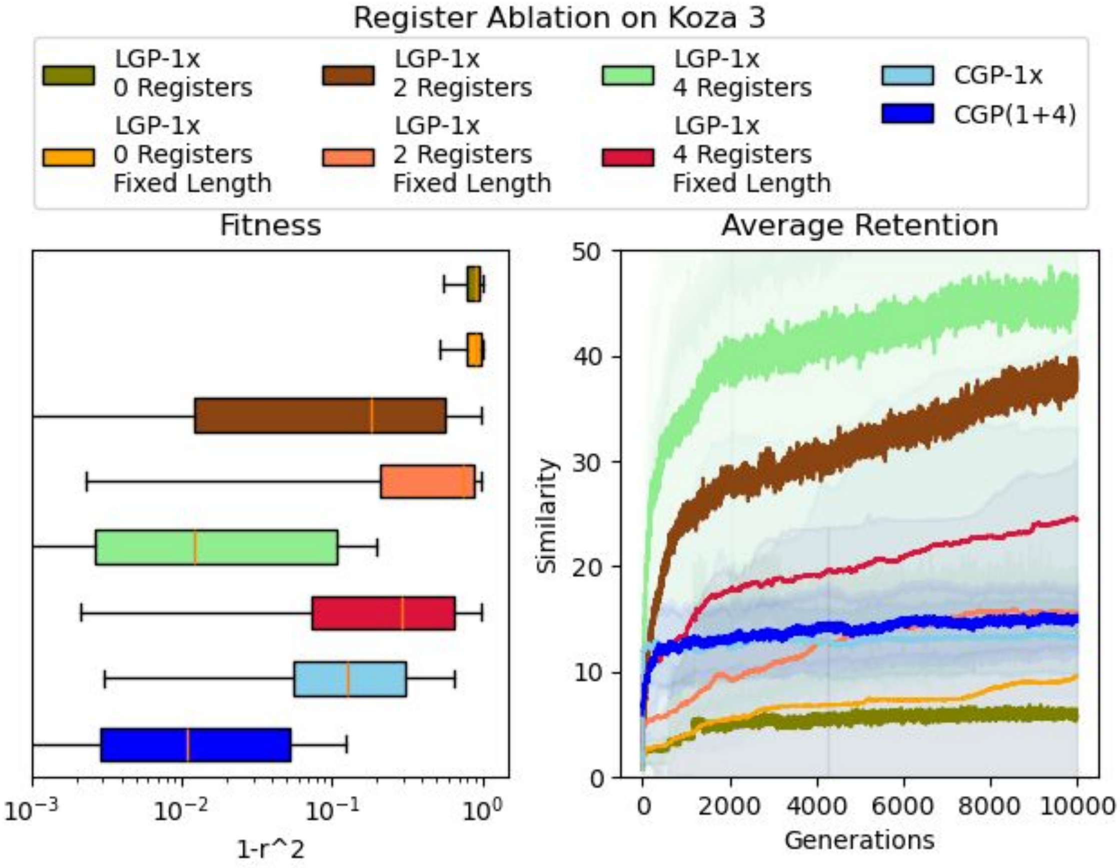
Parent-Child Similarity

- A simple Alignment Scoring algorithm can tell us how similar parents are to their children
- We specifically measure the similarity between the best parent of a couple and their best child



Design Differences

- LGP very distinctively makes use of intermediate calculation registers, and CGP obviously does not
- LGP individuals can be of different sizes, whereas all CGP individuals are fixed to a certain size
- We performed Ablation Testing with different amounts of Registers and alternating the ability to self-regulate program size



Registers are Extremely Important

- Without an adequate number of registers, crossover suffers severe destruction, leading to worse outcomes

WHY?

- We hypothesize that the calculation registers, because they mediate interactions between instructions, serve as anchor points for high-fitness substructures
- CGP has no such luxury, as changing a single connection gene can destroy the existing graph trace

Size Regulation is also Important

- It is also advantageous to allow LGP individuals to change sizes as evolution progresses

WHY?

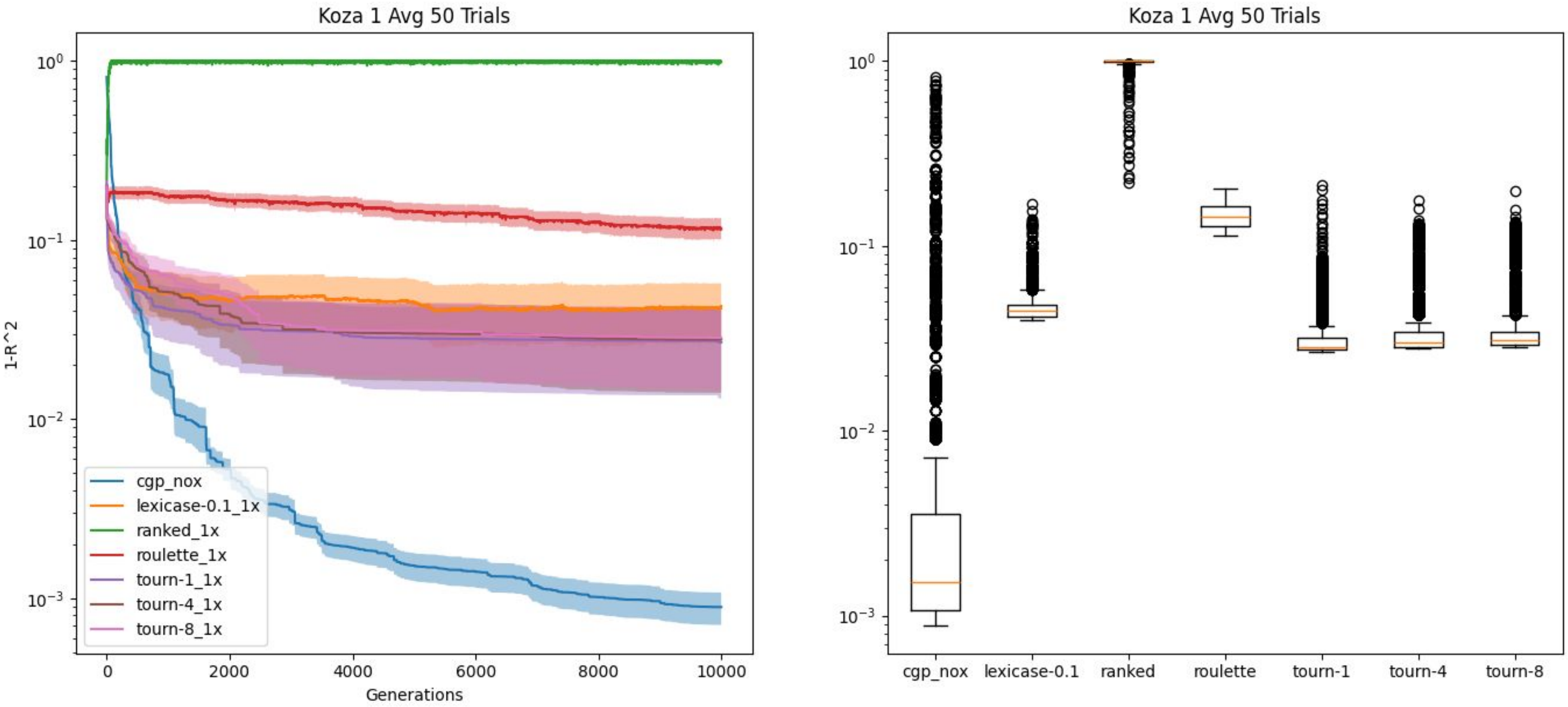
- Following Banzhaf and Bakurov (2024), we postulate that this allows material to be exchanged with a smaller risk of amputating existing high-fitness substructures
- This might be because the crossover points can be at different indices, whereas in CGP both parents would use the same index

