Locality as a Problem Hardness Measure in Genetic Programming

Edgar Galván-López

Trinity College Dublin

School of Computer Science & Statistics
Dublin, Ireland

EC Methods (1/2)

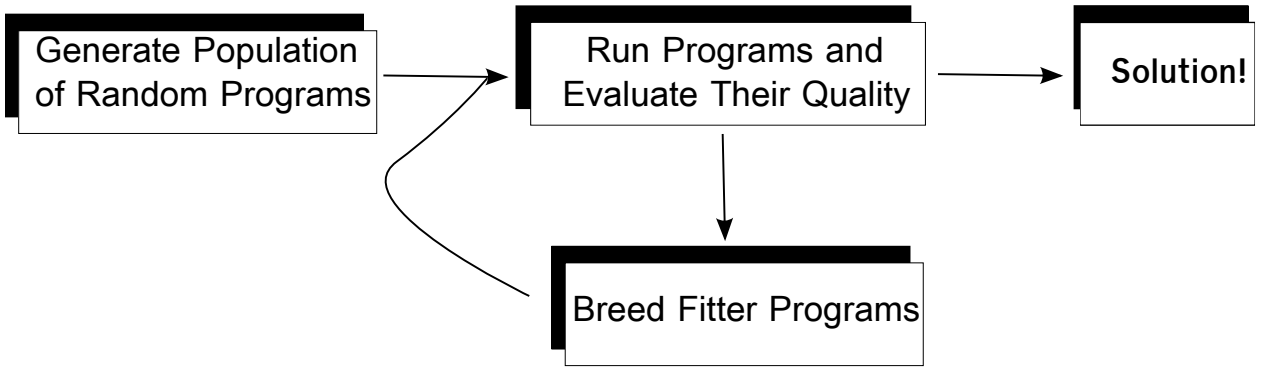
- Evolutionary Computation (EC) techniques allow computer systems to learn.
- EC methods (e.g., Genetic Algorithms, Genetic Programming) are inspired by biological mechanisms of evolution.

EC Methods (2/2)

- EC systems have successfully been used in a number of problems (e.g., computer vision, data analysis, evolvable hardware).
- Hummies are also a good example of the success of EC methods.
- However, it is still difficult to know why some problems are so hard for EC systems.
- In particular, little is known about problem difficulty in GP.

Genetic Programming (GP)

- GP is a domain-independent method for getting computers to *automatically* solve a problem starting from a *high-level* statement of what needs to done!



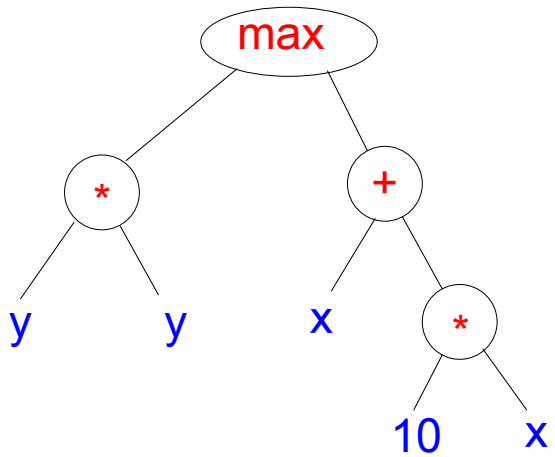
General Steps of GP

1. Randomly create an initial population of individuals from the available primitives.
2. Iterate the following until the termination criterion is satisfied:
   - 2.1 Execute each program and assess its fitness.
   - 2.2 Select one or two programs to participate in genetic operators.
   - 2.3 Create new programs by applying genetic operators with specified probabilities.
3. Return the best-so-far individual.

Program Representation

- Programs are expressed in GP as syntax trees rather than as lines of codes.

max(y * y, x + 10 * x) =



Functions

Terminals

It helps to visualise evolution dynamics

- The concept of a fitness landscape [10] has dominated the way geneticists think about biological evolution and has been adopted within the EC community and many others as a way to visualise evolution dynamics.

Loose Definition of a Fitness Landscape

- A fitness landscape can be seen as a plot where each point on the horizontal axis represents all the genes in that individual corresponding to that point.

Formal definition of fitness landscape

- More formally, a fitness landscape is normally defined as a triplet (x, χ, f): (a) a set x of configurations, (b) a notion χ of neighbourhoood, distance or accessibility on x, and finally, (c) a fitness function f.

Locality originally studied in bitstrings Genetic Algorithm

- Locality, proposed by Rothlauf [9], refers to how well neighbouring genotypes correspond to neighbouring phenotypes.
- His research distinguished two forms of locality: high and low locality.
- Rothlauf reported good results on locality using bitstrings. More recently, he studied locality using grammatical evolution.

Distance Measure for Locality

- The definition of locality assumes that a distance measure exists on both genotype and phenotype spaces.
- Strictly speaking, in standard GP, there are no phenotypes distinct from genotypes, so we extend the notion of locality to the genotype-fitness mapping.

