Marcus De Maria

Cart Centering Genetic Algorithm

Data Collection and Results:

WITH FIRST 4 PRE-EXISTING FUNCTIONS

Best tree:
((b - b) > (b + ((a - (a * a)) + a)))
Generation: 38
Size: 13
Depth: 5
Fitness: -0.0701047

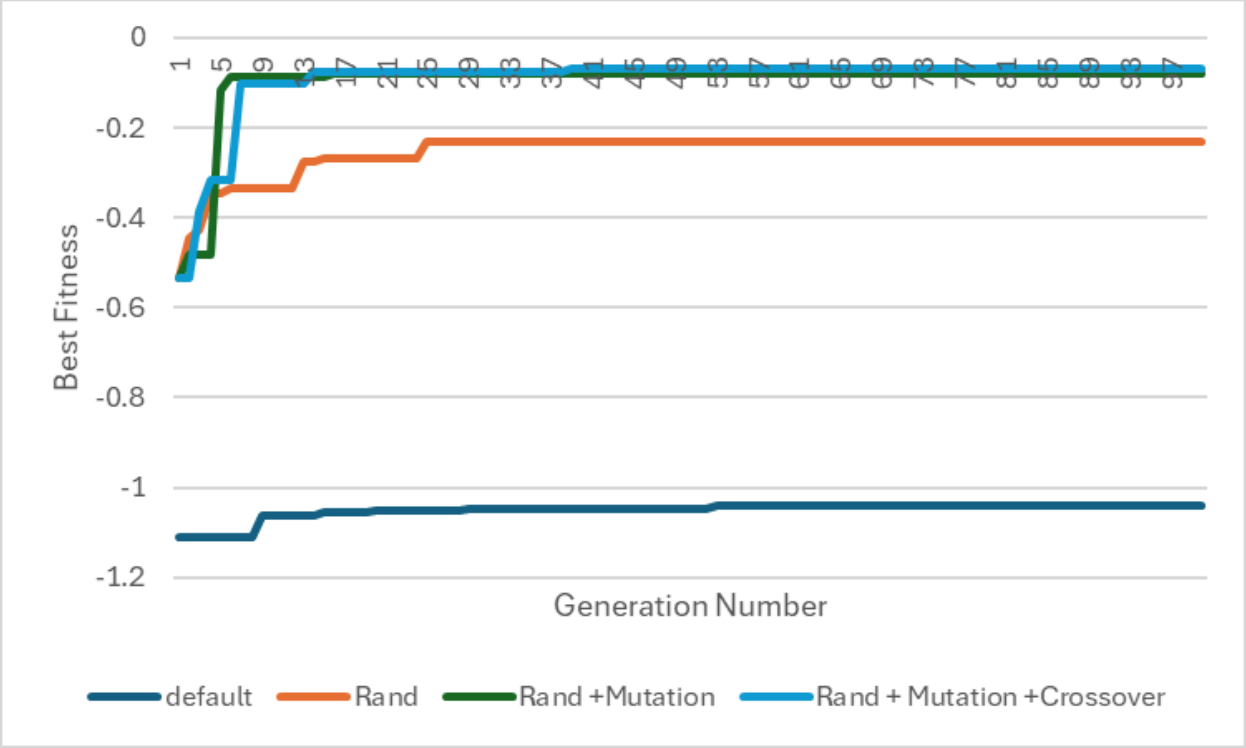

Figure 1: Plot of Part 1/3 Data With Everything Implemented

Marcus De Maria

WITH LEXLESSTHAN

Best tree:
(((a * a) - a) > (a + b))
Generation: 25
Size: 9
Depth: 3
Fitness: -0.0764084

