LONDON'S GLOBAL UNIVERSITY

UCL

Population Diversity, Information Theory and Genetic Improvement

GP experiments in optimising Triangle.c

Bill Langdon and David Clark

EuroGP 2025

What happens if we add diversity into the fitness function?

triangle.c

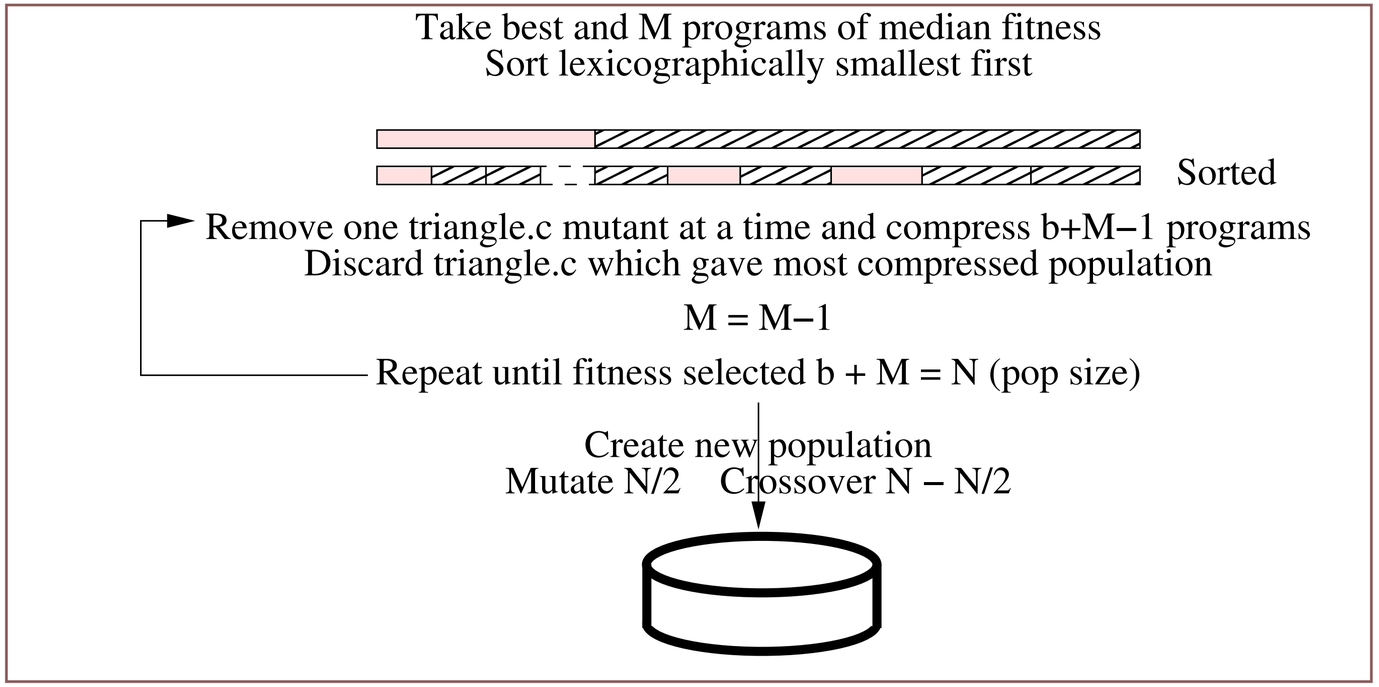
The software engineering triangle benchmark takes three inputs and returns one of four integer values representing the type of the triangle: scalene, isosceles, equilateral or not a triangle