High and Low Locality

- A representation is said to have *high locality* if all neighbouring genotypes correspond to neighbouring phenotypes.
- A representation has *low locality* if some neighbouring genotypes do not correspond to neighbouring phenotypes.

Importance of Locality

- Understanding how well neighbouring genotypes correspond to neighbouring phenotypes is a key element in understanding evolutionary search.
- In the abstract sense, a mapping has *locality* if neighbourhood is preserved under that mapping.

Quantitative Definition of Locality

- Rothlauf gives a quantitative definition of locality: "the locality d_m of a representation can be defined as

$$d_m = \sum_{d^g(x,y)=d^g_{\min}} |d^p(x,y) - d^p_{\min}| \quad (1)$$

where $d^p(x,y)$ is the phenotypic distance between the phenotypes x and y, $d^g(x,y)$ is the genotypic distance between the corresponding genotypes, and $d^p_{\min}$ resp. $d^g_{\min}$ is the minimum distance between two (neighbouring) phenotypes, resp. genotypes".

The implications of locality and non-locality

- Rothlauf claims that a representation that has locality will be more efficient at evolutionary search.
- This, however, changes when a representation has non-locality.
- To explain how non-locality affects evolution, Rothlauf considered problems in three categories, taken from the definition of *fitness distance correlation*.

Definition

- Fitness distance correlation (fdc) measures the hardness of a landscape according to the correlation between the distance from the optimum and the fitness of the solution.
- Given a set given a set $F = \{f_1, f_2, ..., f_n\}$ of fitness values of n individuals and the corresponding set $D = \{d_1, d_2, ..., d_n\}$ of distances to the nearest optimum, we compute the correlation coefficient r,

$$r = \frac{C_{FD}}{\sigma_F \sigma_D},$$

where:

$$C_{FD} = \frac{1}{n} \sum_{i=1}^{n} (f_i - \overline{f})(d_i - \overline{d})$$

Classification of hardness in *fdc*

- According to Jones, a problem can be classified in one of three classes:
  1. *easy* ($r \leq -0.15$), in which fitness increases as the global optimum approaches,
  2. *difficult* ($-0.15 < r < 0.15$), for which there is no correlation between fitness and distance, and
  3. *misleading* ($r \geq 0.15$), in which fitness tends to increase with the distance from the global optimum.

Categories in fdc

- If a given problem lies in the first category (i.e., easy), a non-locality representation will change this situation by making it more difficult and now, the problem will lie in the second category.
- If a problem lies in the second category, a non-locality representation does not change the difficulty of the problem.
- Finally, if the problem lies in the third category, a representation with low locality will transform it so that the problem will lie in the second category.

Extending the Definition of Locality to the Genotype-Fitness Mapping

- We have adopted the traditional GP system, where there is no explicit genotype-phenotype mapping, so we can say that there are no explicit phenotypes distinct from genotypes.
- We will regard two individuals as neighbours in the genotype space if they are separated by a single mutation.
- From this we consider whether genotypic neighbours turn out to be *fitness* neighbours.

Phenotypic neighbours whose fitness are 0 and 1

- Recall that locality refers to how well neighbouring genotypes correspond to neighbouring phenotypes and that a representation that shows to preserving good neighbourhood is preferred.
- This intuitively means that a minimum fitness distance is required. For instance, for discrete values a minimum fitness distance can be regarded as 0 or 1.
- The key question is whether to regard genotypic neighbours whose fitness distances are 0 and 1 to be instances of neutrality, locality, or non-locality.

Def_0: Difference between phenotypic neighbours = 1

- The most straightforward extension of Rothlauf's definition might regard two individuals as fitness-neighbours if the difference of their fitness values is 1 and regard fitness-neutral mutations as non-local. This leads to the following Def_0.

$$d_m = \frac{\sum_{i=1}^{N} |fd(x_i, m(x_i)) - fd_{\min}|}{N} \tag{2}$$

where $fd(x_i, m(x_i)) = |f(x_i) - f(m(x_i))|$ is the fitness distance between a randomly-sampled individual x_i and the mutated individual $m(x_i)$, $fd_{\min} = 1$ is the minimum fitness distance between two individuals, and N is the sample size.

Def_1: Difference between phenotypic neighbours = 0

- It might be preferable to redefine the minimum distance in the fitness space as zero, giving the same locality definition as before but with $fd_{\text{min}} = 0$.

Def_2: Difference between phenotypic neighbours ≥ 2

- Finally, it might be better to treat only true divergence of fitness as indicating poor locality. Therefore we might say that fitness divergence occurs only when the fitness distance between the pair of individuals is 2 or greater: otherwise the individuals are regarded as neighbours in the fitness space. This leads to the following definition:

$$d_m = \frac{\sum_{i=1:fd(x_i,m(x_i))\geq 2}^{N} fd(x_i, m(x_i))}{N} \quad (3)$$

Generalisation of Genotype-Fitness Locality to Continuous-Valued Fitness

- Several aspects of Rothlauf's definition and the extensions to it given before assume that phenotypic and fitness distances are discrete-valued. In particular, it is assumed that a minimum distance exists. For GP problems of continuous-valued fitness it is necessary to generalise the previous definitions.