CGP is thus not robust to destruction caused recombination by the considerations (or lack thereof) **of its design!**

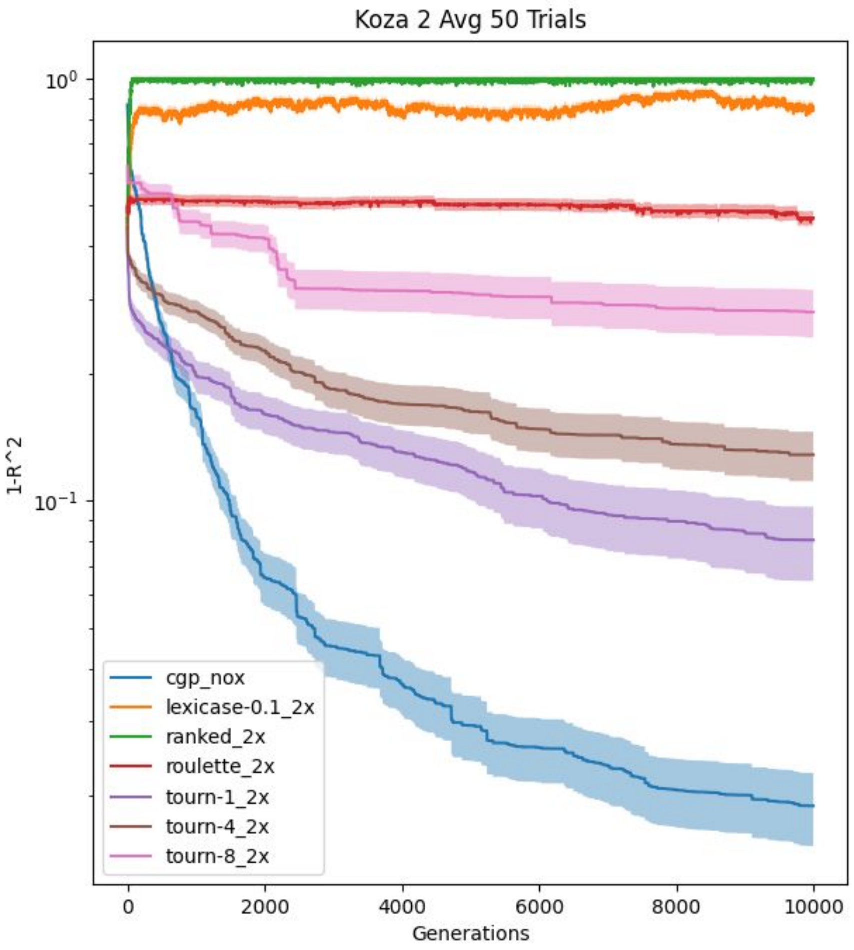
But maybe it's just a bad choice of selection methods… After all, bad children should be selected out, right?

Selection

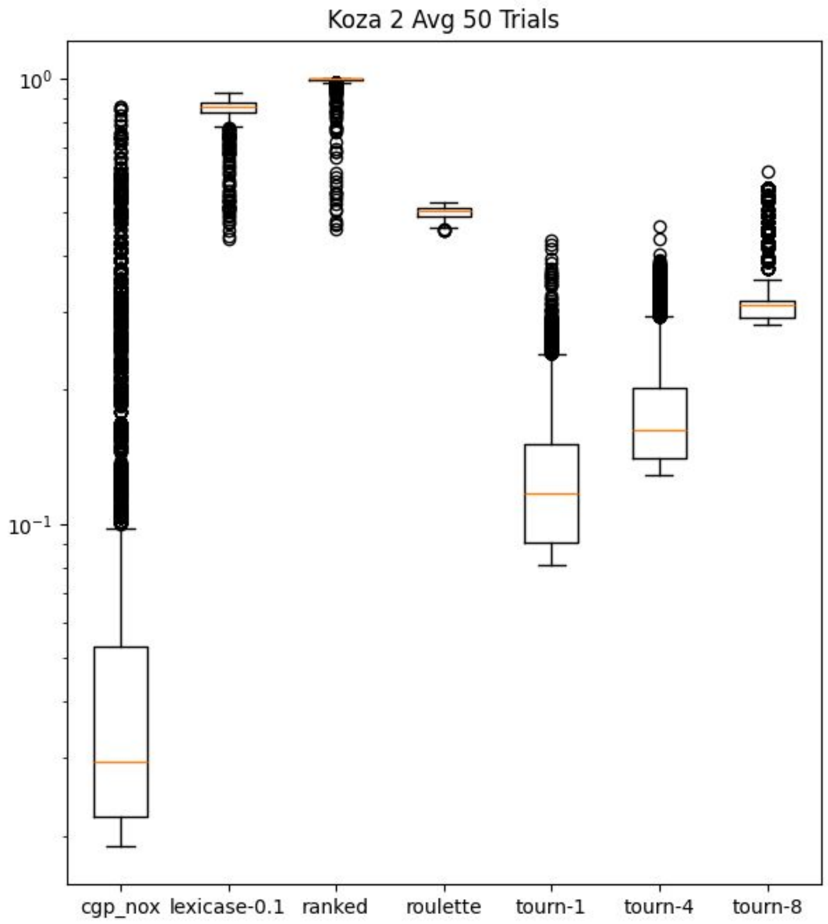
- To rule out alternative explanations, we examine the effect of different selection methods on CGP1X and CGP2X performance
- Methods used:
    - Tournament (n = 1, 4, and 8)
    - Lexicase (e = 0.1)
    - Roulette Wheel
    - Ranked
- Methods are compared to CGP(1+4)
    - No crossover



Koza 1 Avg 50 Trials
1-R^2
Generations
cgp_nox
lexicase-0.1_1x
ranked_1x
roulette_1x
tourn-1_1x
tourn-4_1x
tourn-8_1x
Koza 1 Avg 50 Trials
cgp_nox
lexicase-0.1
ranked
roulette
tourn-1
tourn-4
tourn-8



Koza 2 Avg 50 Trials
1-R^2
10^0
10^{-1}
0
2000
4000
6000
8000
10000
Generations
cgp_nox
lexicase-0.1_2x
ranked_2x
roulette_2x
tourn-1_2x
tourn-4_2x
tourn-8_2x



Koza 2 Avg 50 Trials
10^0
10^{-1}
cgp_nox
lexicase-0.1
ranked
roulette
tourn-1
tourn-4
tourn-8

Objective:

Goal: Crossover has a destructive effect on CGP but not LGP, although both algorithms have a similar structure.