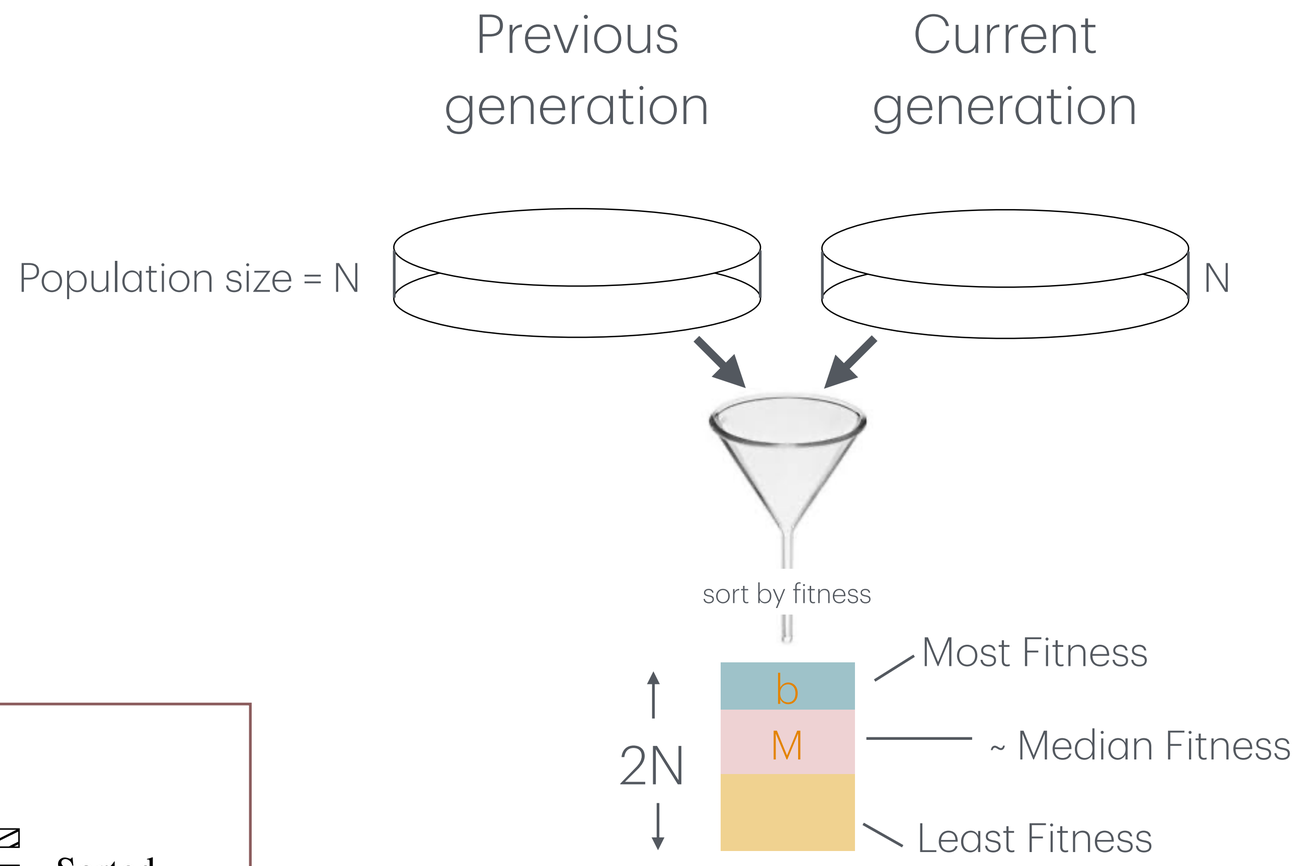
https://github.com/wblangdon/triangle/blob/master/jss/triangle.c

Faster triangle.c

Representation:	C code converted to XML by srcml. Variable length linear sequence of XML mutations. Mutated XML converted to C code and compiled.
Fitness cases:	14 test cases, each 3 sides of triangle and expected classification. Test suite designed to cover original C code. Test suite weighting to favour important outputs: scalene and equilateral (one test each) weight 81, isosceles (three tests) weight 27, not a triangle (nine tests) weight 1, (Section 3).
Selection:	Fitness is the sum of the number of instructions taken by each test multiplied by its weighting $\text{fitness} = \sum_{i=1}^{14} \text{X86 instructions for test } i \times \text{weight } i$ If mutant fails to compile, fails at run time, exceeds 2 second time out or gives wrong answer on any test its fitness is so bad it will never have children. 1^{st} fitness based rank selection and 2^{nd} contribution to population diversity, see Figure 1 and Section 2.2.
Population:	Panmictic, non-elitist, generational, size 1, 2, 5, 10 $\cdots$ 1000.
initial pop	Every triangle.c is mutated exactly once. All compile and run (page 6 item 2.). Initial fitness 7929–12578 (most as unmutated code 9069).
Parameters	
Magpie:	Python version 3.10.1, GGC version 10.2.1, compiler options -O3 -DNDEBUG. Magpie defaults except [search] warmup=1. XML edits: StmtReplacement StmtInsertion StmtDeletion ComparisonOperatorSetting ArithmeticOperatorSetting NumericSetting RelativeNumericSetting StmtMoving
GP :	50% subtree XML crossover, 50% subtree XML mutation (Section 4.1). 100 generations. No size limit.