Def_0 on Continuous-Values Fitness

- Under the first definition (Def_0), the idea is that mutations *should* create fitness distances of 1: lesser fitness distances are non-local, as are greater ones. In the continuous case we set up bounds, α and β, and say that mutations *should* create fitness distances $\alpha < fd < \beta$. When $fd < \alpha$, we add $\alpha - fd$ to d_m, penalising an overly-neutral mutation; when $fd > \beta$, we add $fd - \beta$ to d_m, penalising a highly non-local mutation. Each of these conditions reflects a similar action in the discrete case.

Def_1 on Continuous-Values Fitness

- Under the second definition (Def_1), the quantity being calculated is simply the mean fitness distance. This idea carries over directly to the continuous case.

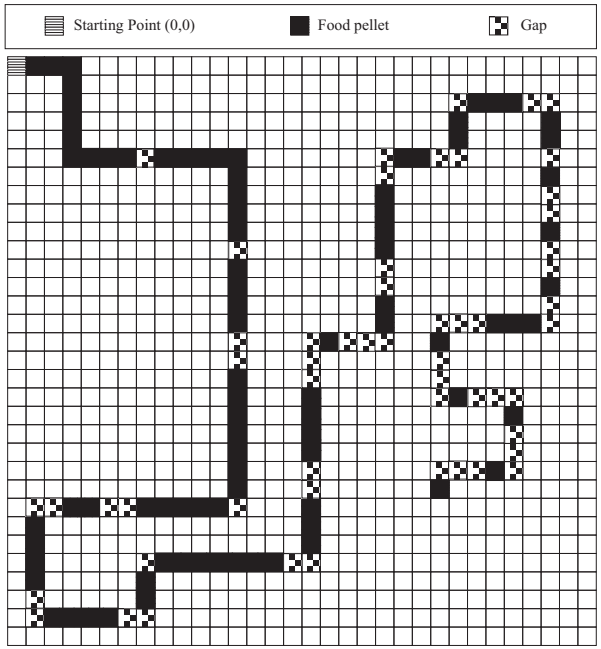
Def_2 on Continuous-Values Fitness

- Under the third definition (Def_2), both neutral and small non-zero fitness distances are regarded as local; only large fitness distances contribute to d_m. Thus, in the continuous case, only when $fd > \beta$ do we add a quantity $(fd - \beta)$ to d_m, penalising mutations of relatively large fitness distances.

Boolean Even-*n*-Parity Problems (n={3,4})

- The goal is to evolve a function that returns true if an even number of the inputs evaluate to true, and false otherwise. The maximum fitness for this type of problem is 2^n. The terminal set is the set of inputs.
- These problems require the combination of several XOR functions, and are difficult if no bias favorable to their induction is added in any part of the algorithm.

Artificial Ant Problem



- The ant must eat all the food pellets (normally in 600 steps) scattered along a twisted track that has single, double and triple gaps along it. The terminal set used for this problem is $T = \{Move, Right, Left\}$.

Real-Valued Symbolic Regression problems

- The goal of this type of problem is to find a program whose output is equal to the values of functions. In this case we used functions $F_1 = x^4 + x^3 + x^2 + x$ and $F_2 = 2sin(x)2cos(y)$.
- Thus, the fitness of an individual reflects how close the output of an individual comes to the target.
- It is common to define the fitness as the sum of absolute errors measured at different values of the independent variable x, in this case in the range [-1.0,1.0].
- We have defined an arbitrary threshold of 0.01 to indicate that an individual with a fitness less than the threshold is regarded as a correct solution, i.e. a "hit".

Even-n-Parity problem

- The terminal set used for this problem is the set of inputs, often called $T = \{D_0, D_1, \cdots, D_{n-1}\}$.
- The standard function set is $F_{E3} = \{NOT, OR, AND\}$.
- An alternative function set for comparison: $F_{E4} = \{AND, OR, NAND, NOR\}$.

Artificial ant problem

- The terminal set used is $T = \{Move, Right, Left\}$.
- The standard function set is $F_{A3} = \{ \text{If}, P2, P3\}$.
- An alternative function set for comparison: $F_{A4} = \{ \text{If}, P2, P3, P4\}$. The only difference is the addition of an extra sequencing function, $P4$, which runs each of its four subtree arguments in order.

Real-Valued Symbolic Regression problems

- The terminal set is defined by the variables used in the function (e.g., $T = \{x\}$).
- The standard function set $F_{S6} = \{+, -, *, \%, Sin, Cos\}$.
- An alternative function set for comparison purposes: $F_{S4} = \{+, -, *, \%\}$.

Dummy arguments

- It is called uniform [6] because all internal nodes are of the same arity.

