Geometric Semantic Genetic Programming

Alberto Moraglio
University of Birmingham

Krzysztof Krawiec
Poznan University of Technology

Colin Johnson
University of Kent

ThRaSH 2011

Contents

- Semantics in Genetic Programming
- Semantic Geometric Search Operators
- Semantic Operators for:
  - Boolean Functions
  - Symbolic Regression
  - Simplified Programs
- Experiments
- Conclusions & Future Work

SEMANTICS IN GENETIC PROGRAMMING

Traditional GP

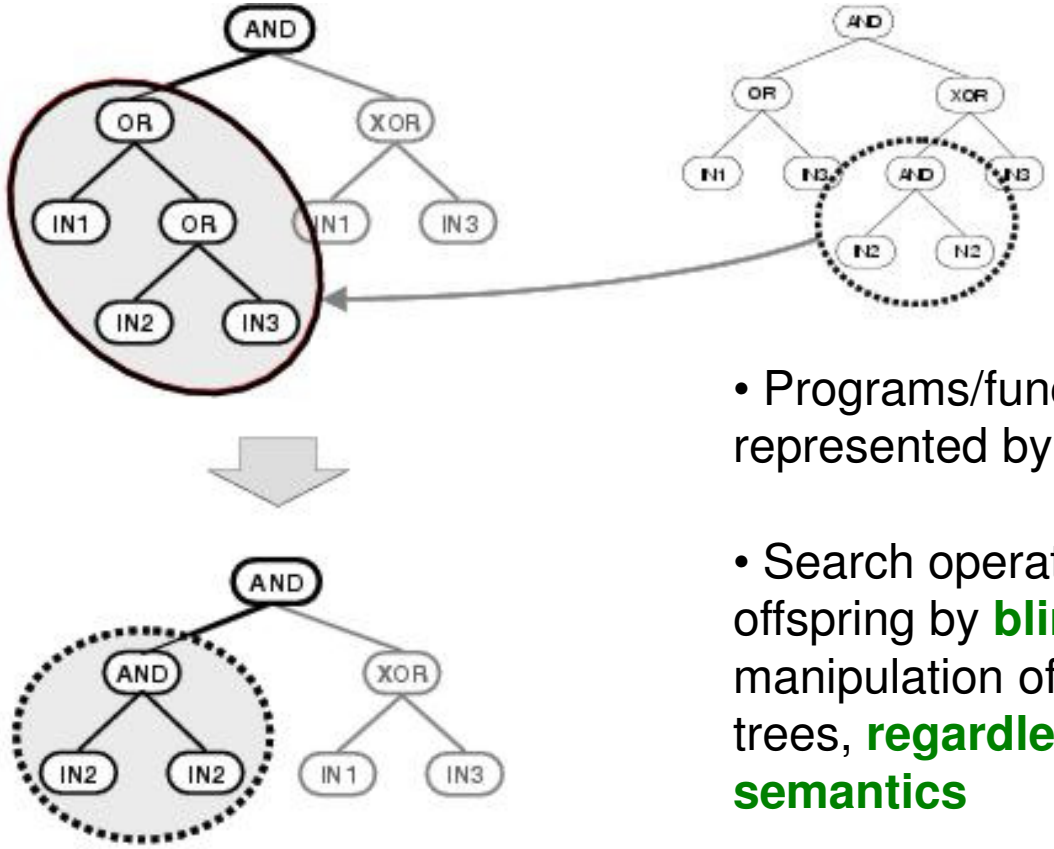


Figure form Michael Lones PhD thesis

- Programs/functions are represented by parse trees

- Search operators produce offspring by **blind** syntactic manipulation of parent parse trees, **regardless of their semantics**

- Crossover preserves syntactic well-formedness, but **why should it work at all?**

Traditional GP

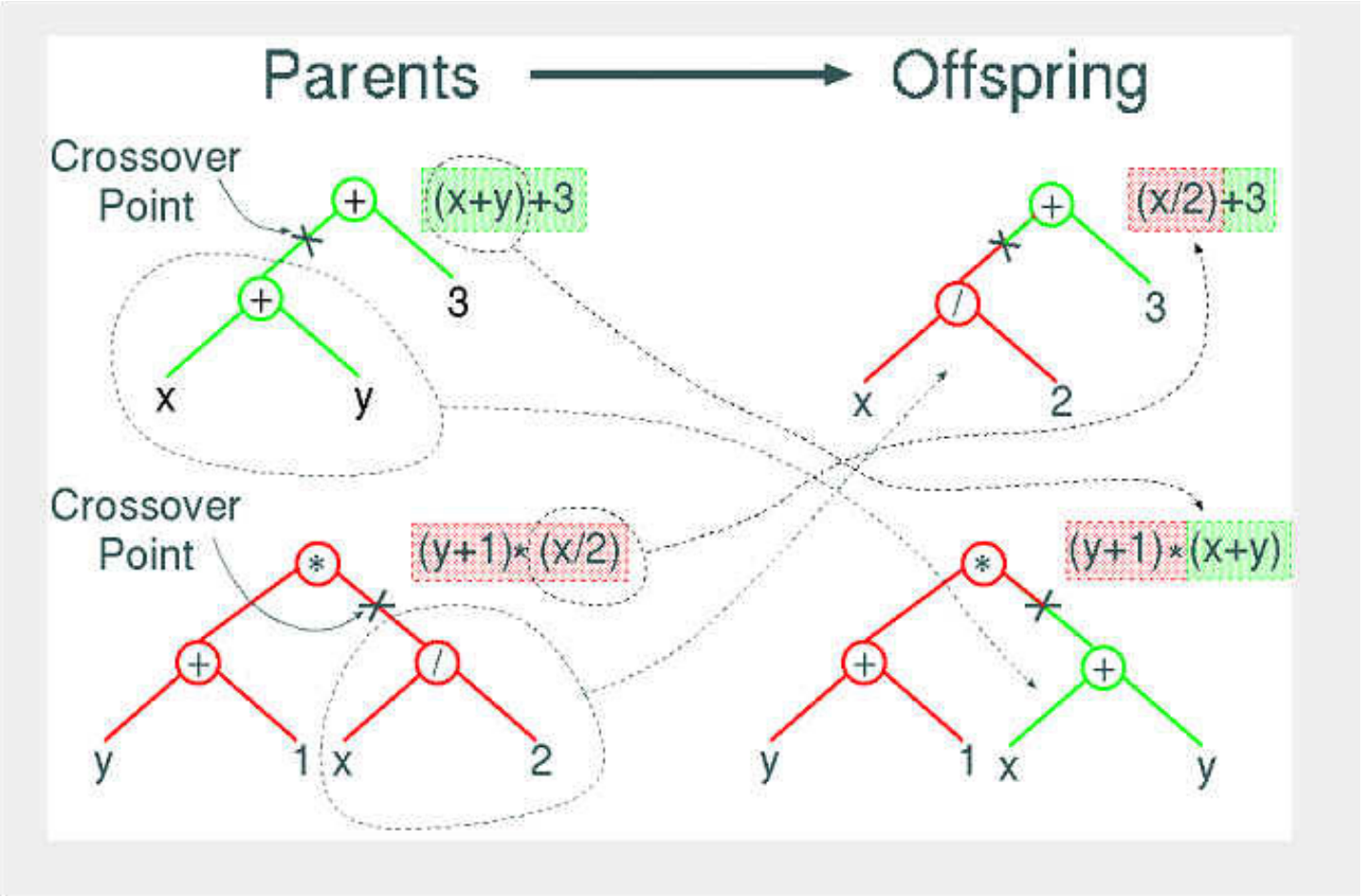


Figure by Riccardo Poli

What is “semantics”?