How we use Algorithmic Information in the context of GP

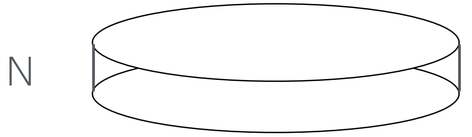




Discard D programs from best and median fitness

S = M - D so that

Next generation size is N = b + S



But what is the compression about?

Vitanyi

$K(x|y)$: The conditional Kolmogorov complexity

The conditional Kolmogorov complexity of a string of symbols, x, given another string, y, is the length of the shortest program that outputs x, given y as input.

$K(x) \stackrel{\triangle}{=} K(x|\epsilon)$. Not computable.

ID: The Information Distance

For two strings x and y,

$$ID(x, y) = \max\{K(x|y), K(y|x)\}$$

Universal and Generic

NID: The Normalised Information Distance

For two strings x and y,

$$\mathrm{NID}(x,y) = \frac{\max\{K(x|y), K(y|x)\}}{\max\{K(x), K(y)\}}$$

Cilibrasi and Vitanyi

Enables comparisons between strings of different lengths

NCD: The Normalised Compression Distance

For two strings x and y,

$$\mathrm{NCD}(x,y) = \frac{C(xy) - \min\{C(x), C(y)\}}{\max\{C(x), C(y)\}}$$