Contribution: seeking to determine if Mutation operators have an effect on this phenomenon.

Implemented Algorithms:

- LGP_1X
- CGP, CGP_1X, CGP_2X

Mutation Operators:

- Basic Mutation
- Adaptive Mutation
- Swap Mutation

Functions:

Koza-1: $x^4 + x^3 + x^2 + x$ [-1, 1]

Koza-2: $x^5 -2 x^3 + x$ [-1, 1]

Koza-3: $x^6 -2 x^4 + x^2$ [-1, 1]

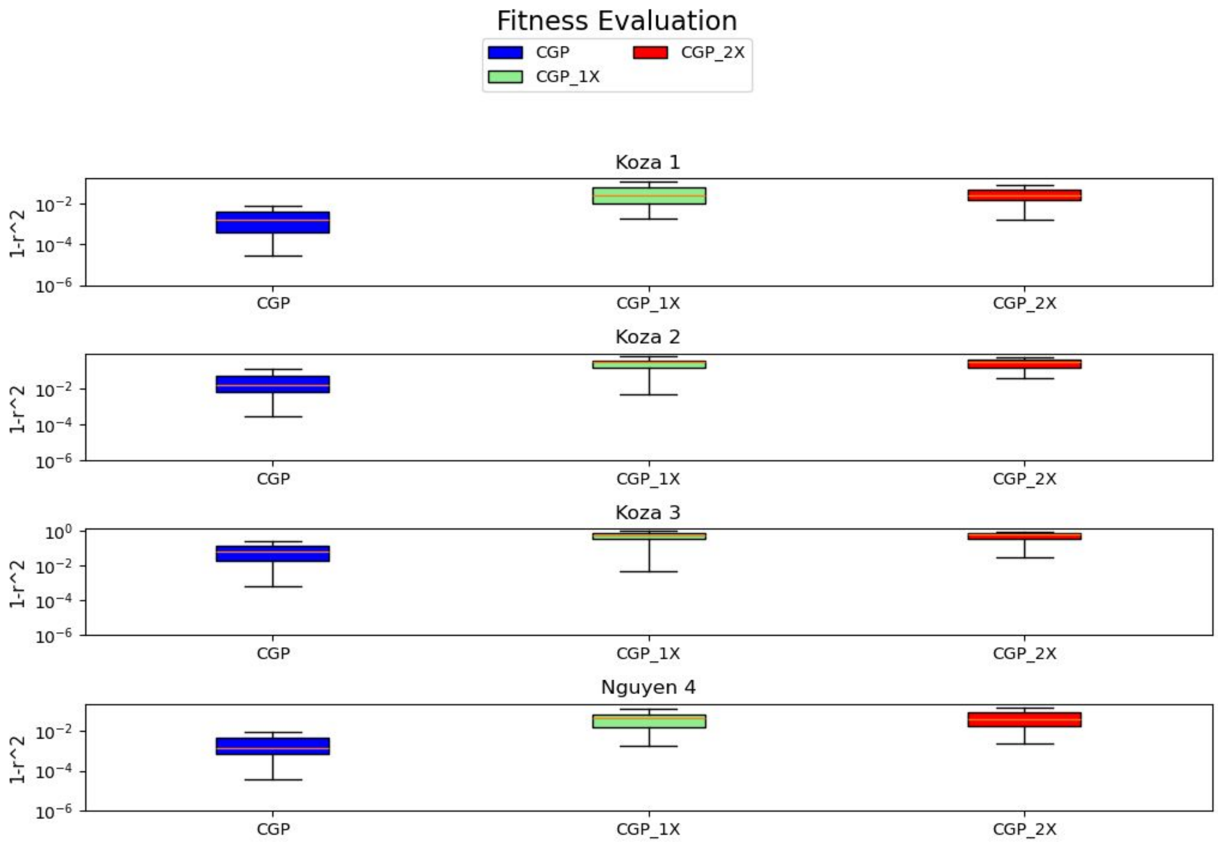
Nguyen-4: $x^6 + x^5 + x^4 + x^3 + x^2 + x$ [-1, 1]

Nguyen-5: $\sin(x^2) \cos(x) - 1$ [-1, 1]