- Different meanings of “semantics”:
  - Feels like a “metaphysical”/indefinable concept
  - Canonical representation (e.g., Binary Decision Diagrams (BDD) for boolean expressions)
  - Formal semantic (i.e., logic/mathematical description of the behaviour of a program)
  - “Search” semantic: the fitness of the program!
  - Mathematical semantic (i.e., the function computed by the program)

Literature review

- Many trees encode the same function: Johnson et al. used BDD to maintain semantic diversity in the population during evolution by discarding trees with duplicate semantic when created
- Search operators can be biased to consider semantic (Nguyen et al.):
  - Mutation: keep offspring only if “semantically similar” to parent
  - Crossover: swap subtrees in parents only if “semantically similar”

Literature review

- Semantic similarity/distance suggests using search operators defined only in terms of distance (i.e., geometric operators) to search the semantic space: Krawiec approximated semantic geometric crossover by using traditional crossover and discarding offspring not in the segment between parents under semantic distance

Research Question

- All above approaches to semantics are indirect:
  - act on syntax
  - keep only if a semantic criterion is satisfied
- Problems:
  - work better but very wasteful
  - no clear relation between syntactic and semantic searches

Research Question

- Is it possible to search directly the semantic space of programs?
- Rephrased: is it possible to have search operators that acting on the syntax of the programs produce offspring that are guaranteed to respect some semantic criterion/specification by construction?
- Krawiec (2010): “given the complexity of the genotype-phenotype mapping in GP a direct implementation of semantic geometric crossover is probably impossible”

SEMANTIC GEOMETRIC SEARCH OPERATORS

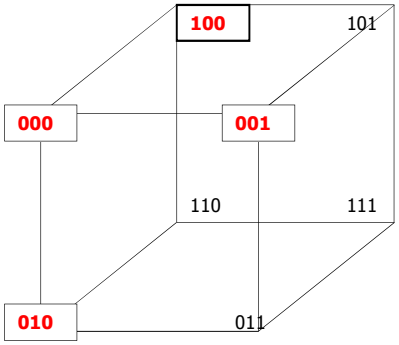
Balls & Segments

$$B(x; r) = \{ y \in S \mid d(x, y) \leq r \}$$

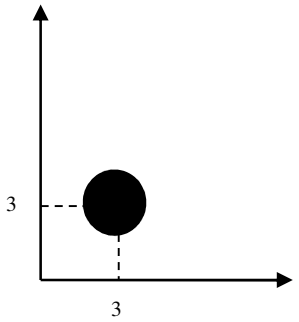
$$[x; y] = \{ z \in S \mid d(x, z) + d(z, y) = d(x, y) \}$$

Squared Balls & Chunky Segments

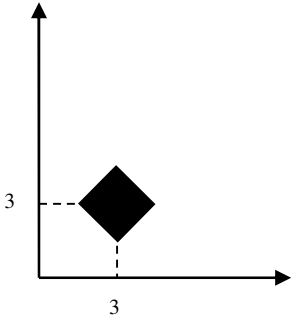
Balls



B(000; 1)
Hamming space

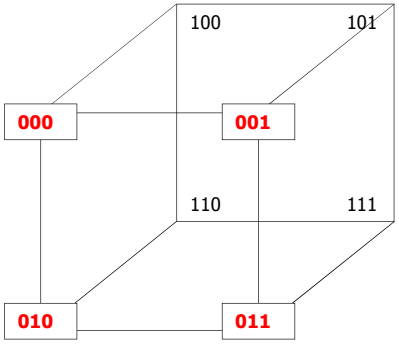


B((3, 3); 1)
Euclidean space

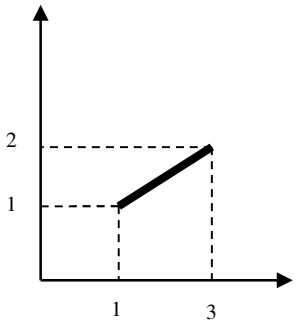


B((3, 3); 1)
Manhattan space

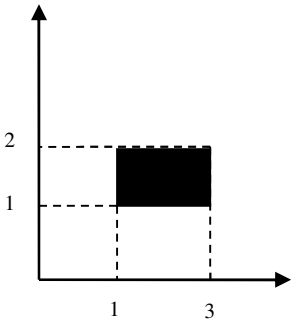
Line segments



[000; 011] = [001; 010]
2 geodesics
Hamming space



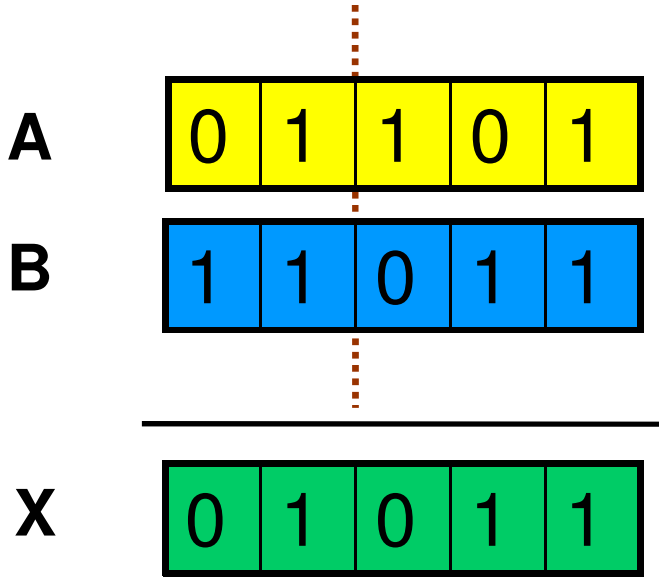
[(1, 1); (3, 2)]
1 geodesic
Euclidean space

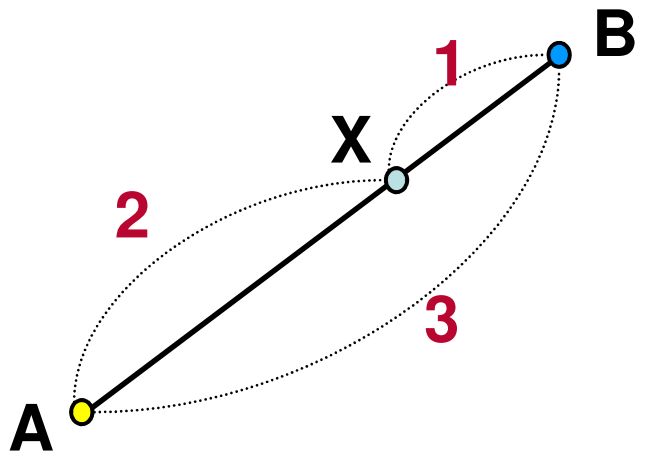


[(1, 1); (3, 2)] = [(1, 2); (3, 1)]
infinitely many geodesics
Manhattan space

Example of Geometric Crossover

- The traditional crossover is geometric under the Hamming distance.





H(A,X) + H(X,B) = H(A,B)

Fitness

- GP is a Machine Learning method:
  - Known: the correct outputs for a fixed given set of inputs {I_i, O_i}
  - Sought: a function belonging to a certain class that interpolates those points, i.e., f(I_i)=O_i for any i
  - Output vector: the vector of the outputs of f is f(I)=(f(I_i))
  - Fitness: a measure on the error on the training set, i.e., distance between the output vectors of f and the target output vector F(f)=D(f(I),O) (ERROR AS DISTANCE)
  - Distance: Hamming distance for Boolean outputs, Euclidean distance for continuous outputs

Semantic Distance

- DEF: the semantic distance between two individuals f1 and f2 is the distance of their output vectors measured with the distance function D used in the definition of the fitness function, i.e., SD(f1,f2)=D(f1(I),f2(I))
- DEF: the space of functions endowed with the distance D is the semantic search space

Semantic Geometric Operators

- Geometric crossover and geometric mutation can be (formally) defined on functions on the semantic search space using the semantic distance SD
- Example: semantic geometric crossover on boolean functions returns offspring boolean functions such that the output vectors of the offspring are in the Hamming segment between the output vectors of the parents

Semantic Fitness Landscape

- Regardless of the specific choice of distance to measure the fitness, the fitness landscape seen by an evolutionary algorithm with semantic geometric operators is a Cone landscape by construction (unimodal with a linear gradient, i.e., FDC=1) which the algorithm can easily optimise
- Cone landscapes being convex landscapes are geometric crossover friendly: the expected fitness of the offspring >= average fitness of their parents (without selection) (Moraglio, FOGA 2011)

Semantic Fitness Landscape

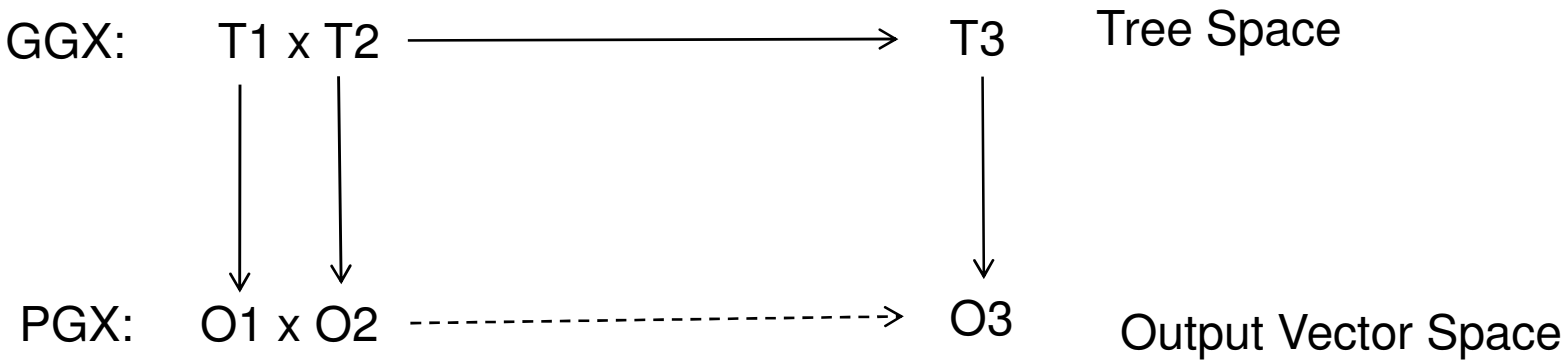
- Symbolic Regression Problem:
  - Find a function that interpolates through a given set of data-points (input-output pairs)
  - Semantic crossover and semantic mutation on the output vectors under Euclidean distance: traditional line crossover and e.g. Gaussian mutation
  - Fitness landscape seen by the evolutionary algorithm is an Euclidean cone

Semantic Fitness Landscape

- Boolean Problem:
  - Find a boolean function that satisfies a given partial truth-table (i.e., data-points, input-output pairs)
  - Semantic crossover and semantic mutation on the output vectors under Hamming distance: a mask-based crossover and e.g. point mutation
  - Fitness landscape seen by the evolutionary algorithm is the one-max!