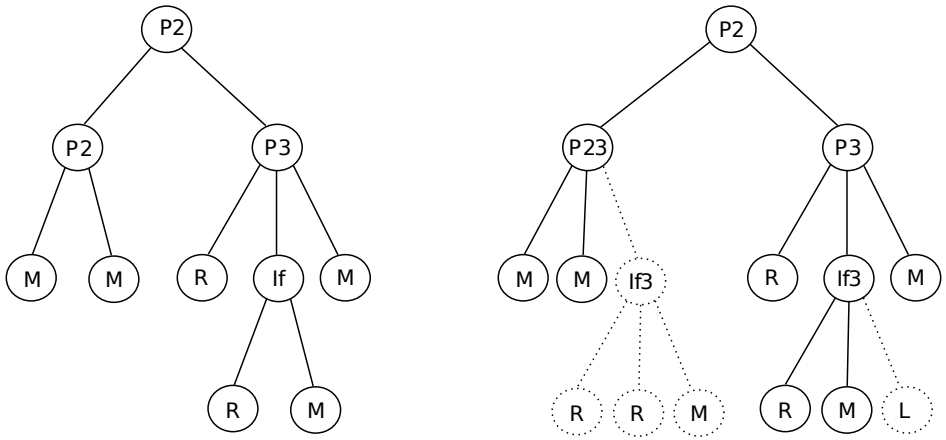


Figure : A typical GP individual (left) and the same individual with uniform arity 3 (right). Dashed lines indicate dummy arguments.

Function sets for Uniform GP

- We defined three function sets:
  - $F_{E3^*} = \{AND, OR, NOT2\}$ for the Even-n-Parity ($n = \{3, 4\}$),
  - $F_{A3^*} = \{If3, P23, P3\}$ for the Artificial Ant and
  - $F_{S6^*} = \{+, -, *, \%, Sin2, Cos2\}$ for the Symbolic Regression problems.
- $NOT2, If3, P23, Sin2, Cos2$ allow the addition of dummy arguments as explained previously.

Function Sets

Table : Function sets used on all the problems.

Problem	*Function Sets*
Even-n-Parity	$F_{E3} = \{AND, OR, NOT\}$ $F_{E4} = \{AND, OR, NAND, NOR\}$ $F_{E3^*} = \{AND, OR, NOT2\}$
Artificial Ant	$F_{A3} = \{IF, P2, P3\}$ $F_{A4} = \{IF, P2, P3, P4\}$ $F_{A3^*} = \{IF3, P23, P3\}$
Symbolic Regression	$F_{S6} = \{+, -, *, \%, Sin, Cos\}$ $F_{F4} = \{+, -, *, \%\}$ $F_{S6^*} = \{+, -, *, \%, Sin2, Cos2\}$

Six different mutation operators

- *One-Point* mutation replaces a node by a new node.
- *Subtree* mutation replaces a randomly selected subtree with another randomly created subtree.
- *Permutation* mutation creates a new individual by selecting randomly an internal node and then randomly permuting its arguments.
- *Hoist* mutation creates a new individual. The resulting offspring is a copy of a randomly chosen subtree of the parent.
- *Size-fair* subtree mutation has two variants: (a) the size of the new individual is given by the size s of a subtree chosen at random within the parent. Size s is then used for creating a new individual randomly; (b) the size of the replacement subree is chosen uniformly in the range $[l/2, 3l/2]$ (where l is the size of the subtree being replaced).

Sampling Method

- To have sufficient statistical data, we created 1,250,000 individuals for each of the six mutation operators described previously.
- These samplings were created using the traditional ramped half-and-half initialisation method using depths $= [3, 8]$.

Even-3-Parity using 6 mutations and 3 function sets.