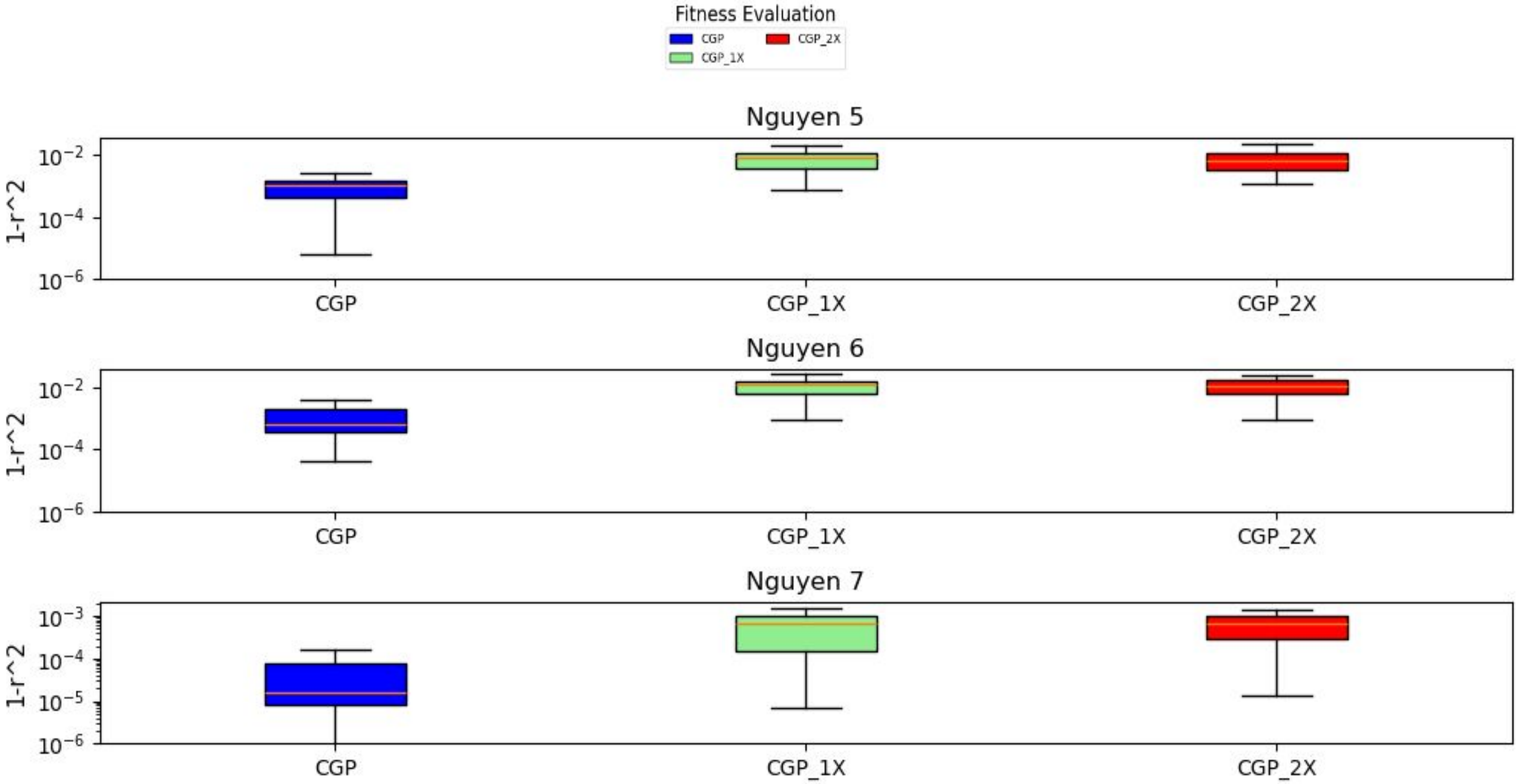
Nguyen-6: $\sin(x) +\sin(x + x^2)$ [-1, 1]

Nguyen-7: $\ln(x+1) + \mathrm{len}(x^2 + 1)$ [0, 2]