Computable approximation using compressors such as 7zip, Bzip

Extension to Multi-Sets

NCD: The Normalised Compression Distance for multisets

For a multiset X,

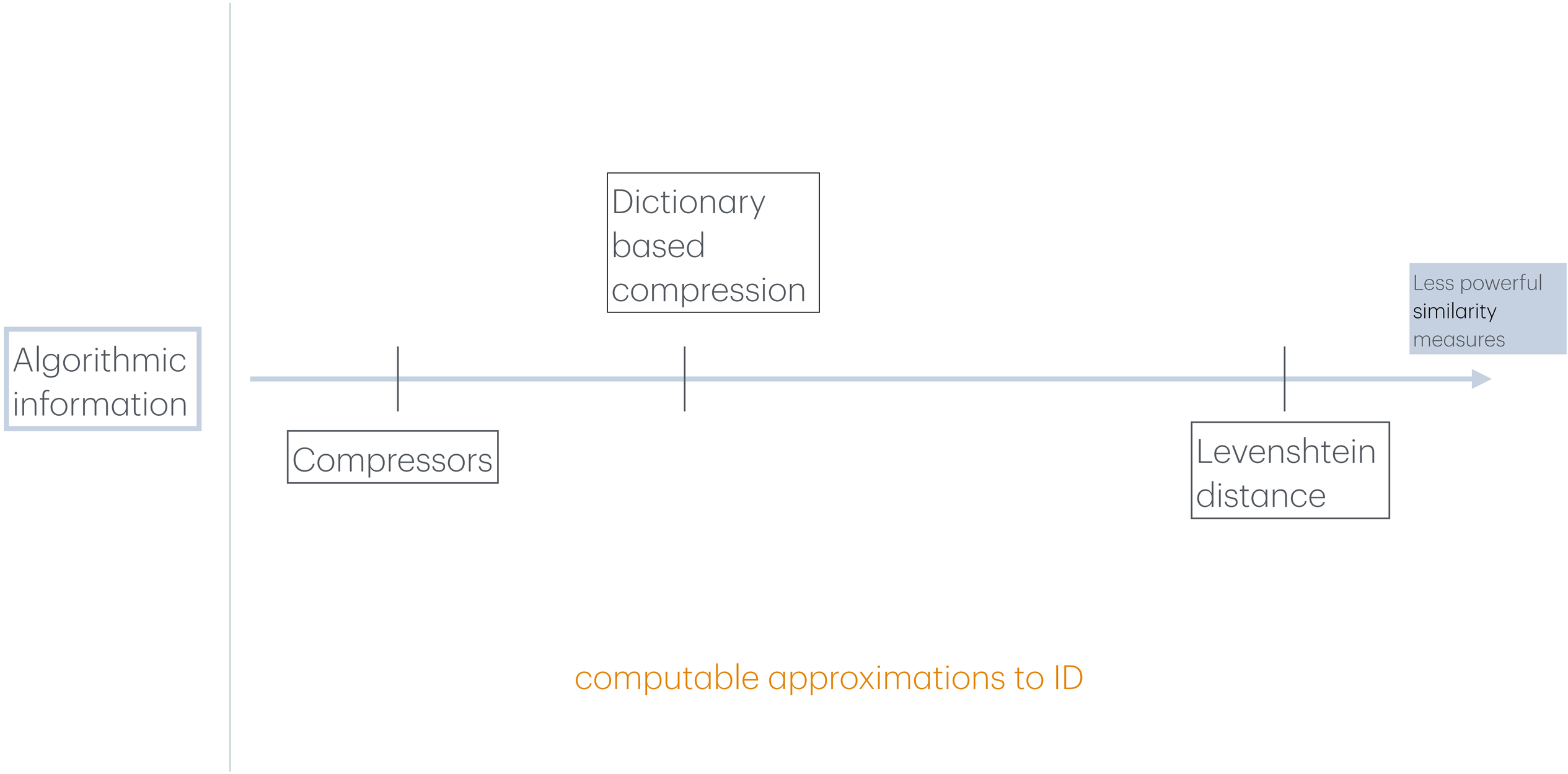
$$\mathrm{NCD}_1(X) = \frac{C(X) - \min_{x \in X}\{C(x)\}}{\max_{x \in X}\{C(X \setminus \{x\})\}} \tag{1}$$

$$\mathrm{NCD}(X) = \max\left\{\mathrm{NCD}_1(X), \max_{Y \subset X}\{\mathrm{NCD}(Y)\}\right\} \tag{2}$$

Time complexity $\mathcal{O}(2^{|X|})$

Cohen and Vitanyi

Population Diameter


Algorithmic information
Compressors
Dictionary based compression
Levenshtein distance
Less powerful similarity measures
computable approximations to ID

Approximate NCD for Multisets (NCDm)

The algorithm starts from the multiset $Y_0 = X = \{x_1, x_2, \ldots, x_n\}$, and proceeds as:

1. Find index i that maximizes $C(Y_k \setminus \{x_i\})$.
2. Let $Y_{k+1} = Y_k \setminus x_i$.
3. Repeat from step 1 until the subset contains only two strings.
4. Calculate $\mathrm{NCD}(X)$ as: $\max_{0 \leq k \leq n-2}\{\mathrm{NCD}_1(Y_k)\}$.