CONSTRUCTION OF SEMANTIC OPERATORS

Projection of Search Operators via Genotype-Phenotype Map

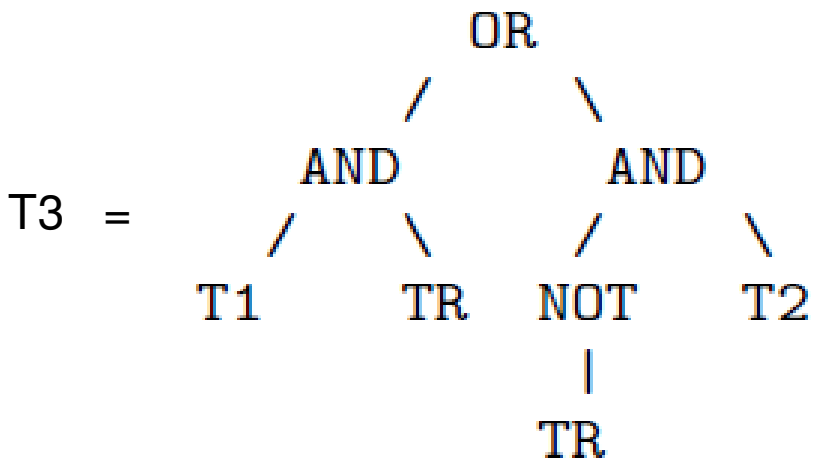


- A search operator on genotypes GGX induces a corresponding search operator on phenotypes PGX such that for any T1, T2: GGX(T1,T2)=T3 then PGX(gp(T1),gp(T2))=gp(T3)

- Sought: an operator GGX of trees that induces via G-P map an operator PGX on output vectors which is geometric crossover w.r.t. a given semantic distance

- GGX implements the semantic crossover acting on the syntax

Semantic Crossover for Boolean Expressions



T1, T2: parent trees
TR: random tree

Claim

The output vector of the offspring T3 is in the Hamming segment between the output vectors of its parent trees T1 and T2 for any tree TR and w.r.t. any subset of given inputs {I_i} and any Boolean Problem P, i.e.,

for any TR, T1, T2, {I_i}, P: O_I(T3) in [O_I(T1), O_I(T2)]_HD

Demonstration: problem

- 3-parity problem: we want to find a function P(X1,X2,X3) that returns 1 when an odd number of input variables is 1, 0 otherwise.

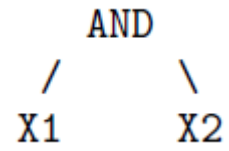
X1	X2	X3	Y
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

Demonstration: fitness & distance

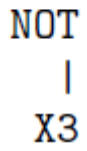
- Input cases: we consider all 8 entries of the truth table (normally the input cases are just a small subset)
- Target output vector is Y = 01101001
- The output vector O(T) of tree T is last column of the truth table
- The fitness f(T) of a tree T is the Hamming distance between its output vector and the target output vector, i.e., f(T)=HD(O(T),Y)
- The semantic distance between two trees T1 and T2 is the Hamming distance between their output vectors, i.e., SD(T1,T2) = HD(O(T1),O(T2))

Demonstration: tree crossover

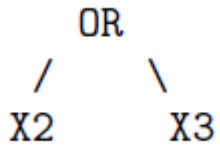
T1 =



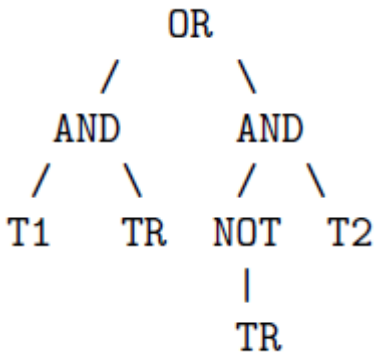
TR =



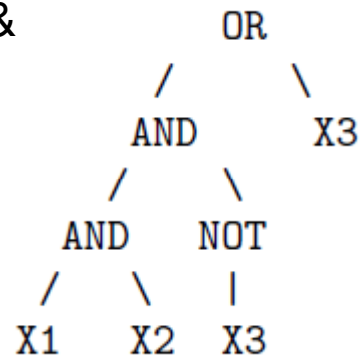
T2 =



T3 =



substitution & simplification



Demonstration: output vector xover

X1	X2	X3	Y	T1	T2	TR	T3
0	0	0	0	0	0	1	0
0	0	1	1	0	1	0	1
0	1	0	1	0	1	1	0
0	1	1	0	0	1	0	1
1	0	0	1	0	0	1	0
1	0	1	0	0	1	0	1
1	1	0	0	1	1	1	1
1	1	1	1	1	1	0	1

• The output vector of TR acts as a crossover mask to recombine the output vectors of T1 and T2 to produce the output vector T3.

• This is a geometric crossover on the semantic distance: output vector of T3 is in the Hamming segment between the output vectors of T1 and T2.

Isomorphism

- The crossover on trees is isomorphic to a geometric crossover on the output vectors under Hamming distance because the logical expression used to construct the offspring T3 from T1, T2 and TR (i.e., T3=T1*TR + T2*TR') describes how to use the recombination mask O(TR) on the output vectors O(T1) and O(T2).

Derivation Sketch

- Offspring function: T3=T1*TR + T2*TR'
- For any input i: T3(i)=T1(i)*TR(i) + T2(i)*TR(i)'
- For any entry of the output vectors i:
  O(T3)(i) = O(T1)(i)*O(TR)(i) + O(T2)(i)*O(TR)(i)'
- The last expression describes for every position i, the selecting action of the recombination mask bit on the corresponding bits in the parents to determine the bit to assign to the offspring (i.e., 1 bit multiplexer function piloted by mask bit).

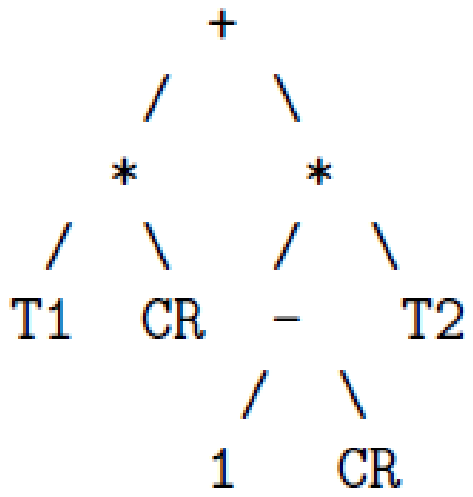
Construction Method

- Take the geometric crossover on output vectors associated with the semantic distance
- Consider the recombination operator action on a single entry output vector
- Replace the single entry by a tree with input I
- Use the domain specific language to describe the recombination action on a single entry
- That description of the recombination action to obtain the offspring IS the offspring

Semantic Crossover for Arithmetic Expressions

Function co-domain: real
Output vectors: real vectors

T3 =

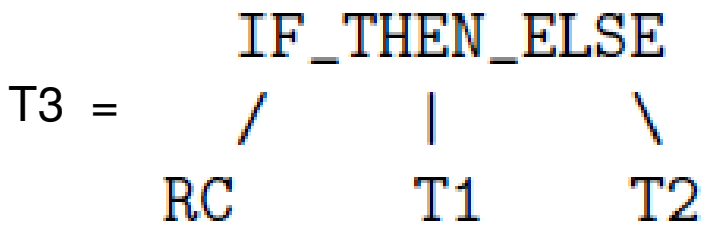


Semantic distance = Manhattan
CR = random function with co-domain [0,1]

Semantic distance = Euclidean
CR = random real in [0,1]

Semantic Crossover for Programs

Function co-domain: symbol
Output vectors: symbol string



Semantic distance = Hamming
RC = random function with boolean co-domain
(i.e., random condition function of the inputs)

Remarks: Domain Language

- Domain-language must be expressive enough to describe the recombination
- Unlike traditional syntactic operators which are of general applicability, semantic operators are domain-specific but not problem specific:
  - Semantic distance
  - Domain language

Remarks: Simplification

- Offspring grows in size very quickly, as the size of the offspring is larger than the sum of the sizes of its parents!
- To keep the size manageable we need to simplify the offspring without changing the computed function:
  - Boolean expressions: boolean simplification
  - Math Formulas: algebraic simplification
  - Programs: simplification by formal methods

Remarks: syntax does not matter!