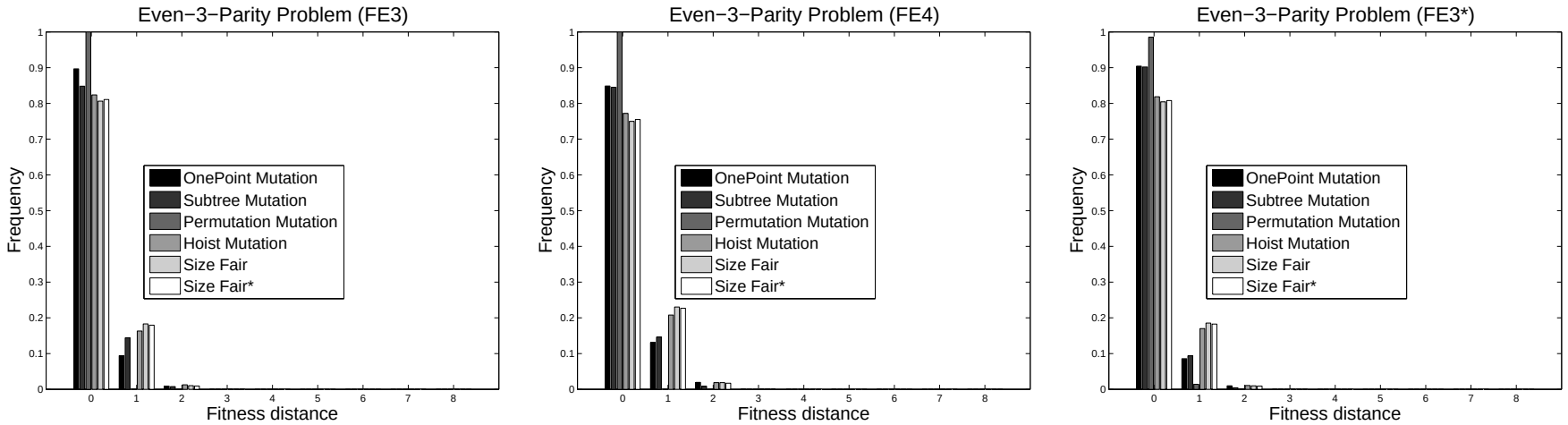


Figure : Standard function set (left), alternative function set (centre) and a function set for Uniform GP (right).

- Notice that when using the permutation mutation operator and using FE3 and FE4 on the Even-*n*-Parity problem, the *fd* is always 0 because all of the operators in these function sets are symmetric.

Even-4-Parity using 6 mutations and 3 function sets.

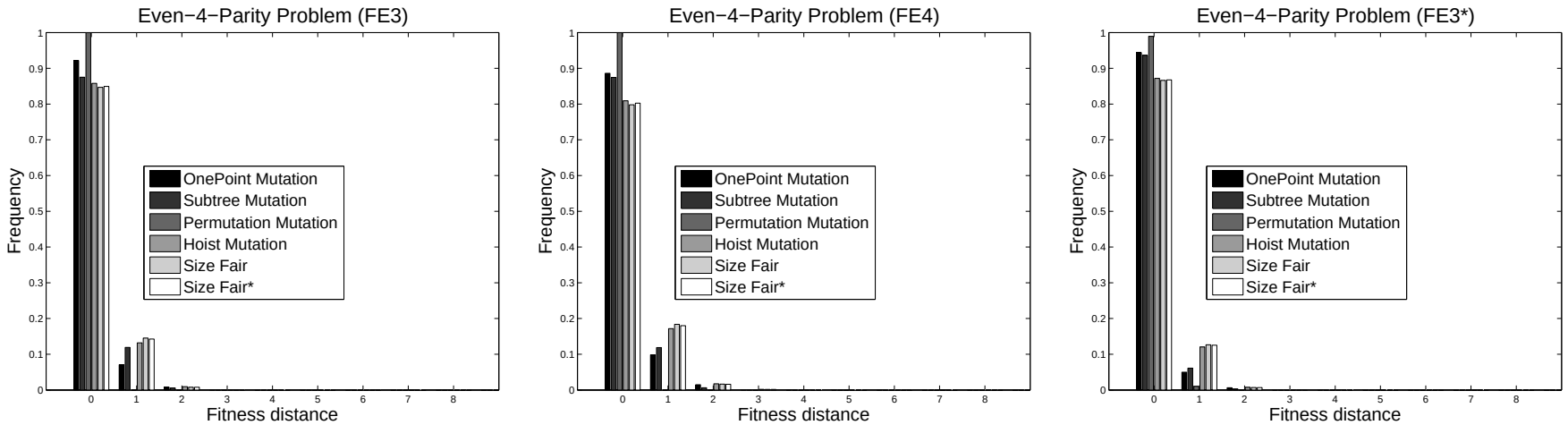


Figure : Standard function set (left), alternative function set (centre) and a function set for Uniform GP (right).

Artificial Ant Problem using 6 mutations and 3 function sets.