Time complexity $\mathcal{O}|X|^2)$

We compare three approximations

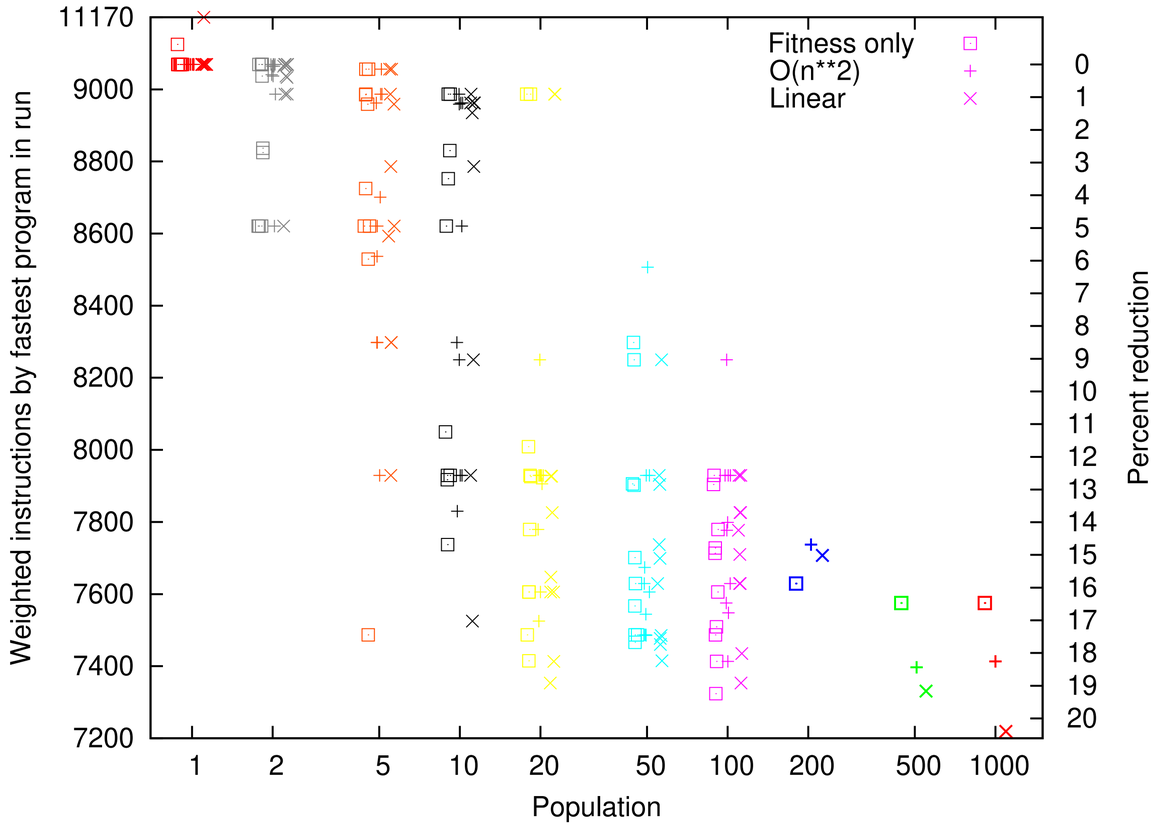
- NCDm (**quadratic** in the population size)
- A **linear** approximation that simply removes strings based on smallest size in each step
- Random selection of victims