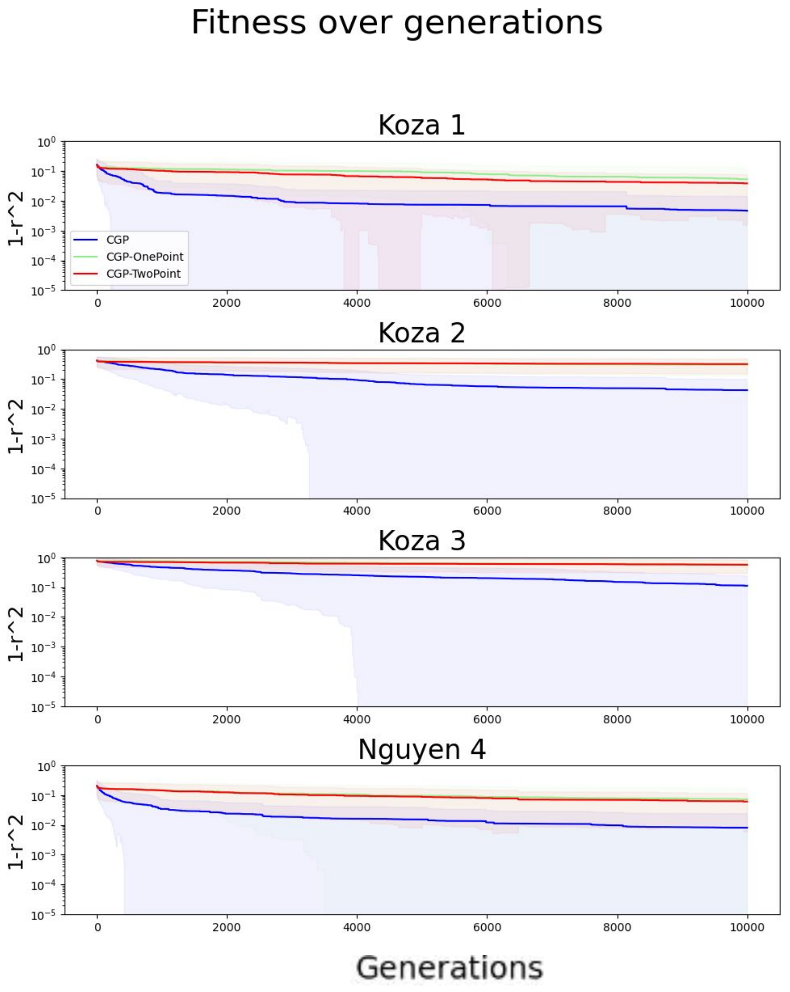
Generations: 10000, Trials= 50



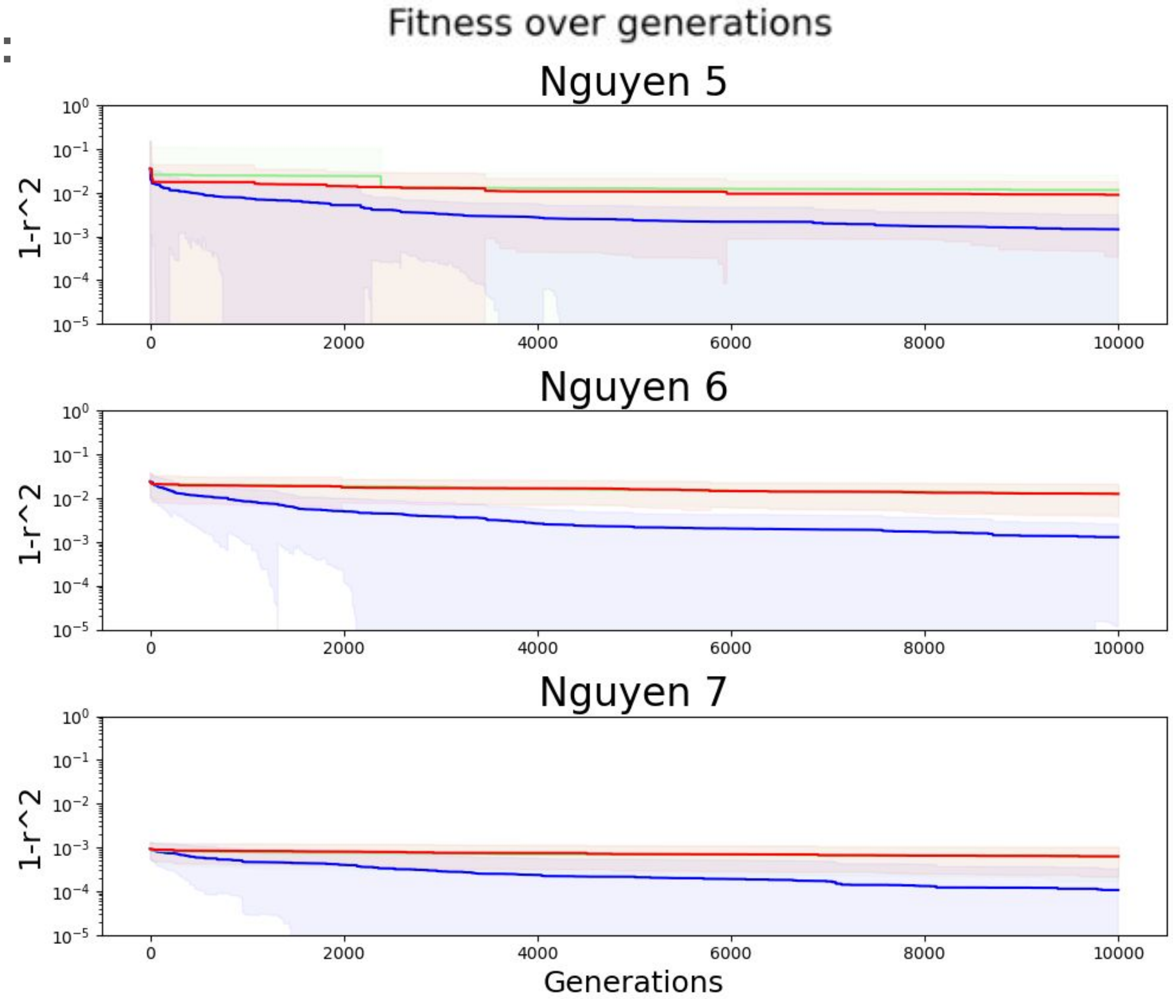
Basic Mutation(2): Generations: 10000, Trials= 50



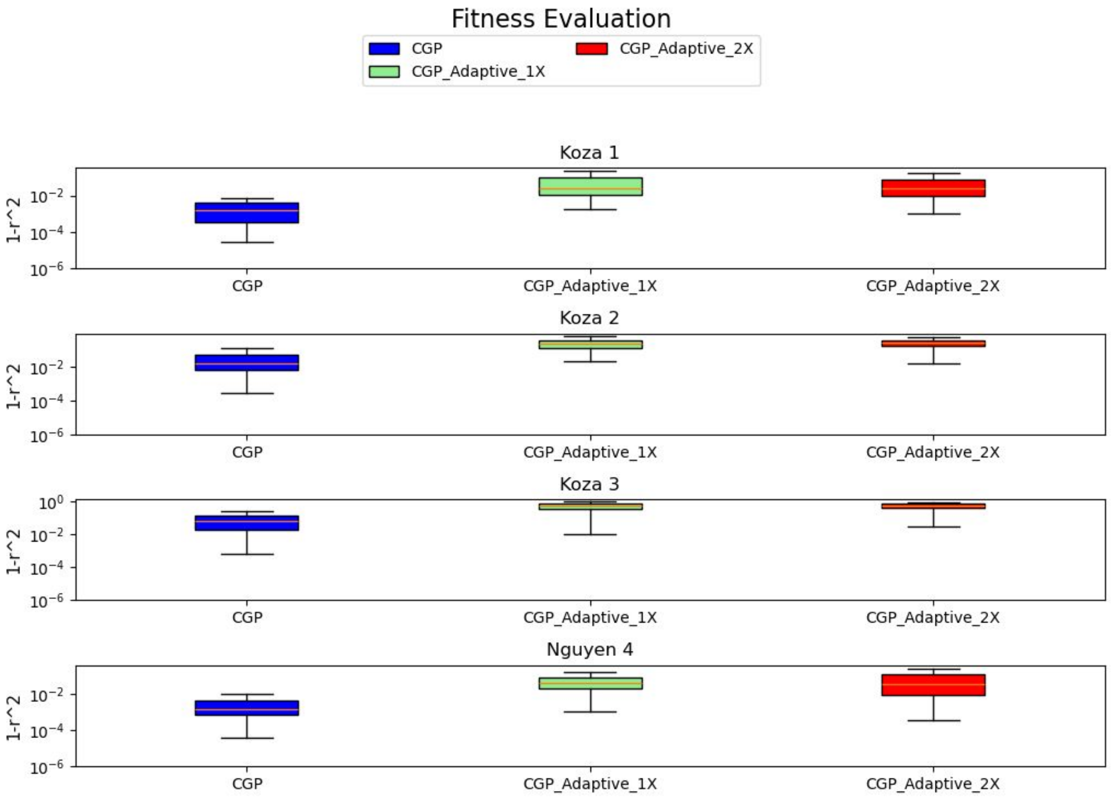
Basic Mutation(1):



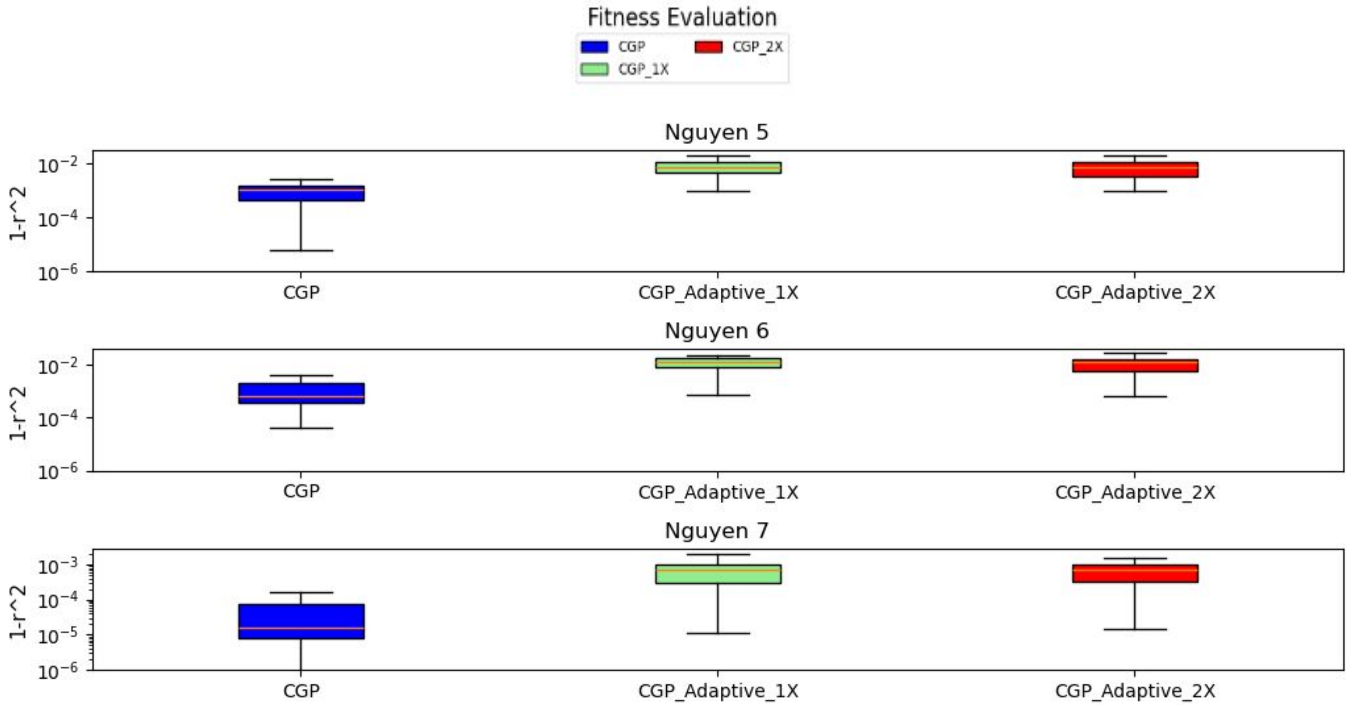
Basic Mutation(2):