- The offspring is a function obtained from a functional combination of parent functions
- The offspring is defined purely functionally, independently from how the parent functions and itself are actually represented (e.g., trees)
- The genotype representation does not matter: solution can be represented using any genotype structure (trees, graphs, sequences)/language (java, lisp, prolog) as long as the semantic operators can be described in that language

Remarks: syntax does matter!

- In practice genotype structure/language matters, as it influences:
  - The way random solutions (genotypes) are generated (different representations suggest different “natural” ways of generating solutions). This affects the induced probability distribution on phenotypes.
  - Semantic diversity in the initial population
  - Dependencies in the crossover mask
  - Easiness of algebraic simplification
  - Generalisation/inductive bias of optimal functions found (on unseen input combinations)

Semantic Mutations

- It is possible to derive mutation operators on trees that induce slight variant of known mutations operators on output vectors
- Boolean expressions and Programs: point-mutation on output vectors
- Arithmetic expressions: box mutation

EXPERIMENTS

Experimental Set-Up

Problems

- Symbolic Regression of *Polynomials*
- Classifier (nested IF)
- Parity (Boolean)
- Multiplexer (Boolean)

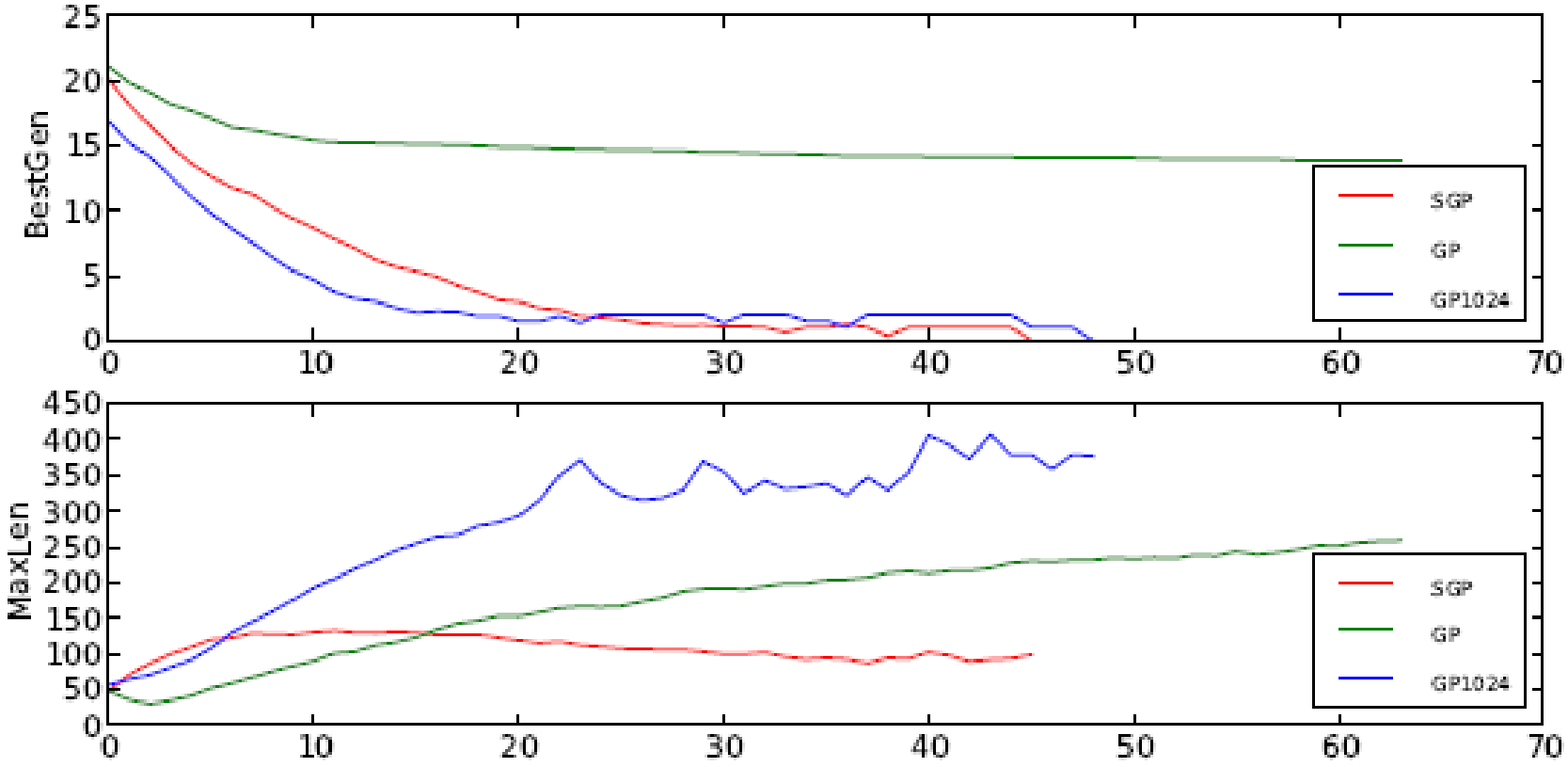
Comparison

- SGP, GP, GP1024
- *GP: same number of evaluations of SGP*
- *GP1024: 100 times evaluations of SGP*

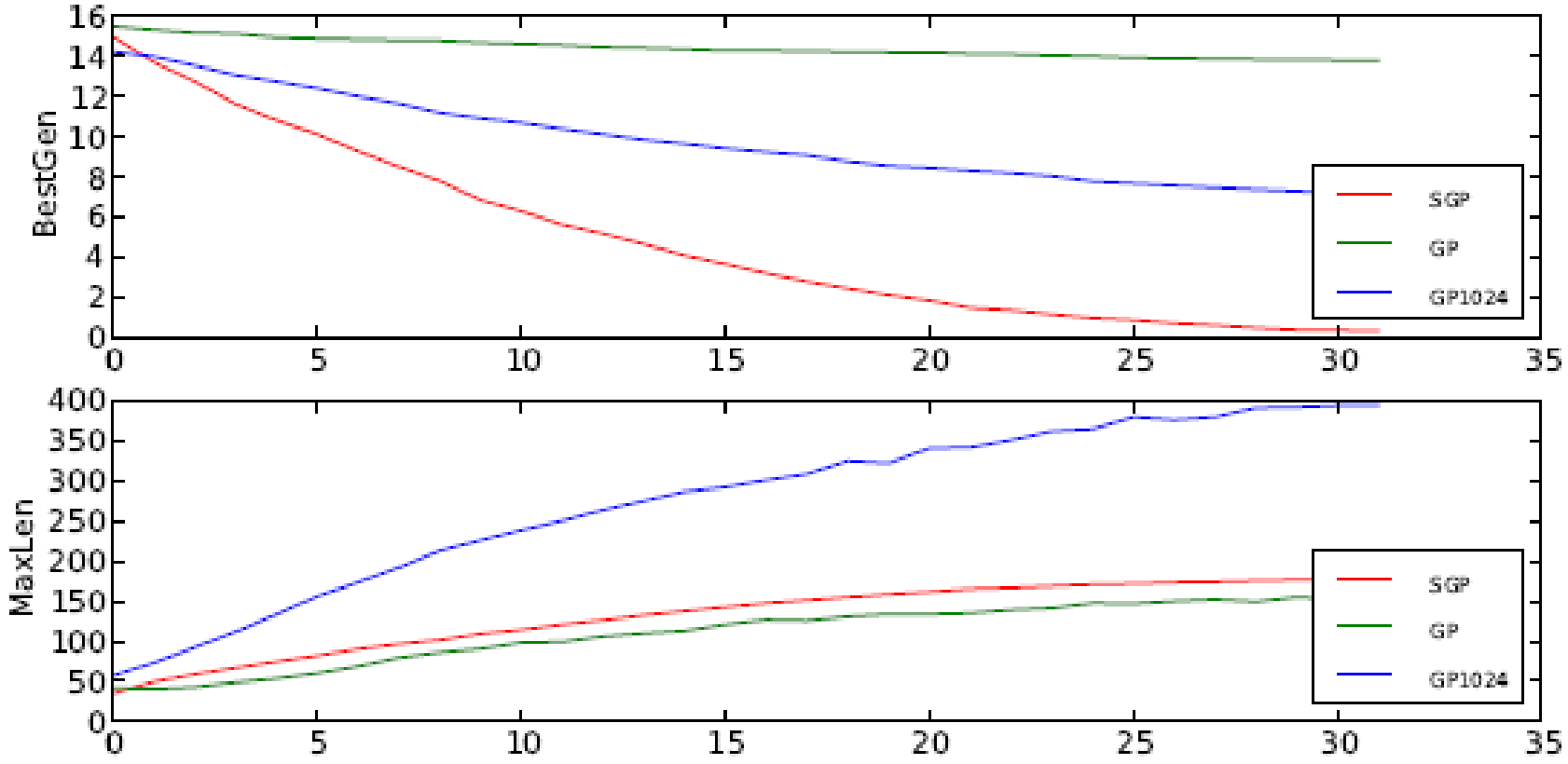
Algorithm (SGP)

- Simple Generational GA with Tournament Selection
- Operators: semantic crossover and semantic mutation followed by simplification
- Implementation: Computer Algebra System (Maple 11), slow but semi-automatic simplification!