$$(\text{plotted with } +,\ \times \text{ or } \square)$$

The number of programs of average fitness is 10% of population on average

Results

Speed up

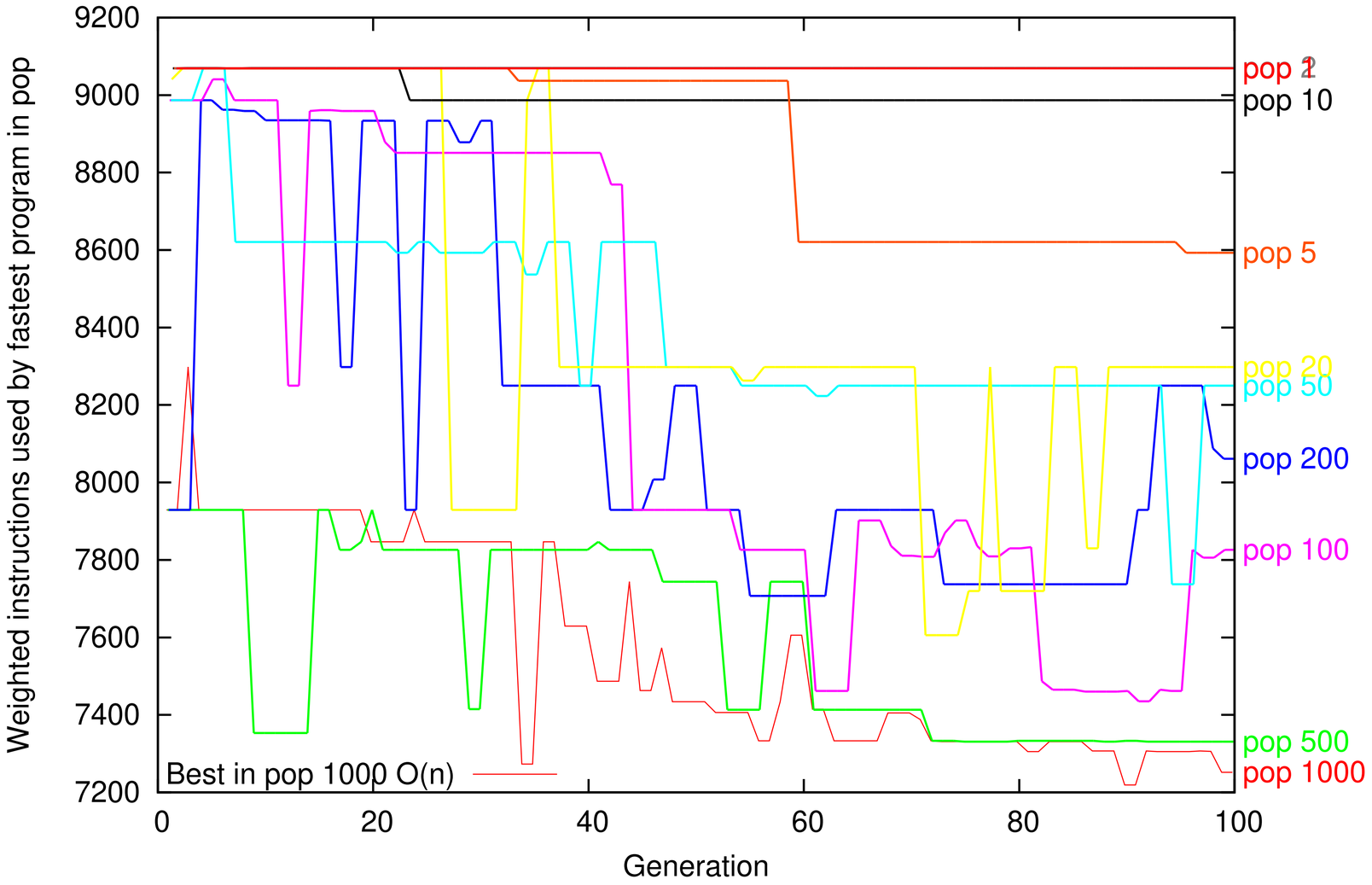


Median improvement 16%

Fastest triangle.c mutant found in ten runs by generation 100 with pop size 1, 2, 5,· · · up to 1000 (only 1 run 200–1000). Select best from current and previous generations to be parents. (One linear × run with population 1 got stuck at fitness 11 170.) Small horizontal noise added to spread data.