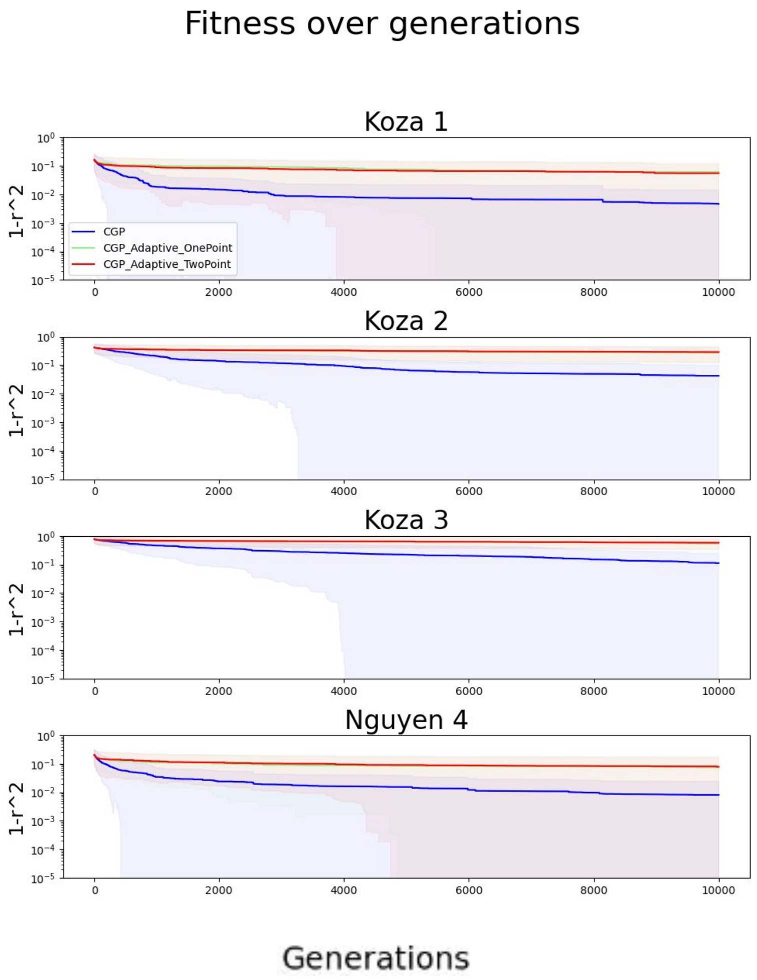
Adaptive Mutation(1): Generations: 10000, Trials= 50



Adaptive Mutation(2): Generations: 10000, Trials= 50

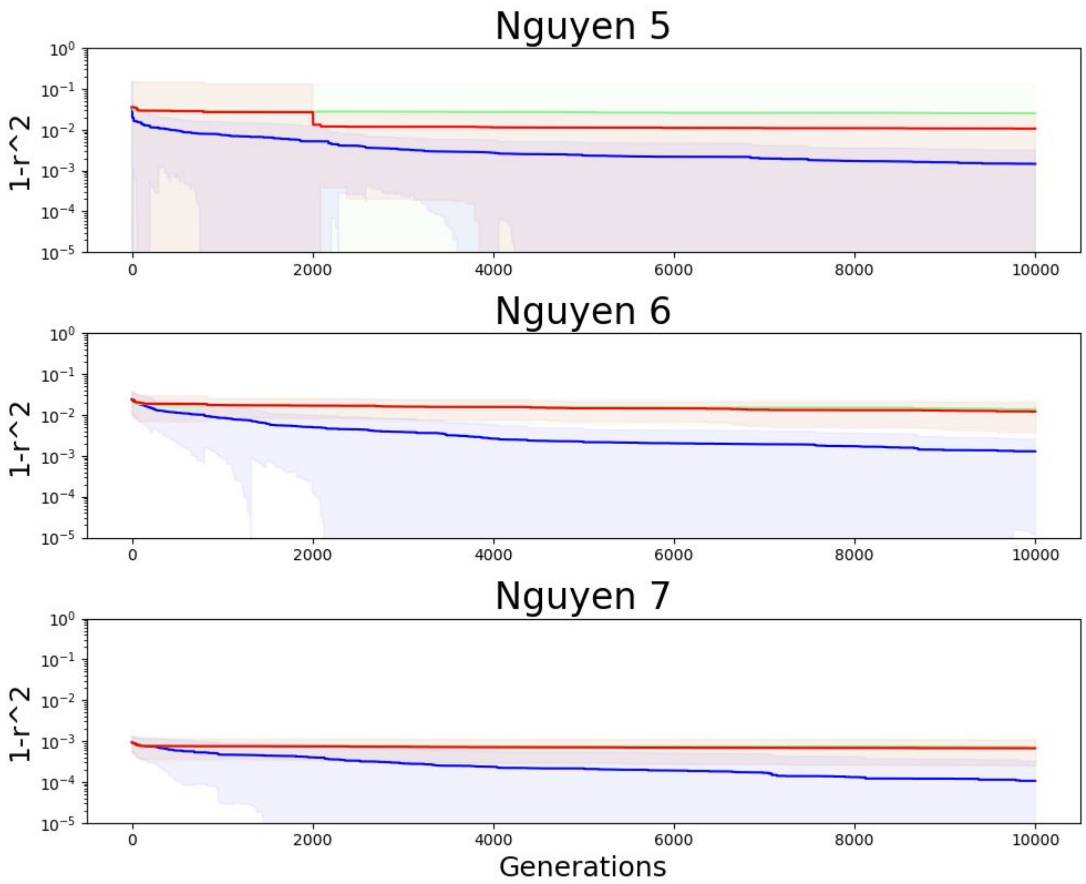


Adaptive Mutation(1):