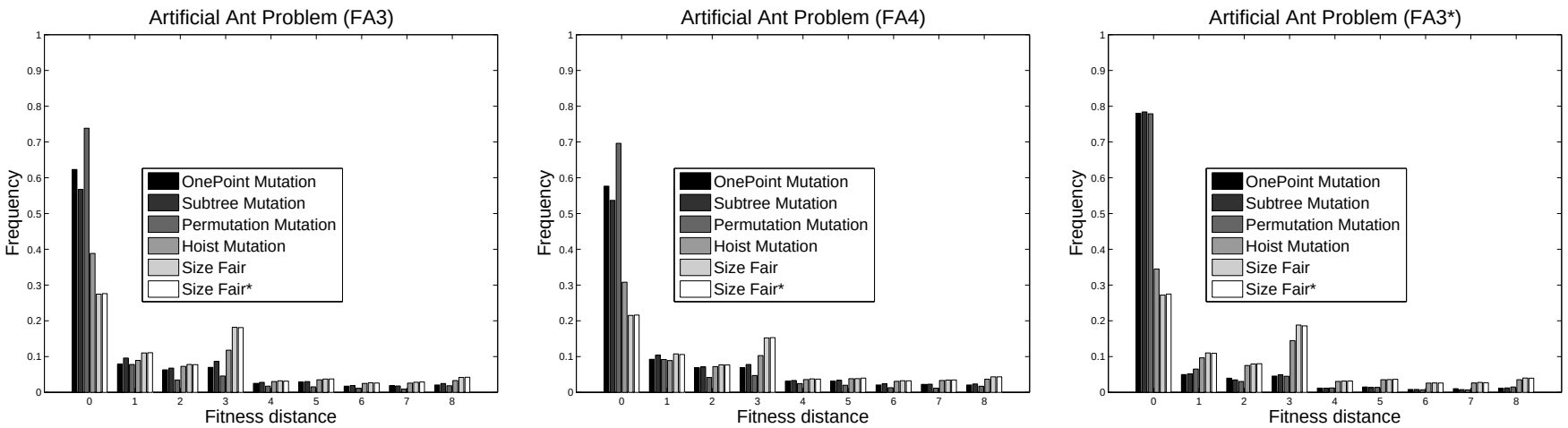


Figure : Standard function set (left), alternative function set (centre) and a function set for Uniform GP (right).

Symbolic Regression F_1 using 6 mutations and 3 function sets.

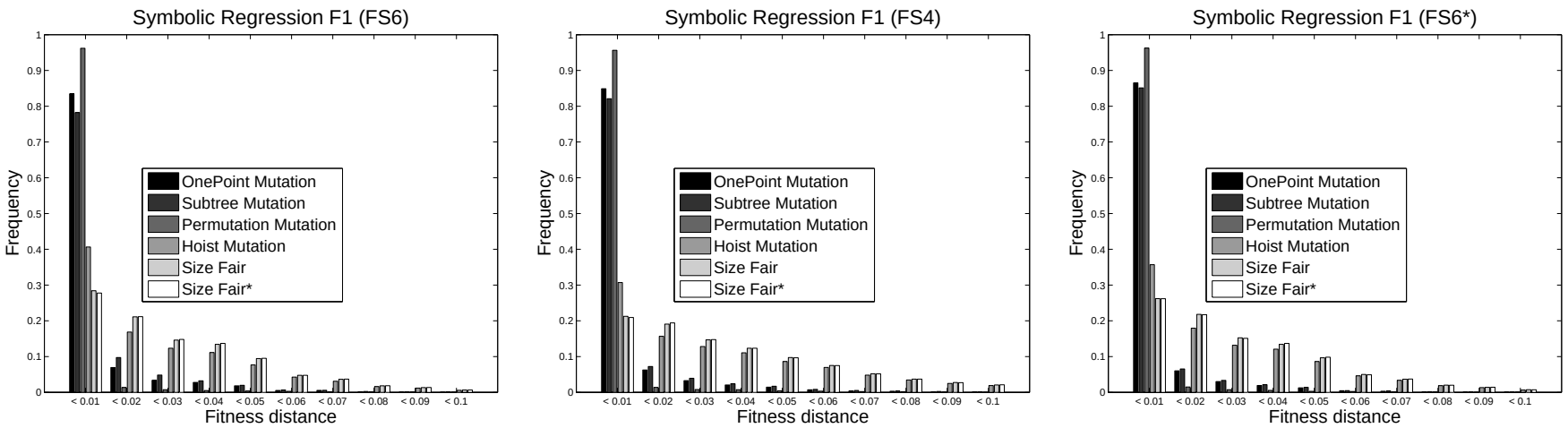


Figure : Standard function set (left), alternative function set (centre) and a function set for Uniform GP (right).

Symbolic Regression F_2 using 6 mutations and 3 function sets.