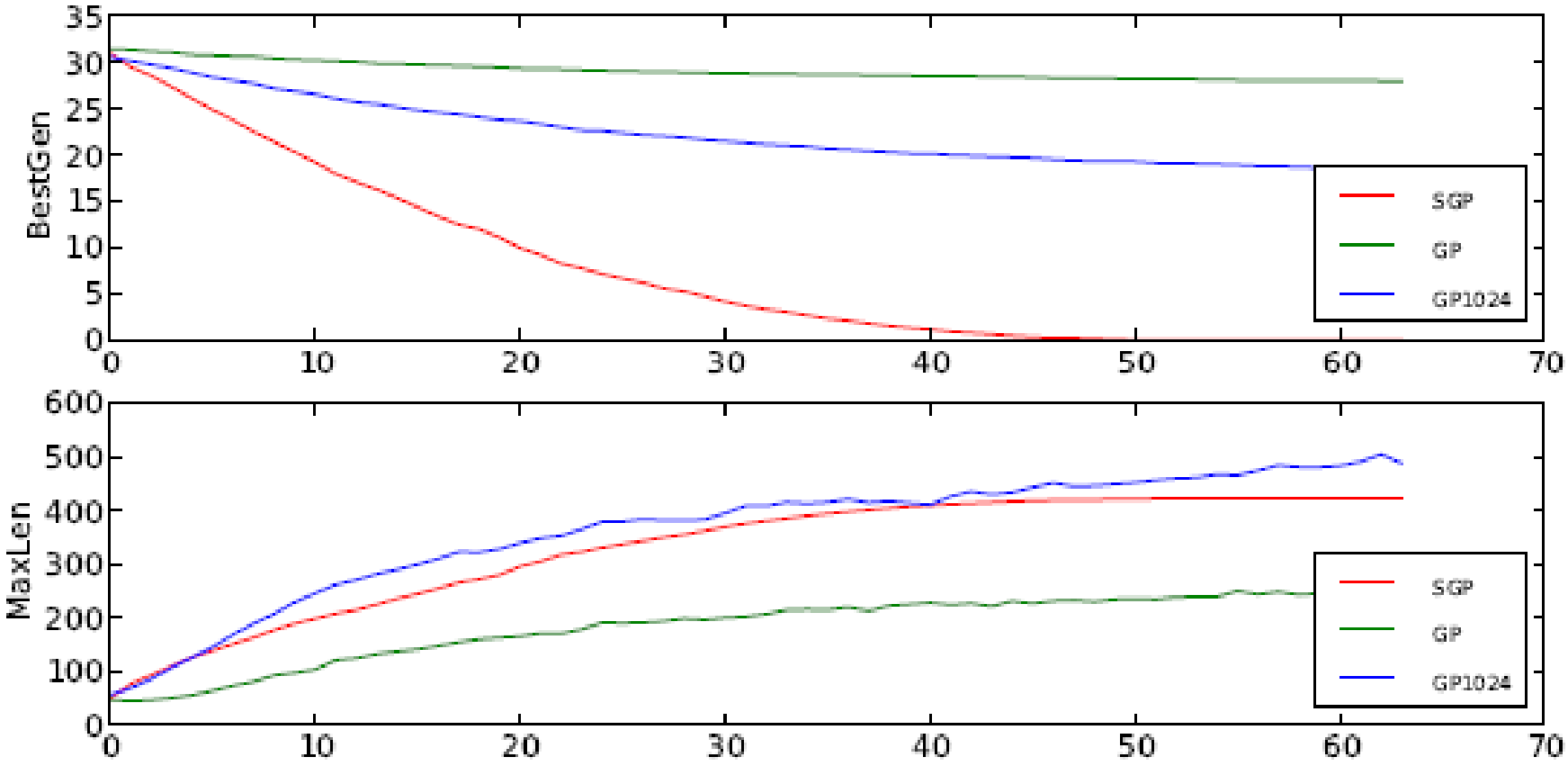
MUX6


BestGen
25
20
15
10
5
0
0
10
20
30
40
50
60
70
SGP
GP
GP1024
MaxLen
450
400
350
300
250
200
150
100
50
0
0
10
20
30
40
50
60
70
SGP
GP
GP1024

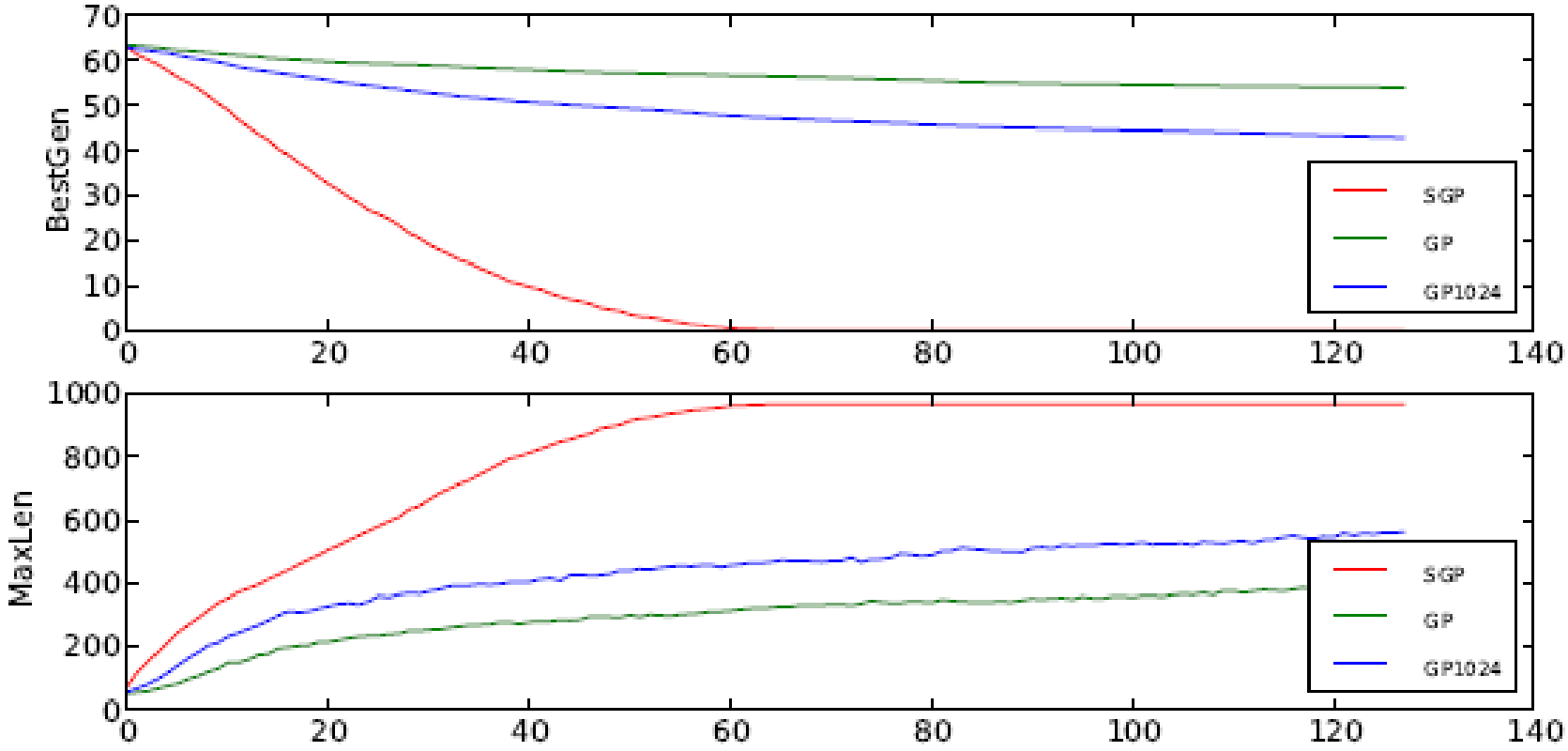
PARITY5


BestGen
16
14
12
10
8
6
4
2
0
0
5
10
15
20
25
30
35
SGP
GP
GP1024
MaxLen
400
350
300
250
200
150
100
50
0
0
5
10
15
20
25
30
35
SGP
GP
GP1024