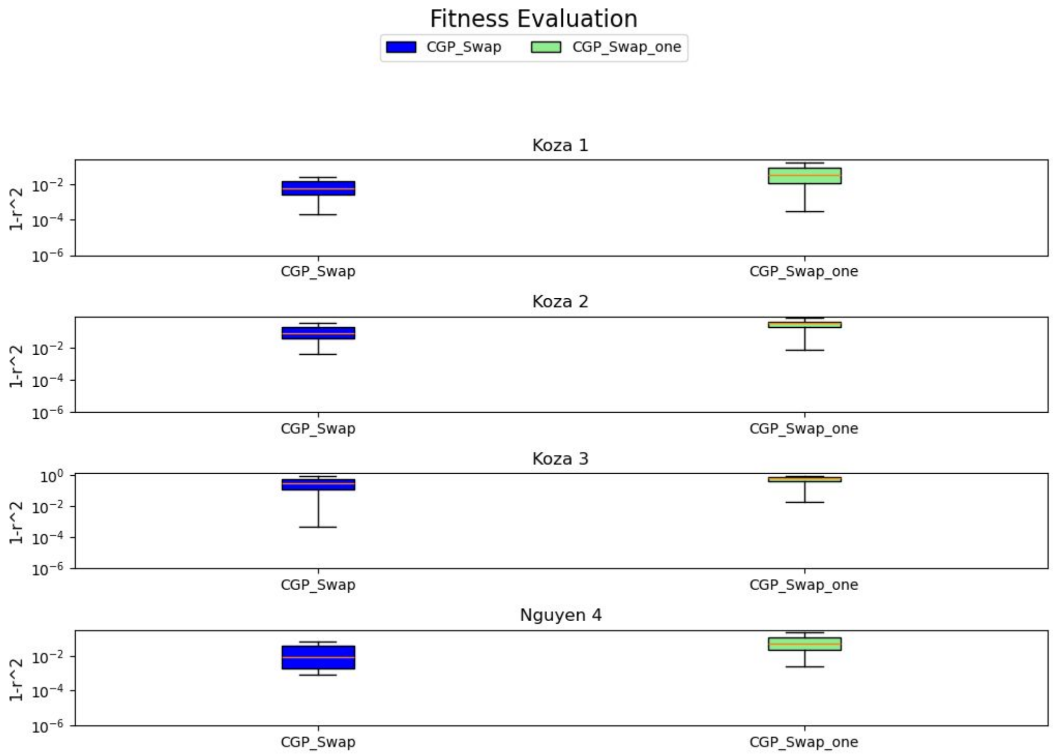


Adaptive Mutation(2):