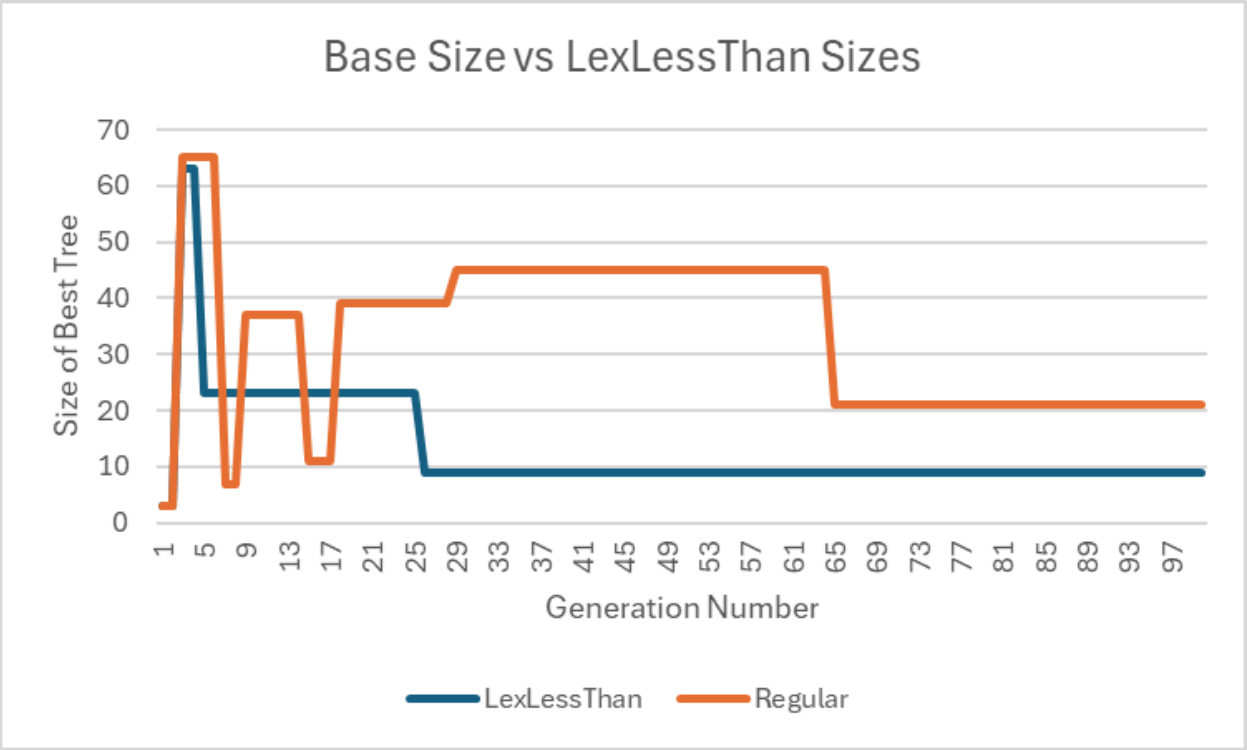

Figure 3: Size of Best Tree Part 2 Comparison

WITH CROSSOVER

Best tree:
(((a + b) / (a > a)) > a)
Generation: 94
Size: 9
Depth: 3
Fitness: -0.0690369

The plotline of the crossover function used in part 3 can be seen in Figure 1.

Marcus De Maria

Short Answer:

Tree Traversal by functions

1. This function is performing a depth-first traversal of a binary tree. Specifically, it's doing a pre-order traversal because it processes the current node before recursively traversing its left and right subtrees. In pre-order traversal, the root node is processed before the left and right subtrees. The condition checking for the abs of a or b adds an in-order aspect as it follows the left-root-right pattern. So this code checks for statements using both pre-order and in-order.

2. This function evaluates the expression represented by the binary tree. It recursively traverses the tree, starting from the given position p, and evaluates each sub-expression until it reaches external nodes (leaves), which represent operands (numbers or variables).
   If the current position p corresponds to an internal node (operator), it recursively evaluates the left subtree and, if the operator has arity greater than 1 (i.e., it's a binary operator), it also evaluates the right subtree. Then it applies the operator to the evaluated operands.
   If the current position p corresponds to an external node (operand), it returns the value of the operand. If the operand is a variable ("a" or "b"), it returns the corresponding value passed as arguments a or b.
   So, this function is evaluating the expression using a depth-first tactic of post-order traversal, visiting the root last.

3. This function doesn't explicitly perform a tree traversal. Instead, it relies on the positions() method to generate a list of positions in the tree. Together the methods explicitly call the preorder() function to traverse the tree in a pre-order manner. Therefore, this method uses a pre-order traversal strategy to generate the list of positions in the binary tree.

4. This function is a recursive function to calculate the size of the subtree rooted at a given node v. It traverses the tree in a depth-first manner, specifically using a post-order traversal. In post-order traversal, the function visits the left subtree, then the right subtree, and finally the root node. This is evident from the fact that it first recursively calculates the size of the left subtree (size(v->left)), then the right subtree (size(v->right)), and finally includes the current node (return 1 + lsize + rsize).