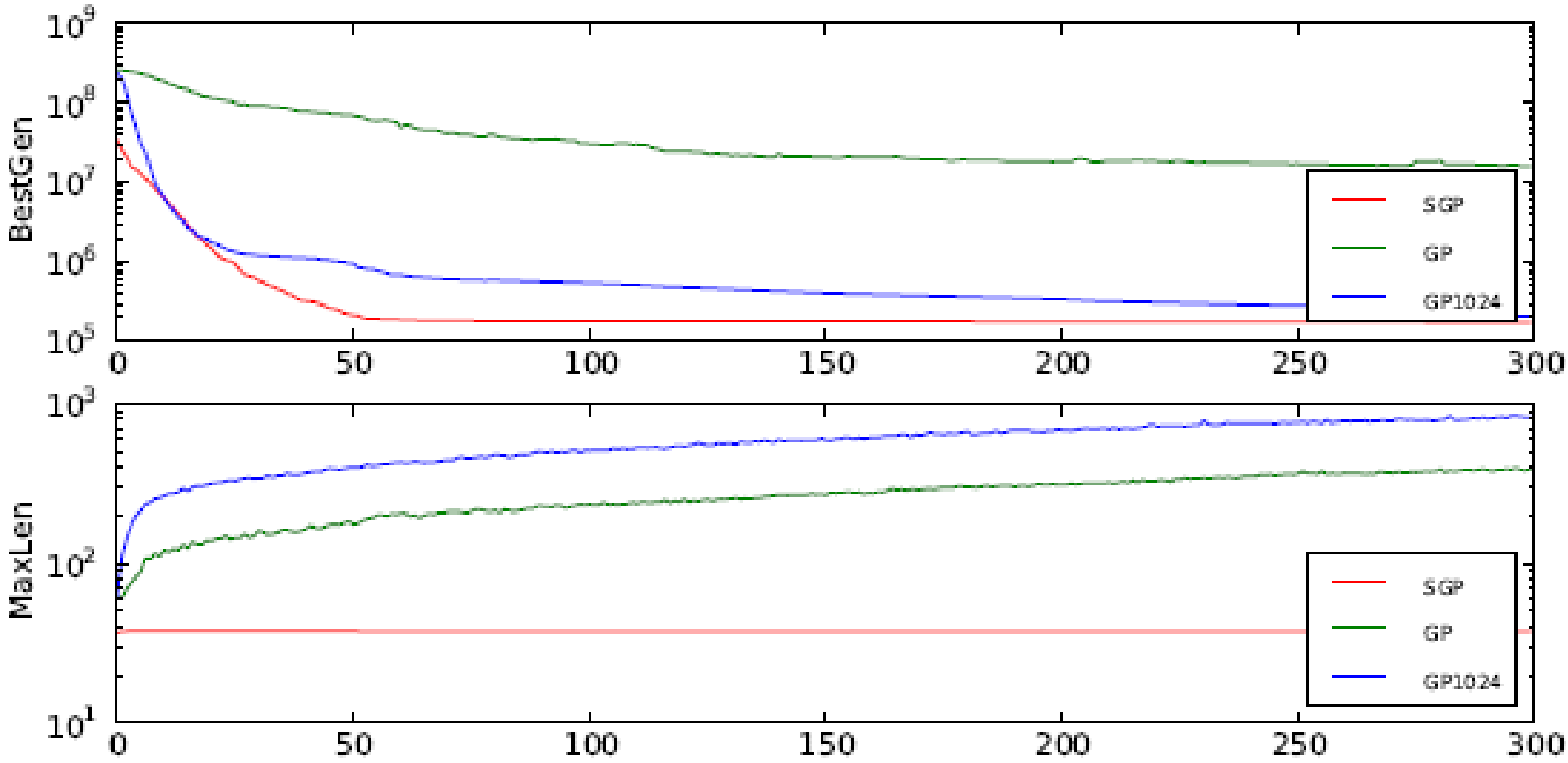
PARITY6


BestGen
0
5
10
15
20
25
30
35
0
10
20
30
40
50
60
70
SGP
GP
GP1024
MaxLen
0
100
200
300
400
500
600
0
10
20
30
40
50
60
70
SGP
GP
GP1024

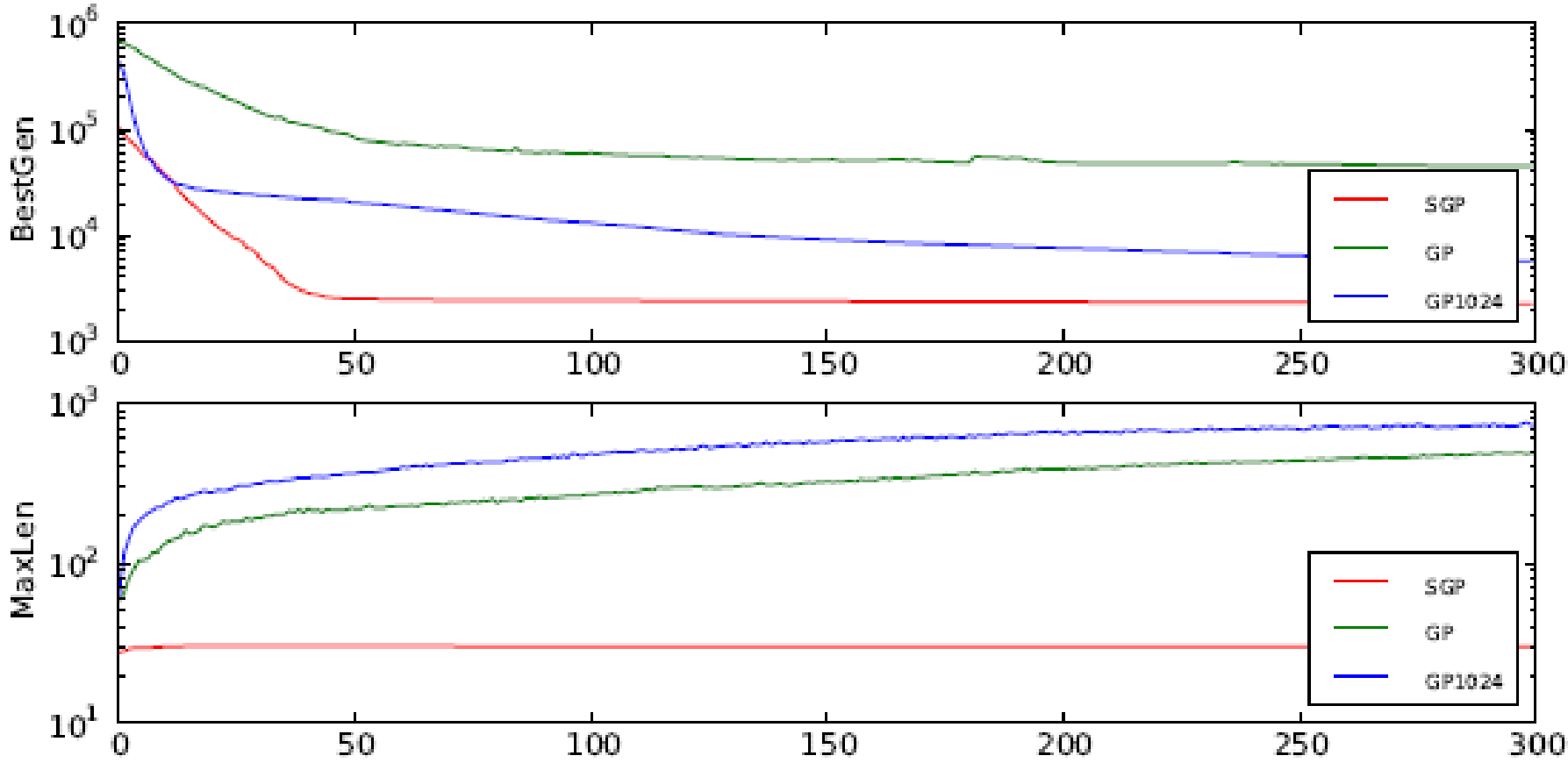
PARITY7


BestGen
70
60
50
40
30
20
10
0
0
20
40
60
80
100
120
140
SGP
GP
GP1024
MaxLen
1000
800
600
400
200
0
0
20
40
60
80
100
120
140
SGP
GP
GP1024