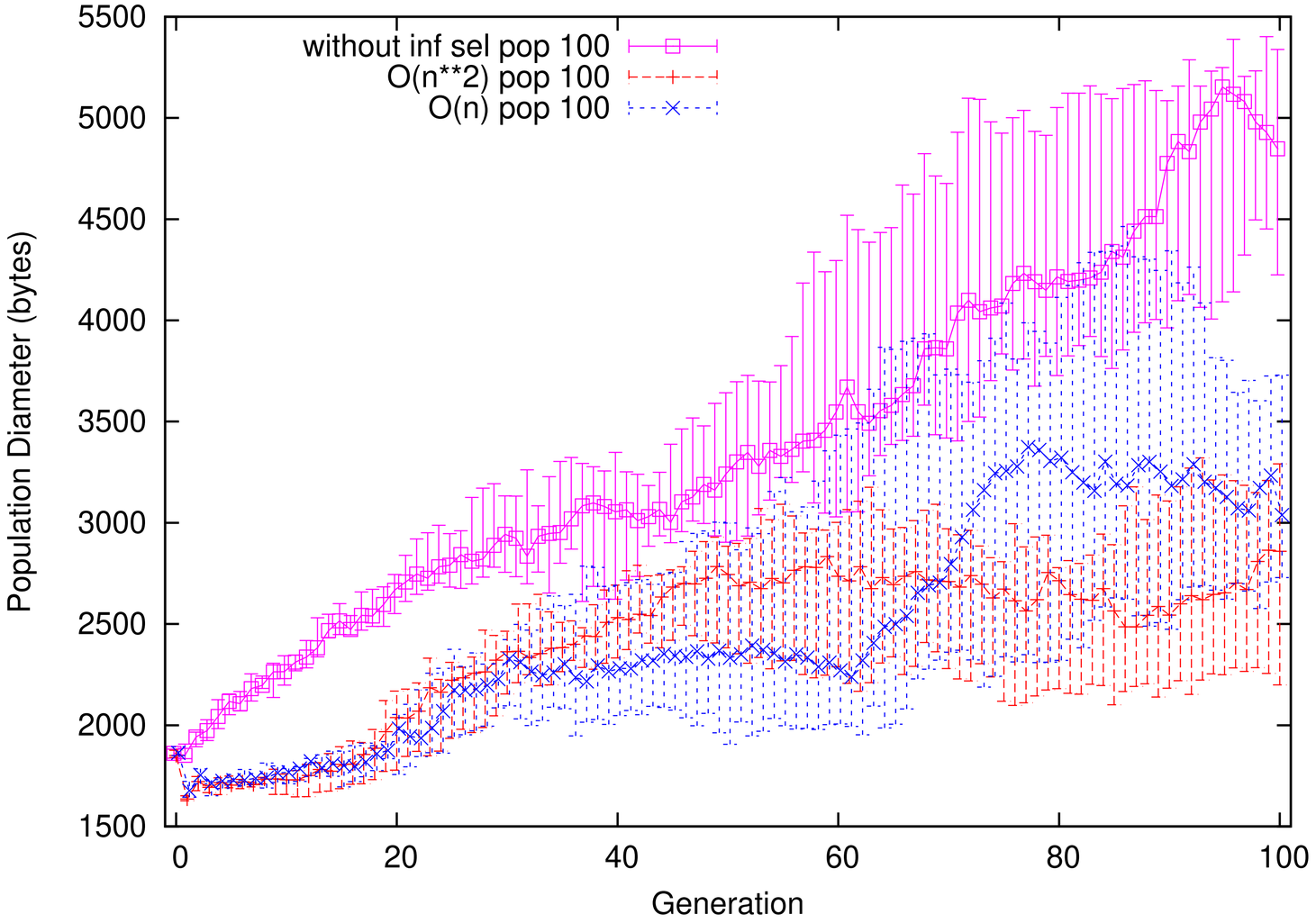
Results

Fitness evolution



Results

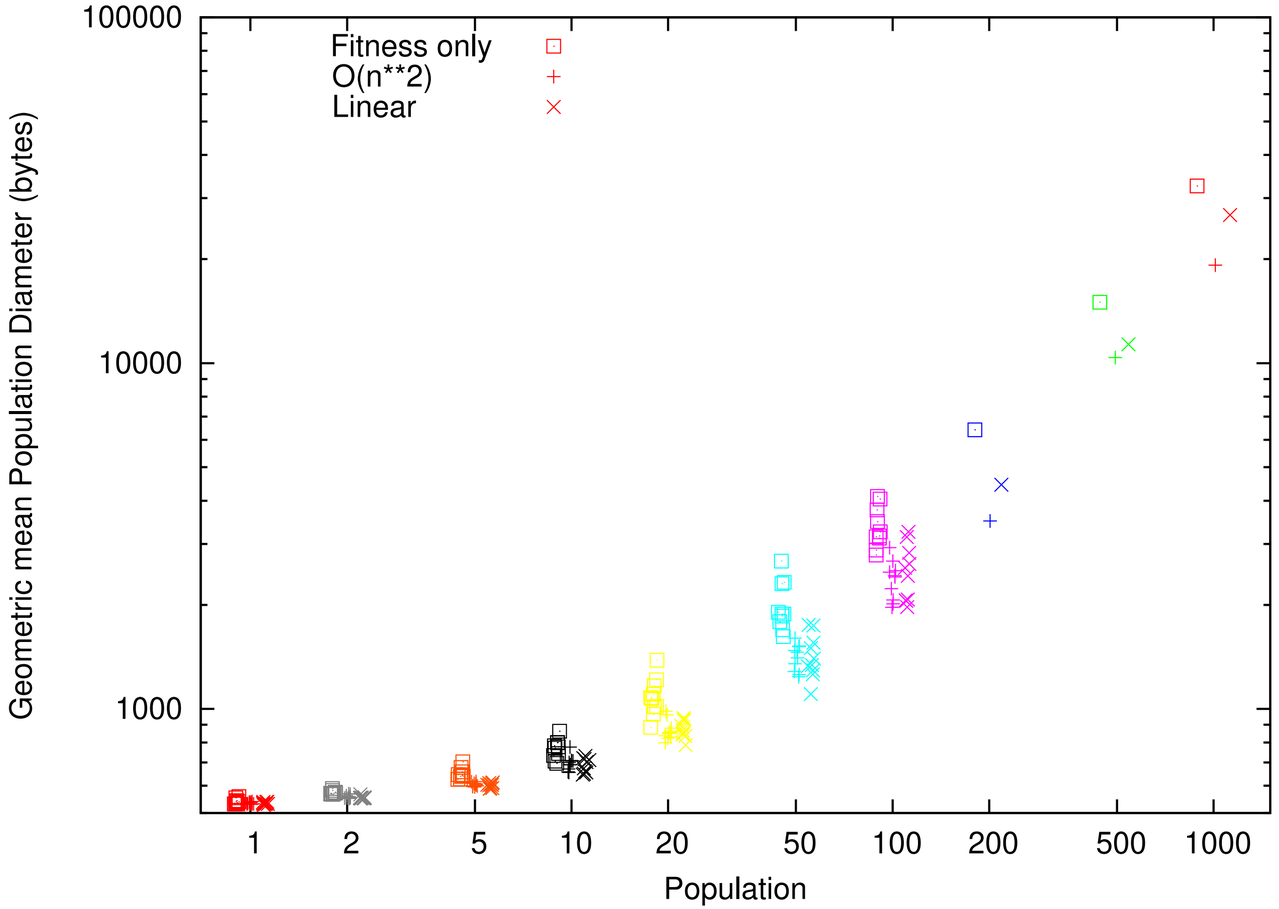
Population Diameter evolution



Evolution of median population diameter for ten runs with the three selection schemes. GI populations of 100. The error bars give the interquartile spread across ten runs.

Results

Mean population diameter



Average compressed population of triangle.c mutants in ten runs up to generation 100 with populations 1, 2, 5,· · · up to 1000 (only 1 run 200–1000). Small horizontal noise added to spread data

Results

Example

Example Magpie changes to triangle.c which reduces its (weighted) instruction count from 9069 to 7544 (17% improvement). Pink code removed, green inserted. (Initially 1300 bytes, mutant 1438 bytes.)

```
@@ -14,9 +14,7 @@

      int triang ;

-    if( side1 <= 0 || side2 <= 0 || side3 <= 0){
-        return 4;
-    }
+

     triang = 0;

@@ -27,6 +25,9 @@
         triang = triang + 2;
     }
     if(side2 == side3){
+        if( side1 <= 0 || side2 <= 0 || side3 <= 0){
+            return 4;
+        }
         triang = triang + 3;
     }

@@ -45,6 +46,10 @@
         return 3;
     }
     else if ( triang == 1 && side1 + side2 > side3) {
+        if(side1 + side2 <= side3 ||
+ side2 + side3 <= side1 || side1 + side3 <= side2){
+            return 4;
+        }
         return 2;
     }
     else if (triang == 2 && side1 + side3 > side2){
```

Conclusions

- Hard to assess effect from including NCDm as part of the fitness
- Not much difference between (1) fitness only, (2) quadratic and (3) linear versions

PTO

- NCDm works best on **looooooooooooooooooooooooooooooooooong** strings
- Dictionary based compression works better on short strings (maybe chose wrong compression method?). See Shoco (from Meta).
- Very ambitious would be to replace fixed population size with fixed population diameter size — at least for the initial, expansive/exploratory phase of GI/GP
- NCDm algorithm could be used alongside fitness in fitness landscapes with many plateaus

thank you for listening