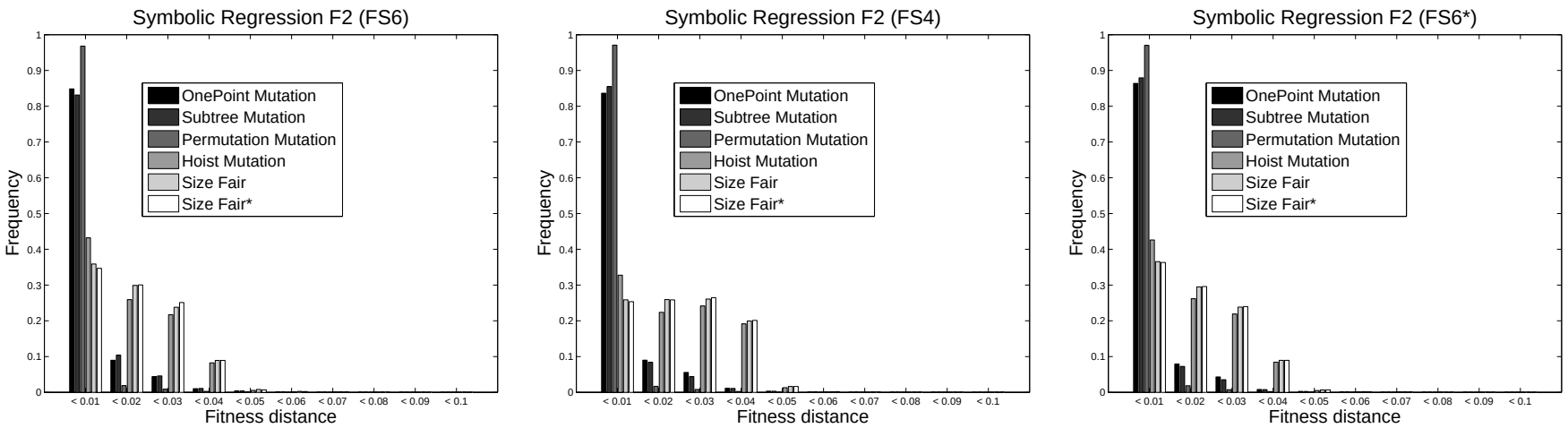


Figure : Standard function set (left), alternative function set (centre) and a function set for Uniform GP (right).

Parameters

Table : Parameters used to conduct our experiments.

Selection	Tournament (size 7)
Initial Population	Ramped half and half (depth 3 to 8)
Population size	200, 250, 500
Generations	125, 100, 50 (25,000 divided by population size)
Runs	100
Mutations	One Point, Subtree, Permutation, Hoist Size-Fair & Size-Fair Range
Mutation rate	One single mutation per individual
Termination	Maximum number of generations

- we performed $100 * 45 * 6$ runs in total[1].

[1] 100 independent runs, 45 different settings (i.e., three different combinations of population sizes and number of generations, five different problems and three different function sets for each of the problems - $3 * 5 * 3$), and 6 different mutation operators.

Locality and Performance on the Even-3-Parity Problem

Table : Locality values. Lower is better.

Mutation Operators	Def_0			$Def_1 1$			Cond.($Def_2 1$)		
	F_{E3}	F_{E4}	F_{E3*}	F_{E3}	F_{E4}	F_{E3*}	F_{E3}	F_{E4}	F_{E3*}
One Point	0.1125	0.1727	0.1059	0.9057	0.8699	0.9145	0.0091	0.0213	0.0102
Subtree	0.1599	0.1638	0.1026	0.8562	0.8541	0.9064	0.0081	0.0089	0.0045
Permutation	0	0	0.0154	1	1	0.9859	0	0	0.0006
Hoist	0.1902	0.2497	0.1932	0.8376	0.7936	0.8304	0.0139	0.0217	0.0118
Size Fair	0.2046	0.2710	0.2063	0.8173	0.7707	0.8150	0.0109	0.0209	0.0106
Size Fair*	0.1993	0.2641	0.2017	0.8207	0.7742	0.8179	0.0100	0.0191	0.0098

Table : Performance (measured in terms of average of the best fitness values over all runs). Numbers within parentheses indicate number of runs able to find the global optimum. Higher is better.

Mutation Operators	$P = 500, G = 50$			$P = 250, G = 100$			$P = 200, G = 125$		
	F_{E3}	F_{E4}	F_{E3*}	F_{E3}	F_{E4}	F_{E3*}	F_{E3}	F_{E4}	F_{E3*}
OnePoint	6.97	7.77 (81)	7.51 (64)	6.99	7.87 (88)	7.13 (28)	6.52	7.75 (81)	7.00 (18)
Subtree	7.41 (42)	7.76 (80)	7.63 (69)	7.28 (30)	7.85 (86)	7.15 (29)	6.86 (1)	7.66 (79)	7.09 (21)
Permutation	5.94	4.95	5.88	5.94	4.95	4.98	4.95	5.94	5.41
Hoist	6.00	5.15	5.03	6.00	5.06	5.00	5.16	6.00	4.00
Size Fair	6.00	5.03	5.00	5.99	5.03	5.00	4.98	6.00	4.98
Size Fair*	6.00	5.03	5.00	5.99	5.02	5.00	4.99	6.00	5.00

Number of correct predictions of *good locality* on performance for all the problems.

	One Point	*Subtree*	*Permut.*	*Hoist*	*Size Fair*	*Size Fair**	*Total*
	Even-3-Parity						
Def_0	0	0	-	2	2	2	6
Def_1	3	3	-	1	1	1	**9**
Def_2	0	0	-	0	0	0	0
	Even-4-Parity						
Def_0	1	1	-	1	0	1	4
Def_1	1	1	-	3	3	3	**11**
Def_2	0	1	-	0	0	0	1
	Artificial Ant						
Def_0	0	0	3	0	3	3	9
Def_1	0	0	3	0	3	3	9
Def_2	0	0	3	0	3	3	9
	Symbolic Regression F_1						
Def_0	2	2	0	0	0	0	4
Def_1	0	0	0	3	3	3	**9**
Def_2	2	2	0	0	0	0	4
	Symbolic Regression F_2						
Def_0	1	1	2	1	1	0	6
Def_1	3	1	2	1	1	1	**9**
Def_2	1	1	1	1	1	1	6

Number of correct predictions of *all locality values* on performance for all the problems