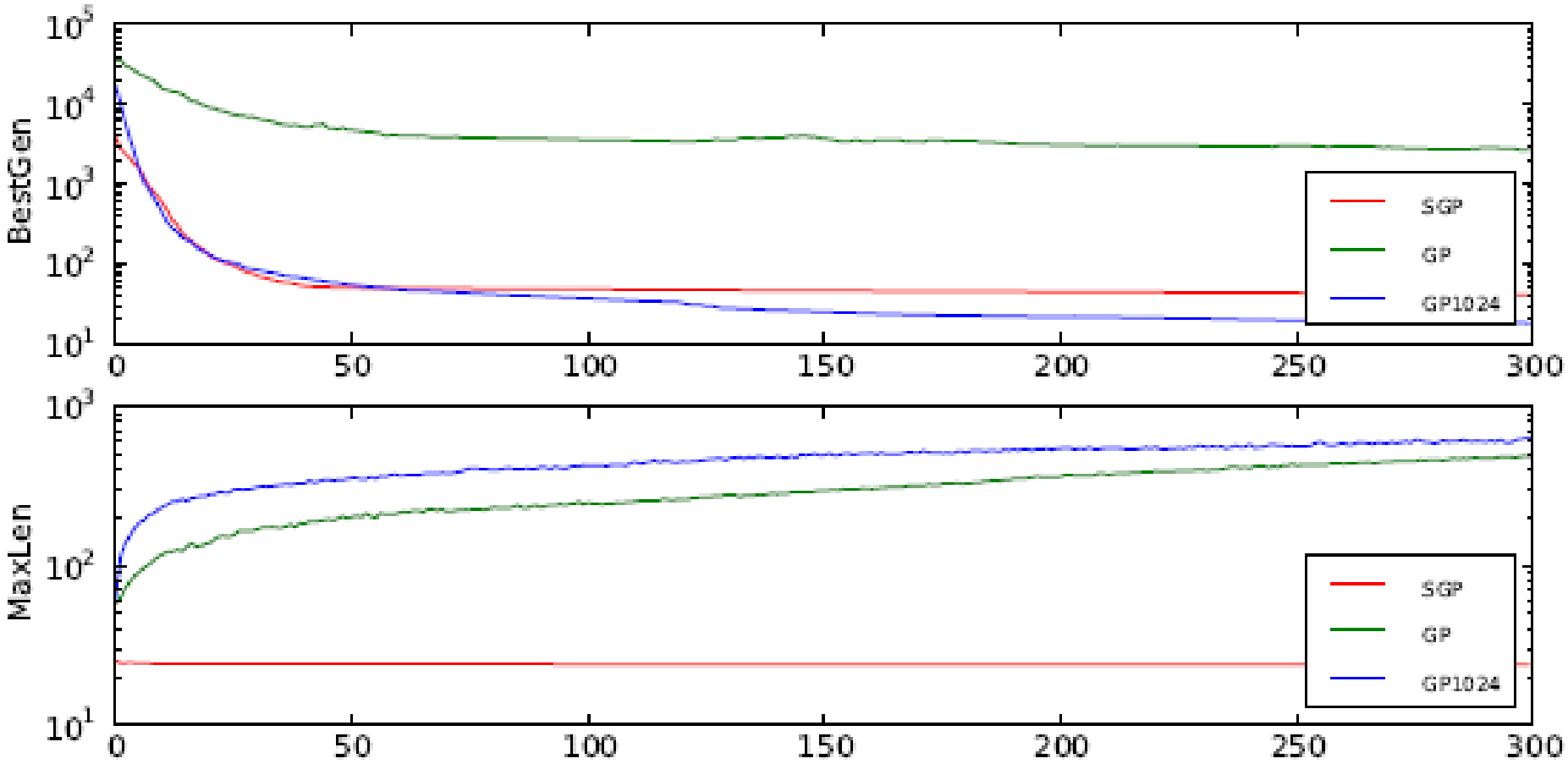
REGRESSION1



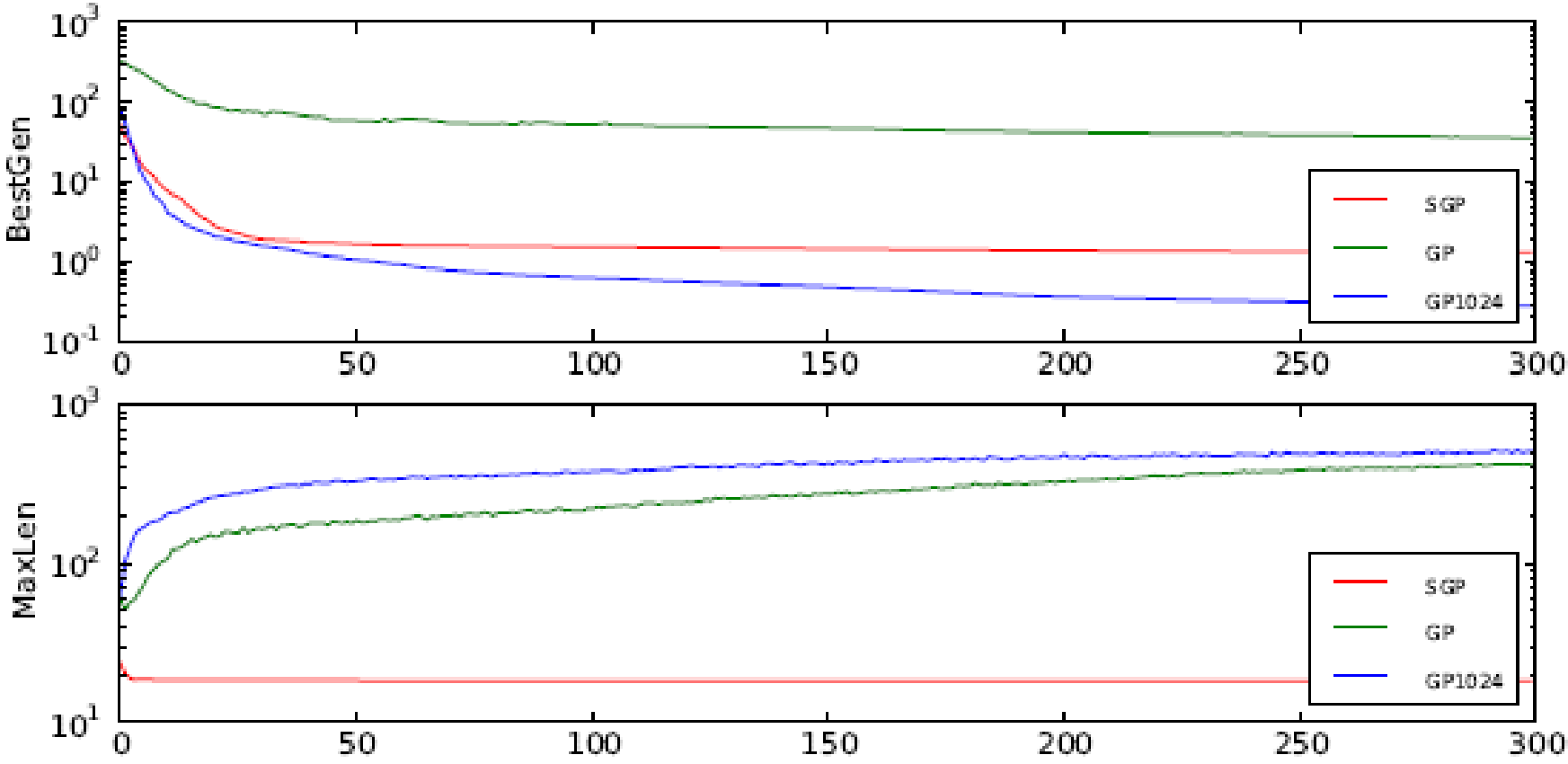
REGRESSION2



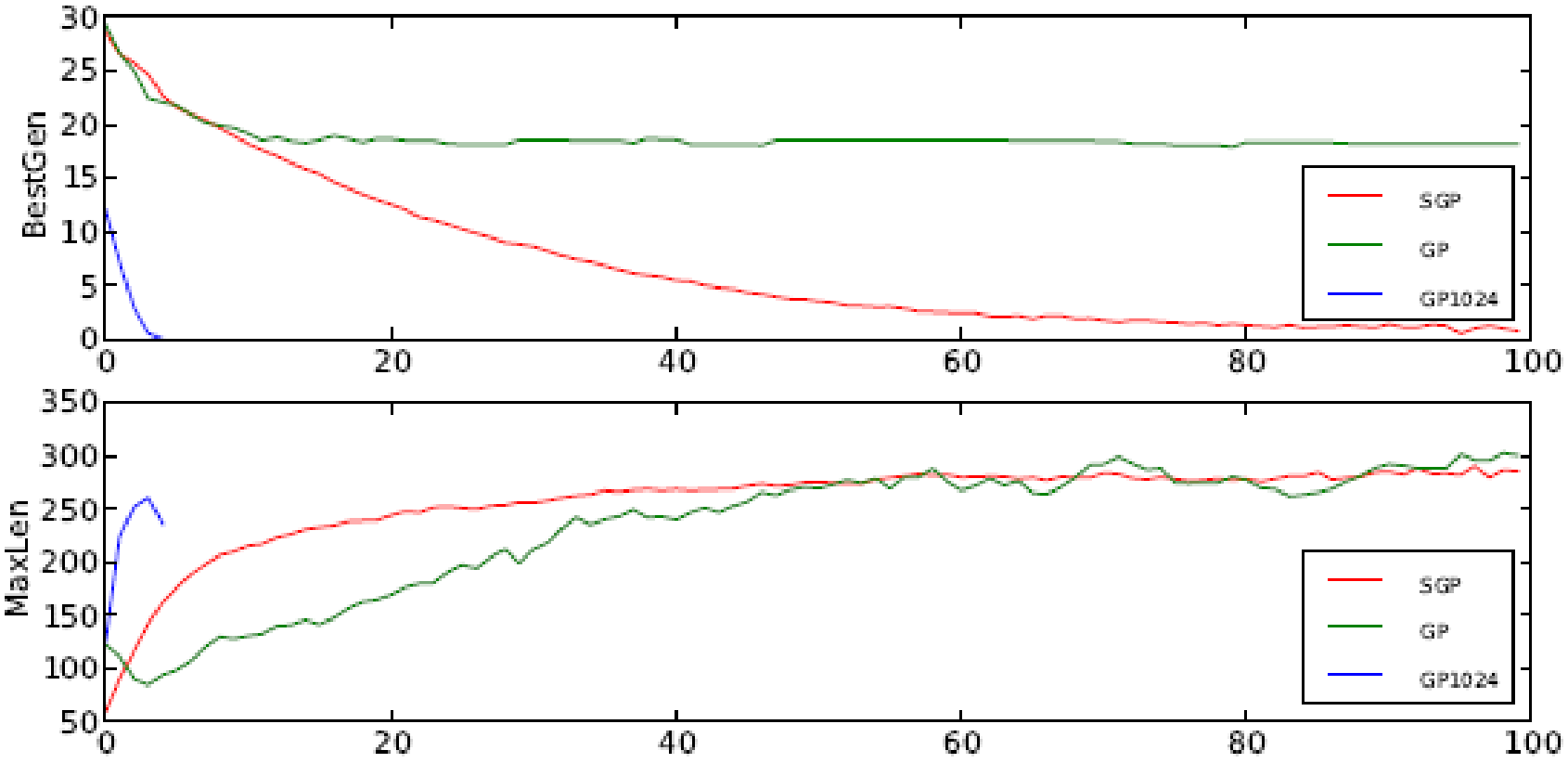
REGRESSION3



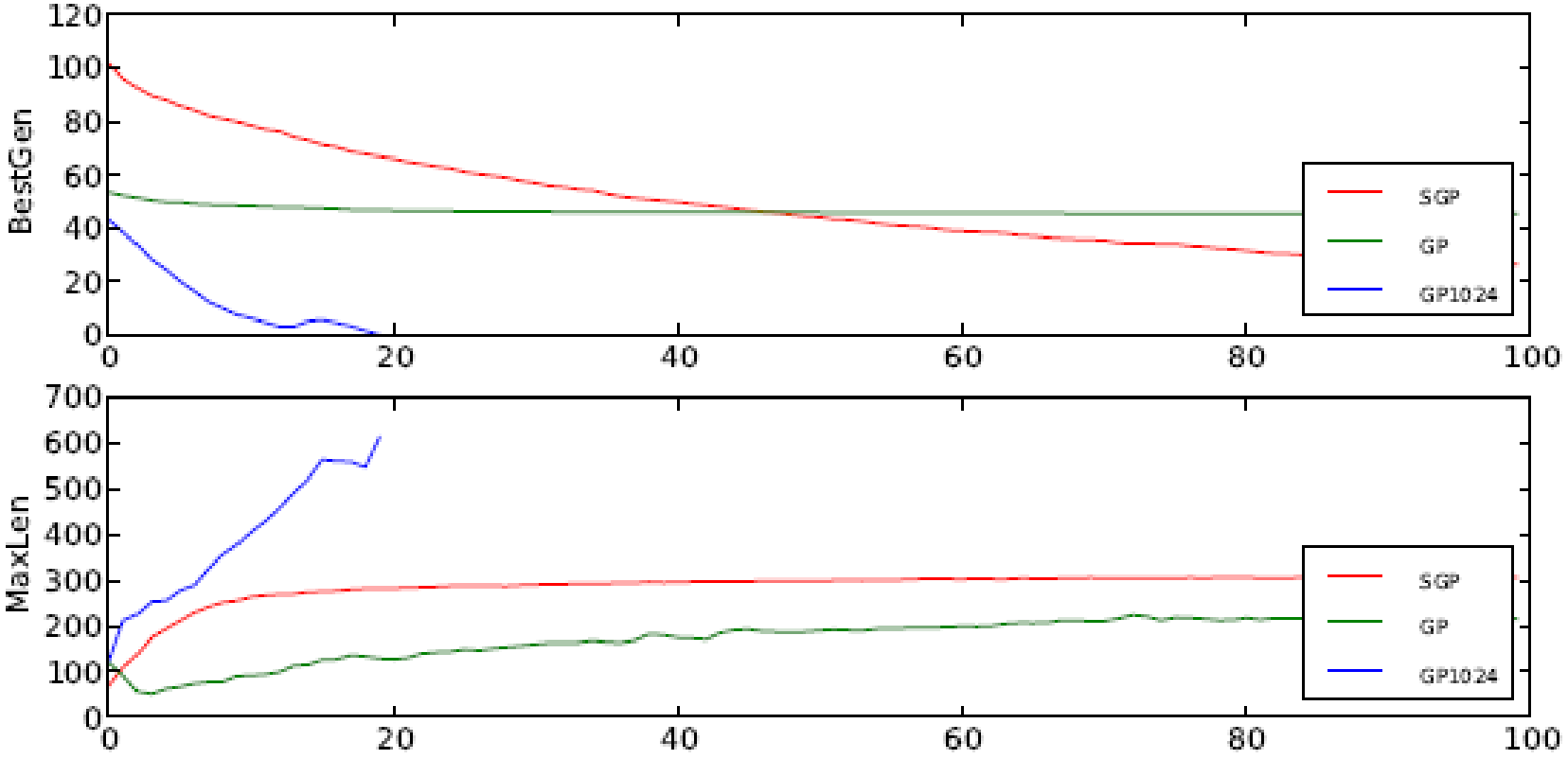
REGRESSION4



CLASSIFIER2



CLASSIFIER5 (debugging!)



CONCLUSIONS

Summary

- Traditional GP does syntactic search, ignoring the meaning of the programs
- Previous attempts to semantic: syntactic search + rejection if not meeting semantic requirements (wasteful)
- What is the semantic space? Is it possible to search the semantic space directly?
- The semantic space is the space of functions endowed with a distance used in the definition of the fitness
- Searching the semantic space is good because semantic geometric operators see a cone landscape (i.e., reduction to one-max!)

Summary

- Due to the complexity of the genotype-phenotype map building syntactic search operators equivalent to geometric operators on the semantic space was thought to be hopeless
- We put forward a new perspective and showed that the genotype-phenotype map is remarkably simple, illustrated a systematic method to derive semantic operators and derived semantic operators for Boolean, Arithmetic and Program domains.
- The semantic operators are purely functional constructions and are independent on the specific genotypic representation.

Summary

- The size of the offspring is larger than the sum of the sizes of their parents, so algebraic simplification is required.
- Experiments showed that the performance of SGP are comparable with standard GP with 10-100 times more trials. Also, the algebraic simplification is effective at maintaining the size of the programs manageable.

Future Work

- Complete Experimental Analysis
- Study the effect of specific genotypic representation on semantic diversity and search biases
- Port of the framework to Java to make it quicker
- Non-algebraic methods for simplifying expression (preserving semantic only on the given set of inputs)
- Derivation of semantic operators for more complex domain (e.g., programs processing variable length lists)
- Use SGP to evolve non-trivial Turing Complete programs, with loops or recursion (this is a GP open challenge)
- Use languages that are algebraic-simplification friendly, e.g., pure functional languages, pure logic languages

Thank you for your attention!