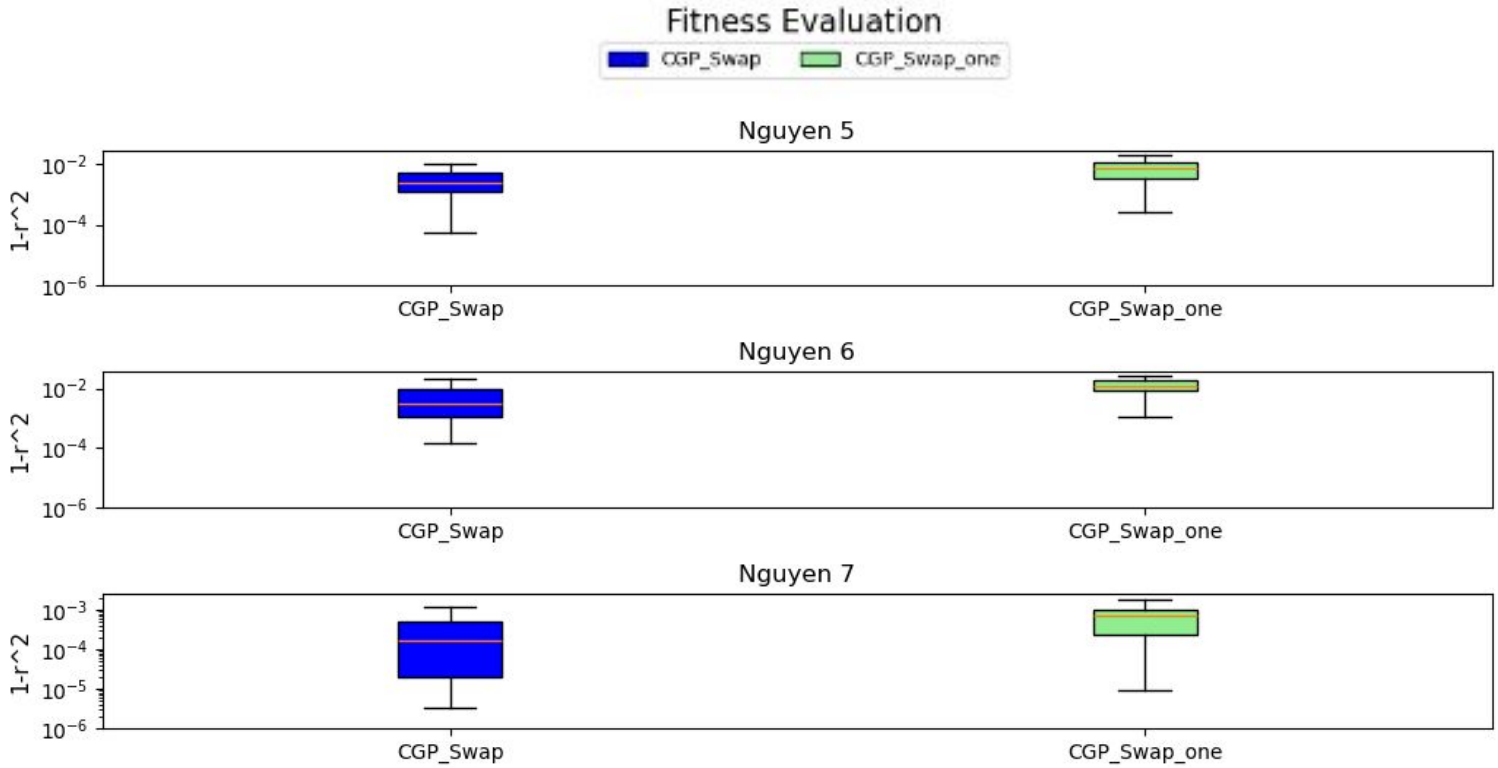
Fitness over generations



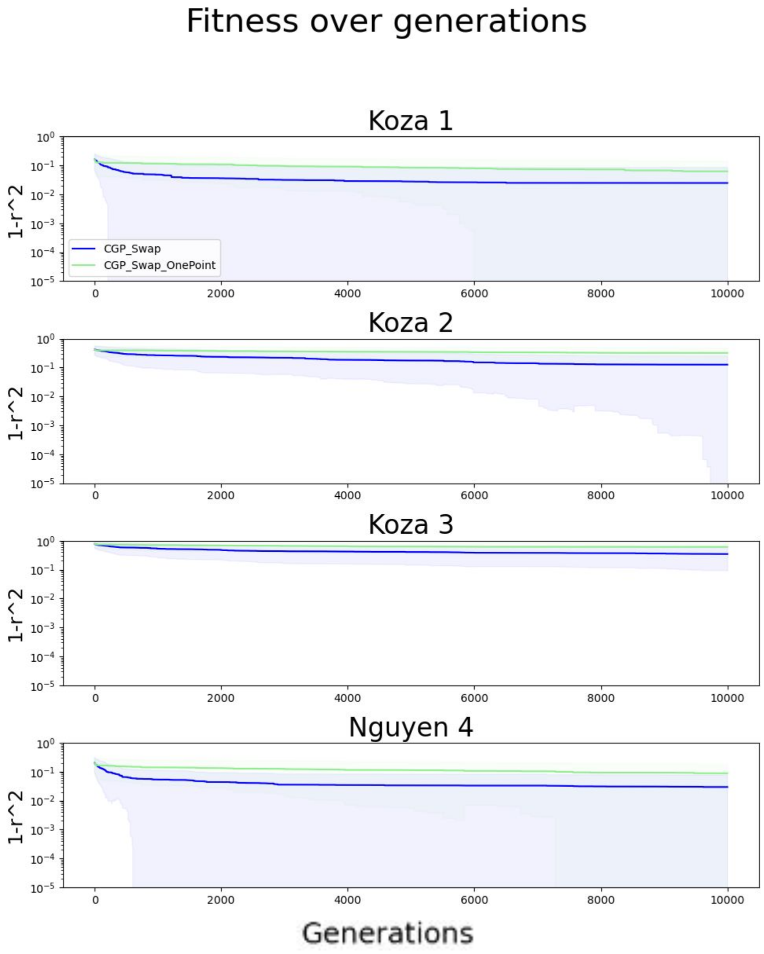
Swap Mutation(1)   Generations: 10000, Trials= 50



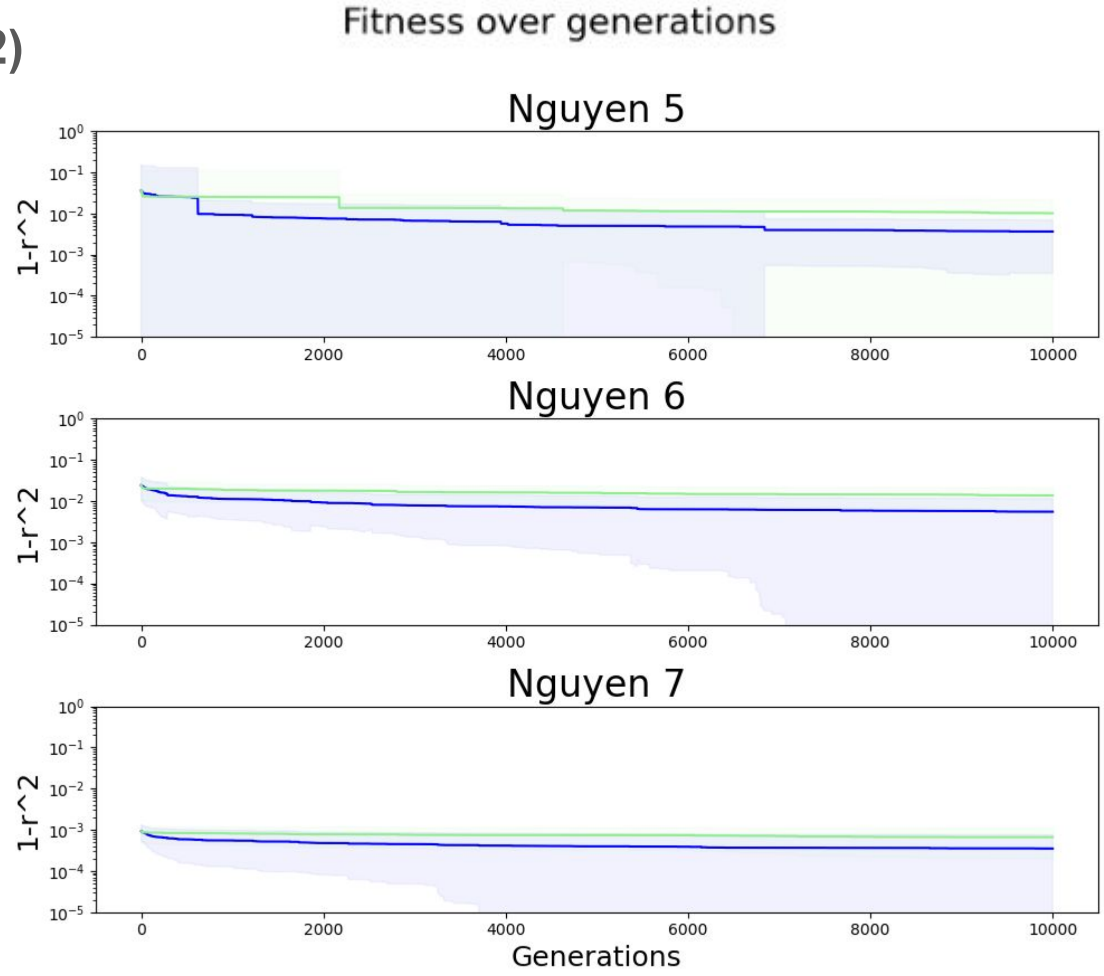
Swap Mutation(2)   Generations: 10000, Trials= 50



Swap Mutation(1)



Swap Mutation(2)



Future Goals

- We hope to compile a journal paper for submission this summer!
- There are still new questions to ask:
  - Why don’t bad children get filtered out during selection?
  - Are mutations really producing offspring that are so poor?
  - Are there guided mutations that can ameliorate crossover destruction?
  - Can we create introduce elements from LGP to CGP to facilitate crossover?