	F_{E3}, F_{A3}, F_{S6}	F_{E4}, F_{A4}, F_{S4}	$F_{E3*}, F_{A3*}, F_{S6*}$	Total
	Even-3-Parity			
Def_0	7	0	2	9
Def_1	3	9	3	**15**
Def_2	3	0	3	6
	Even-4-Parity			
Def_0	8	2	0	10
Def_1	5	7	11	**23**
Def_2	6	2	1	9
	Artificial Ant			
Def_0	7	12	9	28
Def_1	7	12	9	28
Def_2	7	12	9	28
	Symbolic Regression F_1			
Def_0	6	4	12	22
Def_1	10	16	12	**38**
Def_2	5	0	4	9
	Symbolic Regression F_2			
Def_0	3	3	7	13
Def_1	10	2	4	**16**
Def_2	4	2	5	11

Extending Locality to the Genotype-Fitness Mapping

- It has been argued that locality is a key element in performing effective evolutionary search of landscapes.
- In this talk, I have shown how it is possible to extend the original genotype-phenotype definition of locality to the genotype-fitness mapping, considering three different scenarios of fitness neighbourhood.
- This has been supported by developing a mathematical framework and performing extensive EC runs.

Locality VS. Non-Locality

- The results presented in this talk are consistent with those shown by Rothlauf in his original studies.
- For a particular problem, there is an agreement between the three definitions of locality and the performance achieved by the GP system.
- Def_1 (neighbours $= 0$) seems to be more accurate than the other two definitions of locality.

Limitations & Future Work

- A natural step to take from here is by considering the use of crossover.
- To fully complete the study of locality in "non-traditional" GP, it is necessary to study the genotype-phenotype-fitness space.

More on Locality

Details of this presentation can be found in [4]. Other revelant readings: [1, 5, 3, 2, 7, 8].

Acknowledgments

- Dr. James McDermott – MIT, USA / UCD, IE.
- Prof. Anthony Brabazon, Dr. Michael O'Neill – UCD, IE.

Bibliography I

[1] E. Galvan, L. Trujillo, J. McDermott, and A. Kattan.
Locality in continuous fitness-valued cases and genetic programming difficulty.
In O. Schtze, C. A. Coello Coello, A.-A. Tantar, E. Tantar, P. Bouvry, P. Del Moral, and P. Legrand, editors, *EVOLVE - A Bridge between Probability, Set Oriented Numerics, and Evolutionary Computation II*, volume 175 of *Advances in Intelligent Systems and Computing*, pages 41–56. Springer Berlin Heidelberg, 2013.

[2] E. Galván-López, J. McDermott, M. O'Neill, and A. Brabazon.
Defining locality in genetic programming to predict performance.
In *Evolutionary Computation (CEC), 2010 IEEE Congress on*, pages 1–8, 2010.

[3] E. Galván-López, J. McDermott, M. O'Neill, and A. Brabazon.
Towards an understanding of locality in genetic programming.
In *Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation*, GECCO '10, pages 901–908, New York, NY, USA, 2010. ACM.

[4] E. Galván-López, J. McDermott, M. O'Neill, and A. Brabazon.
Defining locality as a problem difficulty measure in genetic programming.
Genetic Programming and Evolvable Machines, 12(4):365–401, Dec. 2011.

[5] E. Galván-López, M. O'Neill, and A. Brabazon.
Towards understanding the effects of locality in gp.
In *Artificial Intelligence, 2009. MICAI 2009. Eighth Mexican International Conference on*, pages 9–14, 2009.

[6] E. Galván-López and R. Poli.
An empirical investigation of how degree neutrality affects gp search.
In A. Hernandez, R. Monroy, and C.Reyes, editors, *MICAI 2009: Advances in Artificial Intelligence*, Lecture Notes in Computer Science, pages 728–739. Springer.

Bibliography II

[7] J. McDermott, E. Galván-Lopéz, and M. ONeill.
A fine-grained view of gp locality with binary decision diagrams as ant phenotypes.
In R. Schaefer, C. Cotta, J. Koodziej, and G. Rudolph, editors, *Parallel Problem Solving from Nature, PPSN XI*, volume 6238 of *Lecture Notes in Computer Science*, pages 164–173. Springer Berlin Heidelberg, 2010.

[8] J. McDermott, E. Galván-Lopéz, and M. ONeill.
A fine-grained view of phenotypes and locality in genetic programming.
In R. Riolo, E. Vladislavleva, and J. H. Moore, editors, *Genetic Programming Theory and Practice IX*, Genetic and Evolutionary Computation, pages 57–76. Springer New York, 2011.

[9] F. Rothlauf and D. Goldberg.
Redundant Representations in Evolutionary Algorithms.
Evolutionary Computation, 11(4):381–415, 2003.

[10] S. Wright.
The Roles of Mutation, Inbreeding, Crossbreeding and Selection in Evolution.
In D. F. Jones, editor, *Proceedings of the Sixth International Congress on Genetics*, volume 1, pages 356–366, 1932.